School of Mechanical and Electrical Engineering

Faculty of Health, Engineering and Sciences

Enabling Peer-to-Peer Remote Experimentation in Distributed Online Remote Laboratories

A thesis submitted by

Ananda Maiti

For the award of

Doctor of Philosophy

Australia

2016

Abstract

Remote Access Laboratories (RALs) are online platforms that allow human user interaction with physical instruments over the Internet. Usually RALs follow a client-server paradigm. Dedicated providers create and maintain experiments and corresponding educational content. In contrast, this dissertation focuses on a Peer-to-Peer (P2P) service model for RALs where users are encouraged to host experiments at their location. This approach can be seen as an example of an Internet of Things (IoT) system. A set of smart devices work together providing a cyber-physical interface for users to run experiments remotely via the Internet.

The majority of traditional RAL learning activities focus on undergraduate education where hands-on experience such as building experiments, is not a major focus. In contrast this work is motivated by the need to improve Science, Technology, Engineering and Mathematics (STEM) education for school-aged children. Here physically constructing experiments forms a substantial part of the learning experience. In the proposed approach, experiments can be designed with relatively simple components such as *LEGO Mindstorms* or *Arduinos*. The user interface can be programed using *SNAP!*, a graphical programming tool.

While the motivation for the work is educational in nature, this thesis focuses on the technical details of experiment control in an opportunistic distributed environment. P2P RAL aims to enable any two random participants in the system - one in the role of maker creating and hosting an experiment and one in the role of learner using the experiment - to establish a communication session during which the learner runs the remote experiment through the Internet without requiring a centralized experiment or service provider. The makers need to have support to create the experiment according to a common web based programing interface. Thus, the P2P approach of RALs requires an architecture that provides a set of heterogeneous tools which can be used by makers to create a wide variety of experiments.

The core contribution of this dissertation is an automaton-based model (*twin finite state automata*) of the controller units and the controller interface of an experiment.

This enables the creation of experiments based on a common platform, both in terms of software and hardware. This architecture enables further development of algorithms for evaluating and supporting the performance of users which is demonstrated through a number of algorithms. It can also ensure the safety of instruments with intelligent tools. The proposed network architecture for P2P RALs is designed to minimise latency to improve user satisfaction and learning experience. As experiment availability is limited for this approach of RALs, novel scheduling strategies are proposed.

Each of these contributions has been validated through either simulations, e.g. in case of network architecture and scheduling, or test-bed implementations, in case of the intelligent tools. Three example experiments are discussed along with users' feedback on their experience of creating an experiment and using others' experimental setup. The focus of the thesis is mainly on the design and hosting of experiments and ensuring user accessibility to them. The main contributions of this thesis are in regards to machine learning and data mining techniques applied to IoT systems in order to realize the P2P RALs system.

This research has shown that a P2P architecture of RALs can provide a wide variety of experimental setups in a modular environment with high scalability. It can potentially enhance the user-learning experience while aiding the makers of experiments. It presents new aspects of learning analytics mechanisms to monitor and support users while running experiments, thus lending itself to further research. The proposed mathematical models are also applicable to other Internet of Things applications.

Certification of Thesis

This thesis is entirely the work of *Ananda Maiti* except where otherwise acknowledged. The work is original and has not previously been submitted for any other award, except where acknowledged.

Student and supervisors signatures of endorsement are held at USQ.

Acknowledgments

I would like to express my sincere gratitude to my supervisor Assoc/Prof. *Alexander A. Kist*, who has been a tremendous mentor for me. I would like to thank him for encouraging my research and for supporting my growth as a research scientist. His advice on both research as well as on my career have been priceless.

I would like to express my special appreciation and thanks to my co-supervisor Dr. *Andrew D. Maxwell* for his continuous support and guidance throughout my PhD study and related research.

I would also like to thank the members of the RALfie project: Dr. *Lindy Orwin*, Prof. *Peter Albion* and *Wu Ting* for their continuous support and help for the last three years.

This research has been supported through the Collaborative Research Network (CRN) program of the Australian Government. I would like to thank especially Marisa Parker and all other members of the CRN management at University of Southern Queensland for their support during this research.

Table of Contents

List of Abbreviations

Abbreviation	*Meaning*
CI	Controller Interface
CPL	Control Program Logic
CU	Controller Unit (of an experiment)
EBL	Enquiry Based Learning
IDE	Integrated Development Environment
IEM	Instruction Execution Module
IoT	Internet of Things
LMS	Learning Management System
MCU	Micro-Controller Unit
MDP	Markov Decision Process
MTBF	Mean Time Between Failure
MTBR	Mean Time Between Repair
MTTF	Mean Time To Failure
MU	Measurement Unit
P2P	Peer-to-Peer
RAL	Remote Access Laboratory
RALfie	Remote Access Laboratory for fun, innovation and education
RLMS	Remote Laboratory Management System
ROI	Real Object Identification
RTT	Round Trip Time
SNAP	* This is not an abbreviation, but a name.

STEM	Science Technology Engineering and Maths
UI	User Interface
UIM	User Interaction Module
VOC	Virtual Object Creation
VPN	Virtual Private Network
WoT	Web of Things

List of Symbols

Abbreviation	*Meaning*
S	*Finite State Machine for Controller Interface*
Y	*Finite State Machine for Controller Unit; also the MDP for the controller Unit*
γ	*Maximum set size parameter in algorithm CHAC; also Interactivity level*
δ	*Transition rules in Y*
β	*Transition rules in S*
ε	*Maximum distance parameter in CHAC*
ψ	*Network Latency between S and Y*
Σ_x	*Symbol set for FSM x*
π	*Control Policy in MDP*
λ	*'Null' or 'empty' in FSM; failure rate in reliability theory*
Φ	*closeness co-efficient in CRC*
θ	*Usage probability in CRC*
Ω	*Maximum distance parameter in CHAC2*
ϑ	*Decay rate of VIA*
R	*Set of ports on Y and recorded in S as well;*
ω	*Stack in Y*
σ	*Participation probability*
Ω	*Diameter constraint in CHAC2*
Γ	*NDC site arrays*
Π	*Selected NDC sites*
Ξ	*Reliability*

$\mathcal{Y}$	*Set of initial sites*
∂	*2nd Indicator - Relative distance in MDP*
ϖ	*3rd Indicator – Weighted Relative distance in MDP*
α	*Reward strategy in MDP*
$\mathbb{s}$	*set*
$\mathbb{r}$	*read command*
$\mathbb{w}$	*write command*
$\mathbb{u}$	*success*
$\mathbb{a}$	*wait*
$\mathbb{e}$	*error*
$\mathbb{f}$	*fail*
$\mathbb{l}$	*end*

Glossary of Terms

Term	*Meaning*
administrators	*Experts who creates and maintains specification regarding RLMS; same as makers/providers in centralized context*
client	*A computer machine PC or mobile device used by a person to consume online services*
developer	*Creator (and host in P2P RALs) of experiments*
experiment	*An experimental setup and it's activity including user interface accessible on the Internet*
latency	*The time taken for communication or round trip time between client and server in centralized context or CI and CU in P2P context*
learner	*One who accesses an experiment for learning the corresponding scientific concepts*
maker	*Creator and host of experiments including user interface*
rig	*An experimental setup*
server	*A computer machine that runs software to provide online services for 24x7 time*
session	*A time period in which an user or maker accesses and runs an experiment*
user	*One who uses the system for learning; same as learner*
RalfieBox	*A router that is programmed to act as gateway for the MCUs and cameras to connect to the RALfie*

List of Figures

Related Publications (2013-2016)

1. **A. Maiti**, A. A. Kist, A. D. Maxwell, and L. Orwin, *Stem Oriented Remote Laboratories With Peer-To-Peer Architecture*, accepted in Online Experimentation: Emergent Technologies & the Internet of Things, Eds: M. T. Restivo, A. Cardoso, and A. M. Lopes, Publisher – IFSA, Portugal, 2015.
2. **A. Maiti**, A. A. Kist, and A. Maxwell, *Real-Time Remote Access Laboratory with Distributed and Modular Design*, IEEE Transactions on Industrial Electronics, vol. 62, Iss. 6, 2015.
3. **A. Maiti**, A. D. Maxwell, A. A. Kist and L. Orwin, *Merging Remote Laboratories and Enquiry-based Learning for STEM Education*, International Journal of Online Engineering (iJOE), vol. 10. Iss 6, pp. 43-49, 2014.
4. **A. Maiti**, A. D. Maxwell and A. A. Kist, *Features, Trends and Characteristics of Remote Access Laboratory Management Systems*, iJOE, vol. 10. Iss 2, pp. 22-29, 2014.
5. **A. Maiti**, A. A. Kist and A. D. Maxwell, *Variable Interactivity with Dynamic Control Strategies in Remote Laboratory Experiments*, in International Conference on Remote Engineering and Virtual Instrumentation 2016 (REV 2016), Madrid, Spain Feb 24-26 2016.
6. **A. Maiti**, A. A. Kist and M. Smith, *Key Aspects of Integrating Augmented Reality Tools into Peer-to-Peer Remote Laboratory User Interfaces*, in International Conference on Remote Engineering and Virtual Instrumentation 2016 (REV 2016), Madrid, Spain Feb 24-26 2016.
7. A. A. Kist, **A. Maiti**, A. D. Maxwell, L. Orwin, P. Albion, W. Ting and R. Burtenshaw, *The Game and Activity Environment of RALfie*, in REV 2016, Madrid, Spain Feb 24-26 2016.
8. **A. Maiti**, A. A. Kist and A. D. Maxwell, *Building Markov Decision Process Based Models of Remote Experimental Setups for State Evaluation*, Computational Intelligence, 2015 IEEE Symposium Series on, Cape Town, 2015, pp. 389-397.
9. **A. Maiti**, A. A. Kist and A. D. Maxwell, *Latency-Adaptive Positioning of Nano Data Centers for Peer-to-Peer Communication based on Clustering*, IEEE International Conference on Communications 2015 - Workshop on Cloud Computing Systems, Networks, and Applications (CCSNA), London UK, 8-12 Jun 2015, pp. 9981-9987.
10. **A. Maiti**, A. A. Kist and A. D. Maxwell, *Components Relationship Analysis in Distributed Remote Laboratory Apparatus with Data Clustering*, IEEE International Symposium on

Industrial Electronics, Rio de Janeiro, Brazil, 3-5 Jun 2015, pp. 861-866.

11. A. A. Kist, **A. Maiti** and A. D. Maxwell*, Introducing RALfie – Remote Access Laboratories for Fun, Innovation and Education*, IEEE exp.at 2015, Azores Portugal, 1-4 Jun 2015.
12. **A. Maiti**, A. D. Maxwell, A. A. Kist and L. Orwin, *Joining the Game and the Experiment in Peer-to-Peer Remote Laboratories for STEM Education*, IEEE exp.at 2015, Azores Portugal, 1-4 Jun 2015.
13. L. Orwin, A. A. Kist, A. D. Maxwell, **A. Maiti**, *Using Gamification to Create Opportunities for Engagement, Collaboration and Communication in a Peer-to-peer Environment for Making and Using Remote Access Labs*, IEEE exp.at 2015, Azores Portugal, 1-4 Jun 2015.
14. **A. Maiti**, A. D. Maxwell and A. A. Kist, *Design and Operational Reliability of a Peer-to-Peer Distributed Remote Access Laboratory*, in Remote Engineering and Virtual Instrumentation (REV), 2015 12th International Conference on, 25-27 Feb. 2015, pp. 94 - 99, Bangkok.
15. **A. Maiti,** A. A. Kist and A. D. Maxwell*, Time Scheduling in a Peer-to-Peer Remote Access Laboratory for STEM Education*, in IEEE TALE 2014, Wellington, pp. 1-7.
16. **A. Maiti,** A. D. Maxwell, A. A. Kist, and L. Orwin, *Integrating enquiry-based learning pedagogies and remote access laboratory for STEM education*, in IEEE EDUCON 2014 Istanbul, 2014, pp. 706-712.
17. A. A. Kist, **A. Maiti**, A. D. Maxwell, L. Orwin, W. Midgley, K. Noble, et al., *Overlay network architectures for peer-to-peer Remote Access Laboratories*, in Remote Engineering and Virtual Instrumentation (REV), 2014 11th International Conference on, 2014, pp. 274-280.
18. **A. Maiti**, A. A. Kist and A. D. Maxwell, *Using Network Enabled Microcontrollers in Experiments for a Distributed Remote Laboratory*, in REV 2014, pp. 180 – 186.
19. **A. Maiti**, A. A. Kist and A. D. Maxwell, *Estimation of Round Trip Time in Distributed Real Time System Architectures*, Telecommunication Networks and Applications Conference (ATNAC), 2013 Australasian, 20-22 Nov. 2013, pp. 57 - 62.
20. **A. Maiti**, A. D. Maxwell and A. A. Kist, *An Overview of System Architectures for Remote Laboratories,* in IEEE International Conference on Teaching, Assessment and Learning for Engineering (TALE) 2013, pp. 661-666, 2013.
21. A. A. Kist, **A. Maiti**, A. D. Maxwell and L. Orwin, *Performance Evaluation of Network Architectures for Collaborative Real-Time Learning Systems*, 2013 IEEE International Conference on Teaching, Assessment and Learning for Engineering (TALE), Aug 2013, pp.673-678.

1

Introduction

Laboratory education and practice are integral parts of the engineering education curriculum. The combination of theoretical knowledge along with practical experience linking the concepts is essential. They are also a requirement of accreditation bodies such as Engineers Australia (EA). The theoretical delivery commonly consists of lectures and exercises supplemented by textbooks and lecture notes. Practical experience is gained through interaction with real technical instruments and devices that exhibit real phenomenon as described in the theory.

Recent developments in Information and Communication Technology (ICT) have enabled fast and rich ways of exchanging information between people from different domains with a variety of applications. Such improvements in ICT and its infrastructure have enabled the development of Remote Access Laboratories (RALs). The development and use of RALs was identified as a trend in engineering and science education aiming to allow remote, off-site and organized use of real experimental equipment and resources [1]. These laboratories allow students to use the Internet to change input parameters, operate instruments and collect resultant data from equipment setups in remote locations.

RALs primarily fulfilled the role of on-site laboratories where needed in the early years of their development from 1990s to mid-2000s [2]. Over the last decade, modern RAL systems have further enhanced the pedagogies for laboratories by incorporating advanced ICT technologies such as augmented reality and providing a unique educational experience to students.

RALs have been a successful paradigm in providing an alternative platform to practical education in on-site laboratories [2-4]. A Remote Laboratory Management

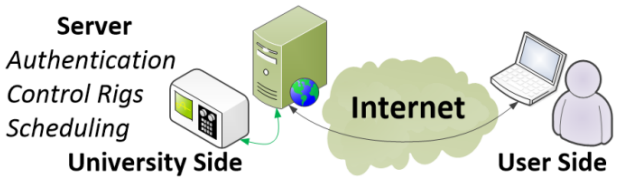


Figure 1.1. The basic centralised architecture of RAL where instruments and RLMS are hosted at the server side.

System (RLMS) is used to manage such RALs. The common functionalities of the RLMS include: scheduling, rig operations, data transport, multimedia tools, data about experiments, experiment user interface, accepting and processing user requests, storing and maintaining user details [5-6]. Figure 1.1 shows an example of a typical client server setup of remote laboratories. The user side connects and runs experiments while the RLMS on the server side is responsible for management functions. Such facilities are usually hosted by universities.

Remote laboratories can be of two types: real hardware based or simulated/virtual laboratories. Instead of using any real experimental apparatus, virtual laboratories use specialized software for experimentation. The real hardware laboratories use physical equipment for experimentation. In the fields of science and engineering, real-hardware based laboratories are common and suitable in many cases as they provide realistic feedback and data. This work focuses on real hardware based RALs only.

Remote laboratories have their origins in efforts to provide remote access to expensive equipment, such as that used in control engineering [2], as early as 1993-95. In Europe, early examples of such projects include Remote Experiment MOnitoring and conTrol (REMOT) project [7] and DYNAmical COnfigurable Remote Experiment Monitoring & Control System (DYNACORE) [8] in late 1990s. Since then many more systems have been deployed in universities around the world, some of the prominent RLSM being iLab [9], Labshare [10], WebLab Duesto [11] and hardware system VISIR [12].

The iLab is a flexible software infrastructure for the implementation of Internet accessible labs at MIT, USA [9, 13]. It uses many programming languages including LabVIEW to operate the instruments through its web servers. The University of Queensland, Australia later extended some of its features [14].

At the University of Technology Sydney (UTS) an RLMS was developed and used from 2000 to 2005. This system came to be known as SAHARA. It was adopted as part of the much broader Labshare project [10]. The Labshare project focuses on collaboration between several Australian institutions including the University of South Australia, University of Technology in Sydney, Curtin University of Technology in Perth, Queensland University of Technology in Brisbane and the Royal Melbourne Institute of Technology. The SAHARA framework provides a generic set of tools for setting up heterogeneous remote laboratories of physical instruments.

Virtual Instrument Systems in Reality (VISIR) was developed at Blekinge Institute of Technology Sweden. It is an online workbench which acts as an open laboratory platform [12]. The objective of the VISIR project is to create a lab community consisting of several participant universities and organizations. This has been used to implement online electronics laboratories at other locations for example at Universidad Nacional de Educación a Distancia (UNED), Madrid.

Figure 1.2 shows an example of a RAL experiment. Figure 1.2 (a) depicts the experiment site and (b) shows the remote users site. The experiment is composed of the corresponding User Interface (UI) stored in the RLMS and is used for taking input and displaying output. It is downloaded to the user's site every time a session is started. The experiment also contains the corresponding experiment controller,

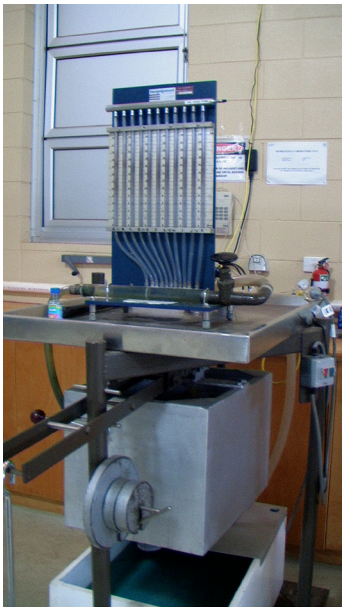


Figure 1.2 (a) An experimental rig

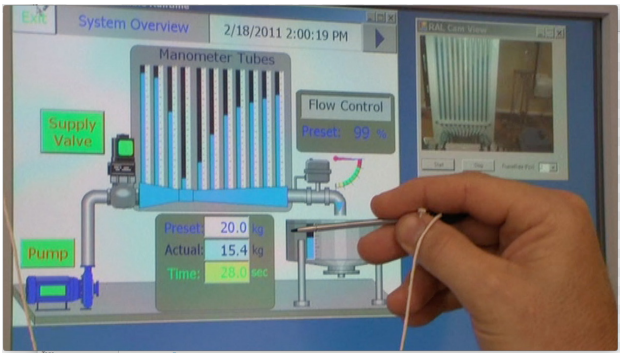


Figure 1.2 (b) A user interface

usually a personal computer or microcontroller. It acts as an intermediary between the user and instruments and the measurement and operational rig itself.

Undertaking a laboratory activity in RALs usually involves three broad steps. First, similar to face-to-face laboratory classes, the student peruses related learning materials describing the aim of the experiment and the underlying concepts. In face-to-face laboratories the next step would be to setup the experimental apparatus sometimes with minor configurations for example a semiconductor laboratory [15] and sometimes completely creating it from basic parts, for example electronics laboratories [12]. In RALs, the rigs are already prepared and ready to use at any time.

In the second step, the student issues commands to the experimental rig through the user interface on the Internet which is specifically designed for the experiment which responds to the user's command. Depending on the type of laboratory, students may have to learn how to use the instruments in an on-site laboratory and determine ways to record measurements. This is generally not required in the case of remote laboratories that often have rich user interfaces.

Finally the student verifies the results obtained from the experiments to understand the underlying learning concept and meet the objectives of the activity which they convey in a lab report. This step is similar in both RALs and on-site laboratories.

RAL systems have been successful in their intended objectives of providing access to resources along with additional services. Advantages of these systems include access from anywhere and anytime, allowing more students to gain access, the safe running of experiments, sharing of resources among universities and technical support and are available as and when needed.

Two key aspects of remote laboratory experiments are their duration and the User Interface. Based on the time intervals between user commands, experiments can be broadly classified into *interactive* and *batched*. Whilst *batched* experiments are easiest to implement and maintain, *interactive* experiments offer richer learning experience [16]. Although the focus of this dissertation is on interactive remote laboratory experiments, batched experiments are also addressed.

1.1 STEM Education and Remote Laboratories

There is a worldwide skill shortage of high school graduates with sufficient Science, Technology, Engineering and Mathematics (STEM) skills[17]. Insufficient numbers of school students developing and maintaining an interest in STEM fields while at school is one of the contributing factors. In Australia, for example, student engagement and participation rates in STEM in secondary schools are low [17-18]. Primary school teachers, and some secondary teachers who are teaching outside their content area, especially in remote area schools, have low levels of content knowledge and pedagogical content knowledge in STEM [19].

ICT enrolments in tertiary courses have experienced negative growth in recent years. As a consequence Australia may not have the skilled workforce to sustain future productivity and economic growth [20]. In response, in recent years, the Australian Government has committed resources to increase student uptake of STEM subjects in primary and secondary schools across the country. One of the key focus areas is ICT skills such as coding [21].

STEM students who engage in experiential learning through the use of experiments develop deep understanding of content [22]. However, students do not all have equal opportunities to participate in hands on experiments in STEM [23]. One way of providing more support for STEM teachers and increase access to experiments for learners is to use RALs. Although RALs have been used in tertiary education for many years [24-25], it is only recently that these facilities have been made available to schools through projects such as Labshare [9] in Australia and GoLabs in Europe [26-27].

The pedagogies for school students in years 5-12 are diverse and differ from pedagogies used in tertiary education [28]. One of the limitations of the traditional RALs for its application in STEM school education is that it only allows experienced and expert developers to create an experiment which reduces scalability i.e. the number of experiments. The instruments and devices used are often costly and complex to build and operate. Also, there is limited scope for collaboration among students.

1.2 RALfie – Remote Access Laboratories for fun, innovation and education

The educational disciplines of science and engineering typically require learners to demonstrate proficiency in bridging the theoretical and experimental world. As part of these experiential learning experiences, RALs can be used for demonstrations of actual events and experiments.

RALfie (Remote Access Laboratories for fun, innovation and education) has been a three year project funded through the Collaborative Research Network initiatives of the Australian Government. It has proposed a Peer-to-Peer (P2P) environment at a conceptual level for the deployment of remote access laboratories where users create lab activities and associated programs and share them through the Internet. The scope of the RALfie project has been to establish the technology requirements and specifications of such a RAL system and implement it to determine the pedagogical advantages and effect in STEM education. The work reported in this dissertation has been the technical foundation for the architecture of the RALfie project.

A P2P system such as the *RALfie* project can overcome some of the limitations of traditional RALs. Participants in it can be both creators of experiments (called *Makers* in the project's agreed terminology) and share them with others or be user of others' experiments. Once individuals are authorised to develop and host an experiment, it can create more flexibility on the laboratory provider side. The students using these laboratories may collaborate with each other on running the experiment setup thus giving the users fresh views of the same problem. This way, new and interesting ideas about practical learning and enquiry-based learning methodology may be implemented.

In the field of Computer Science, Peer-to-peer (P2P) *computing* or *networking* generally refers to a system with multiple individual nodes each of which can be both servers i.e. provide data and be clients i.e. consume data. Such networks should not ideally have a centralized node, the failure of which could cause the network to break down. Most P2P software are focused on media sharing and P2P is therefore often associated with piracy and copyright violation regarding large files. Some P2P networks aim to provide real-time services with live call facilities such as Skype. These aim to provide direct communication between two nodes where the content is generated in real time, although it can allow for lossy communication.

Different forms of social media can also be described as P2P social process [29-30] as they enable direct communication between any two participating nodes where both sides can generate and exchange data. However, in such cases, the participating nodes do not have the responsibility of storing the data.

The term *P2P* in this thesis refers to a *consumer* level system similar to social networks, instead of a *P2P computing* or *network* architecture, that provides an online platform where two participants can communicate and exchange ideas or other resources. Consequently, the actual users' needs greatly influences the P2P RAL technologies. The participants can be both a server node and client node while in the system. The unique challenge here is that the nodes needs to host not only data as files, but physical hardware that must be programmed to run on the internet with real time commands. Thus the most important aim of this P2P system is to enable the creation of the participant nodes with potentially unique individual features such that any two nodes can still communicate and operate. Similar to Skype, the aim of this P2P system is not efficient storage, but to simply enable communication where the messages are generated and exchanged in real time. Obviously the P2P RAL system needs to run on a network architecture which may or may not be a true P2P network.

Thus the P2P RAL can be described as a system where two random participants in the system consisting of one maker creating and hosting an experiment and one learner who wants to use the experiment, can establish a communication session during which the learner runs the remote experiment through the internet without requiring a centralized experiment or service provider. Figure 1.3 depicts a P2P RAL system with

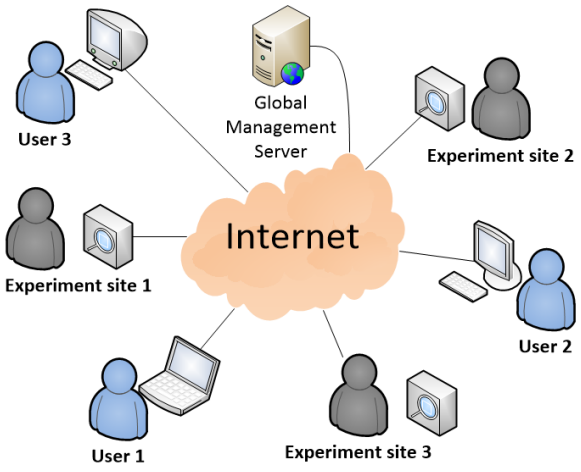


Figure 1.3. P2P RAL system

multiple maker experiments and user sites with a global management system to control access and authentication is also shown. The global management system only provides the links for the online experiments at a given time to the learner's user interface. Once the learner's UI has initiated the communication with the remote experiment, there is no role for the management server. The global management system is transparent with respect to running an experiment. It is essentially a portal for the makers to start using the system. It may be noted that although the P2P RAL systems ideally should run on a *true* P2P network, it is not feasible to create and maintain such a true P2P network and the P2P RAL needs real-time, but lossless communication in a an ad hoc network.

The P2P RAL enables teachers and students to create and maintain their own experimental rigs using hardware and software that may be acquired commonly such as Micro-Controllers Units (MCU). With this, the P2P RAL system aims to bring both the experiment building and operating experience close to the participants. The RLMS implemented on a global management server has reduced functionality compared to a centralized RLMS to provide a set of tools that enables the sharing and collaboration.

The operation of the P2P RAL is depicted in Figure 1.4. The entire system is made up of three conceptual layers – the *organization* layer, *participants* layer and the *systems* layer.

The *organisation* layer targets several objectives, the key being to motivate the students to use the system. It also maintains a structural framework within the set of experiments. It classifies experiments into groups and associates each of them with a certain category which may be related to the level of difficulty or the subject area. It creates the logical links that allow students to look up each other's experiments.

This layer is largely outside the scope of this dissertation. Instead, this thesis is focused on the two underlying layers that can enable the operation of the organization layer. The use of the P2P RAL in STEM Education and in particular this organization layer is further discussed in Chapter 11.

The *participants* layer represents the actual students in the system. There are three types of participants involved in the system:

- *Learners* (*users*): These participants use the system for learning purposes only.

They log in to the system, change experiment parameters and explore outcomes to gain knowledge.

- *Makers* (*developers*, *providers*): These participants share their equipment over the Internet. They assemble rigs, program them and create the user-interface that is accessed over the Internet. In a P2P RAL makers are responsible for making the rigs as developers of the experiments as well as providers when hosting the experiment for others.

  It is noteworthy that makers in a P2P RAL create experiments but they are still *consumers* of the system. Their interaction with the system forms part of their learning outcomes and must be supported by the RLMS. The makers need to have support to create the experiment according to a common web based programing interface. Thus, the P2P approach of RAL requires an architecture that provides a set of heterogeneous tools which can be used by makers to create a wide variety of experiments.

- *Moderators*: A third group of participants is required to assess the quality of experiments and the accuracy of content that are shared. Teachers, for example, can do this.

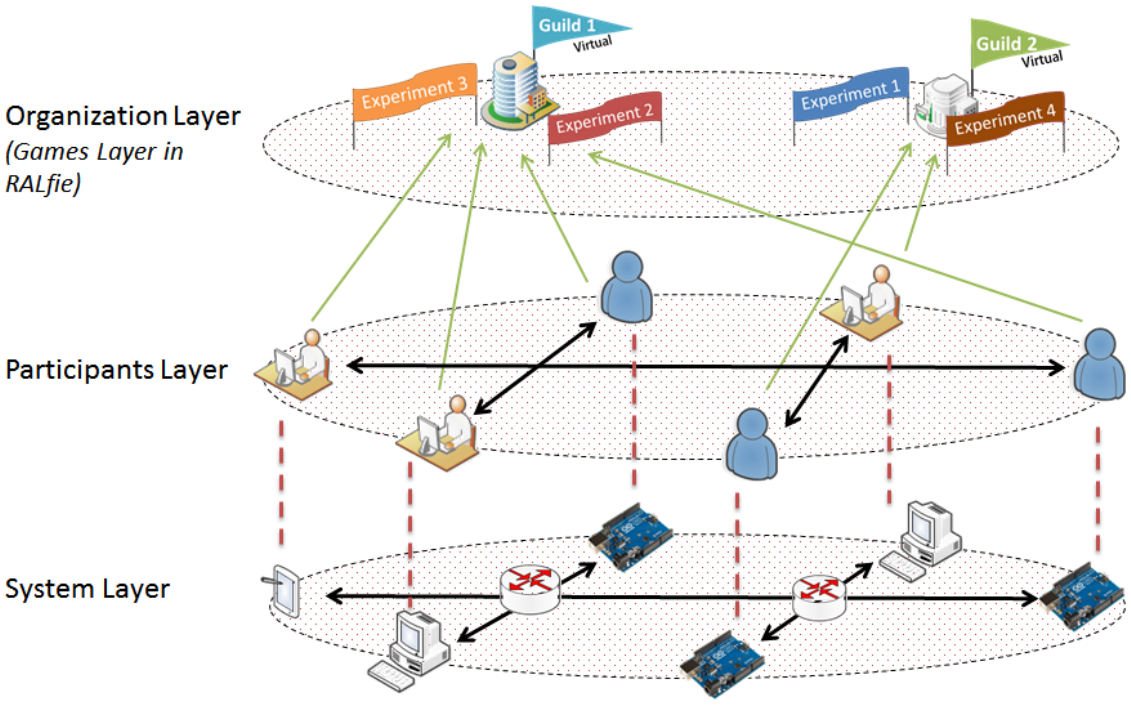


Figure 1.4. The RALfie system architecture

Apart from these three roles, there are the *administrators* responsible for creating and maintaining the online programming environment and related tools with which the makers can create the experiments.

The ratio of *makers* to *learners* may be very low as the number of students able to successfully fulfil the role of *makers*, may be low. However, even if a small percentage of users create and share equipment, it can be used by many others thus potentially inspiring them in the subject matter.

The *systems* layer is the bottommost layer that provides connectivity between users and the ways to control the equipment. The P2P RAL follows a P2P service model rather than actually implementing a real P2P network. It enables communication between any two random sites with their human participants without the need for a centralized service provider. This P2P RAL service model would ideally be built upon a self-sustaining P2P network system. However, this is not practically possible as most structured or true P2P networks are not scalable for large scale real life implementation [31]. Thus the implementation of the P2P follows a hybrid of P2P concepts enabling the end-to-end connection directly with a transparent service provider in between that only relays the commands and data of the experiments. This transparent service provider is not responsible for creating experiments or generating the commands or data for an experiment in any way. Thus, at a conceptual level, users communicate in a peer-to-peer manner, however, this may not be reflected by the underlying network architecture as discussed in later chapters.

1.3 Challenges of a P2P RAL

Within the *RALfie* project and a P2P RAL in a larger context, there are two broad areas of challenges for developing and using such an environment - pedagogical and technical.

1.3.1 Pedagogical Challenges

The *organisation* layer is about addressing the pedagogical need of the RALs application in STEM. Three main pedagogical areas are engagement, collaboration and building rigs.

Engagement deals with the ways to motivate students to use the RAL system. The

organization of the experiments can be implemented in many ways such as a gamified learning environment [32] which was deployed in the RALfie Project. Such a system has a series of activities or quests that must be completed in sequence to earn badges or experience points [33]. A collection of experiments can provide easy searching of topics for the learners. Guilds are groups of makers sharing interest in a particular topic of STEM who provide a community of practice to support each other and new makers. This kind of system provides motivation to continue and engage while learning the corresponding STEM concepts.

Second, collaboration must be encouraged between participants. The availability of experimental rigs designed by peers can encourage others to survey them. It could then potentially draw them into creating their own rigs. The procedures to create and program rigs could be shared as plans and guides in text, photographic or video format.

Third, the building of rigs aspect of the project aimed to establish the best practices to help participants create and use the rigs with community support.

Details of the pedagogical requirements are out of scope and not directly addressed in this thesis. However, these requirements impacted design choices that were made with regards to the technical challenges addressed in this thesis.

1.3.2 Technical Challenges

The proposed distributed RAL architecture to address the pedagogical requirements with regards to STEM poses technical challenges which are identified and addressed in this dissertation.

Experiment Control, Automation and Programming

Each experiment has two end-nodes - one has a remote controller interface at the learners' side and the experiment control unit is located at the maker's side. Unlike a centralized RAL, the design and construction of an experiment is not known to the P2P RLMS in a P2P RAL. The hardware required for constructing the experiment rig may be of varied types and capabilities, must be easily available and must be able to parse a common set of instructions even if the native operations of experiment controllers i.e. MCUs are different. The P2P RAL must be able to deduce every

experiment as a common model such that a common control language and platform can be provided. This homogeneity is required to enable large scale collaboration and increase scalability which is the aim of this type of distributed RALs.

While running an experiment, the learners' commands must be validated to ensure the rig safety. Also, in case of P2P RAL, the series of commands must be automatically evaluated as well. The makers are not expected to implement all the evaluation and support tools as in centralized RALs. A common model is also required to analyse and support makers when creating the experiments as well as user interactions with the experiments.

Connectivity, Authentication and Security

The P2P RAL uses a network architecture that allows each pair of learner-experiment nodes to communicate directly with a possible transparent management node in between. This resembles a true P2P architecture but is more like an unstructured P2P network. The main concern in the P2P RAL is the latency between the nodes. The system supports users from various locations with different kind of devices. Experiment makers are expected to construct a rig, program it to able to connect to the network and finally other users should be able to connect and control it over the Internet. The network capabilities available to the users are different with firewalls and Network Address Translators (NAT) segregating users into specific domains. Unlike traditional RALs, these experiments are not expected to be online continuously as dedicated equipment may not be available, thus the experiments in the system will change dynamically with time. Finally, for running activities properly both communication and end node systems must be responsive. The P2P system must provide authentication of users to ensure the security of the experiments.

Experiment User Interface Design

Apart from creating the rigs, the maker must also create the user-interface for the experiment. This interface must be able to automatically integrate into the RAL system and deploy the programming paradigms along with the communication protocol for experiment control.

1.4 Scope of the Thesis

From the above discussions, it is clear that the P2P RAL research is multi-faceted and has different levels. The broad focus of the thesis is RAL which is based on online engineering principles. The proposed P2P RAL is a new type of RAL distinctly different from the client server or the federated RAL architectures. P2P RAL itself has multiple research aspects including its role in changing STEM education to include the designing and making of experiments as opposed to only using experiments in particular; the technologies to implement and use the P2P RAL; and the aspects of making an experiment in the P2P RAL. Within the technologies for enabling the P2P remote experiments in the RAL, there are two distinct but intertwined issues - the control of an experiment and the underlying network. Within the control of the experiments, three major issues will be addressed in turn with a strong technical focus - evaluation, validation and guidance of the participants and the experiments in the P2P RAL. These three issues are the core research issues in this thesis that enable the P2P RAL to achieve its ultimate goals with respect to education.

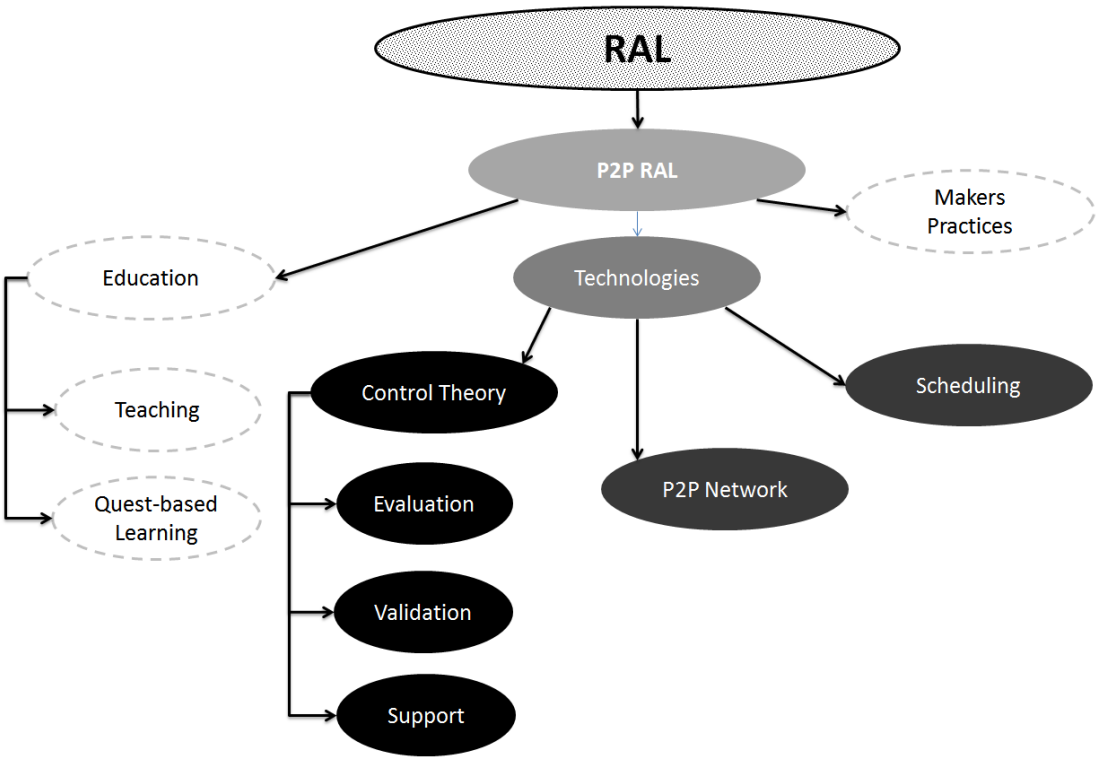


Fig 1.5 The research aspects of the P2P RAL. The core contributions of this work are in the areas depicted by the black leaf nodes of this tree.

1.5 Summary of Contributions

The main contribution of the thesis with respect to RALs is the concept, design and architecture of a distributed Peer-to-Peer RAL. In order to realize this architecture, further research in the aspects of control systems with data mining and machine learning led to the following major contributions in the technical aspects:

- Identifying and addressing issues regarding end node design with an automaton based architecture that is directly implementable with microcontrollers. The automaton provides a generic mathematical model of the controllers and their communications.

- The generic model is then used to propose several technical methods to analyse, support and enhance makers and user experience in a generic platform that is applicable for multiple experiments to be created by makers automatically. The generic model is also applicable to determine the architecture of end nodes in many IoT applications as well.

- Methods to optimize network performance. Round trip time or latency using clustering algorithms have been proposed. The latency is an important factor to ensure a good user experience. A clustering based routing architecture that can ensure availability of experiment related data when required in the P2P RAL's network system is proposed and validated through simulation. This method may be applied in other IoT applications as well.

- The reduced and dynamic availability of experiments require a new scheduling approach for users' access to rigs. Thus a new RAL scheduling mechanism based on availability of equipment is proposed.

- The P2P architecture has different aspects that can fail and affect user access to the experiments. A method to measure reliability of components of rigs, controllers and network is presented to determine the probability of failure of an experiment. A basic form of the reliability measurement method could be applied to determine the reliability of other IoT systems.

These contributions along with their role in the P2P RAL system are further discussed Section 3.6.

This thesis focuses mainly on the technical components such as algorithms and network architecture to create the necessary tools to enable the users to create and incorporate experiments into the P2P RAL system. A detailed study on the actual impact of these tools is not within the scope of this thesis, although some user experience results are reported proving usability of this architecture and its components.

1.6 Thesis Outline

The remainder of the dissertation is organized in following chapters discussing and addressing the individual aspects of the P2P RAL architecture:

Chapter 2 presents the literature review focused on the RAL and the IoT. It provides the context and motivation for the new architecture. Specific literature reviews that relate to individual research questions are discussed in relevant chapters.

Chapter 3 provides the overarching description of the P2P RAL system. It outlines the research questions, discusses how the solutions can work for P2P RAL and how the different aspects of this thesis related to each other. It also states detailed contributions of this dissertation.

Chapter 4 discusses the P2P control system architecture and it introduces a generic experiment model.

Chapter 5 provides a comparative analysis of different hardware that can be used to implement the generic model. It also introduces a prototype system based on the model including discussions on the commands required and their performance analysis.

Chapter 6 introduces an intelligent tool that enables the RLMS to validate commands and support and evaluate user/maker performance or interactions with the experiment.

Chapter 7 presents an extended intelligent tool for advanced evaluation of the users' interactions.

Chapter 8 presents a method to create an adaptive user interface with variable

interactivity based on the concept of experiment interactivity *continuum* which can enable the RLMS to enhance the user-experiment interaction.

Chapter 9 introduces the networking architecture of the P2P RAL system. It focuses on a model to evaluate average system latency and methods to improve the quality of experience of users by minimising end-to-end delay.

Chapter 10 discusses reliability issues of IoT and P2P RAL systems. It addresses how reliability can be measured in this context. A comparison between traditional RALs systems and P2P RAL is discussed.

Chapter 11 provides details on how P2P RAL relates to STEM education along with the RALfie instrumentation platform. Sample experiments are discussed and feedback from user trials with the system is presented.

Chapter 12 investigates how augmented reality can be included in a P2P RAL system. A set of generic tools are introduced that allow for simple augmentation and are based on the generic model.

Chapter 13 presents a scheduling mechanism based on the unique properties of the P2P RAL. This is necessary as experiment nodes are not available all the time and some activities need to be completed in a predefined sequence.

Chapter 14 discusses the conclusions of this work.

The following chapter covers the related literature review in detail.

2

Literature Review

> *This chapter discusses the current state of remote laboratories, their components and effectiveness with respect to STEM education. It also covers the Internet of Things and its influence on the RAL architecture from a peer-to-peer perspective.*

As stated in Section 1.4, the research presented here is multi-faceted. Thus, it is not feasible to discuss a literature review encompassing all aspects of the research aspects in this one chapter. This chapter focuses on two broad issues regarding the RALs that provide the contexts and constraints of the research – nature of RALs and IoT. The contents of this chapter lead to the formation of the research questions in the next chapter.

The architecture and impact of RALs have been widely reported and investigated. This chapter focuses on a detailed review of the RAL systems and establish their suitability for STEM Education. First, the components of an experiment session with respect to an RLMS are described. Some of the prominent RAL systems are analysed. The literature review presented here is not exhaustive with respective to all aspects of remote laboratories. Instead it focuses on the characteristics related to STEM education and the characteristics addressed in this thesis. Third, remote laboratories are compared with IoT to establish P2P RAL as an IoT system which lays the foundation of the P2P RAL architecture.

The specific literature reviews of the different aspects addressing the research questions are discussed in individual chapters.

2.1 Remote Access Laboratories

Within RALs there are traditionally two nodes: the server and the client. The user side consists of the students engaging and learning from use of the experiment, with the server side providing the experiment *rig*, as well as the experiment designers responsible for designing, creating and maintaining the experiment designed to allow experiential learning of concepts and learning materials. RLMS are responsible for arbitrated interaction between all components and interfaces in the system. Typically RLMSs have certain common components:

Scheduling: This aspect of RALs is well-investigated –in remote laboratories. The scheduling aspect highlights the difference between on-site and remote laboratories. Because online users are unaware of each other's activities within a system, interactions with the experiment hardware needs to be coordinated. RLMSs have addressed this concept in different ways [34]. There are two fundamental strategies used: queuing; and time-slotted booking [35]. In some RAL systems where only brief interactions between users and rigs are required, a reservation mechanism is used where users are presented with links to the experiment on a first-come-first-serve basis.

Rig operations: An experimental rig typically consists of a group of devices or instruments under local or remote computer control. The RLMS then makes experiment requests of this system, both sending commands, and then receiving collected data. This involves setting up a connection between these subsystems, and following a particular format for data handshake exchange.

Network access: This is the communication link layer between the user interface and the back-end instrumentation server for example HTTP or Remote Desktop Protocols.

Multimedia tools/data about experiments: Any information system for e-learning must provide documentation regarding the context of the experiment. Many RLMS provide tools to view or analyse data obtained back from an experiment. Often live video feedback is necessitated to observe in real-time the feedback within the experiment. For certain types of experiments this visual feedback may be an important or critical means of obtaining experimental data to

for example the mechanical and control theory laboratories.

Experiment user interface: Users interact with the experiment typically through either a web browser, or a browser based thin-client, or in some cases a standalone application [36]. These UIs allow the users to observe, interact and control the test equipment, as well as acquire the data or results.

Accepting and processing user request: Experiments used for undergraduate and graduate laboratories should have limited controls on the types of inputs that can be accepted. As such, the system needs to prevent improper inputs from damaging the equipment such as an electrical short circuit [12, 37] or high excessive voltage on components. Hence the system should present both corrective and limiting factors within the UI, and/or within the experiment. These methods of protection has been referred to as a virtual fence [38].

User management: This is a fundamental block of any information system, where critical information regarding the users is stored in central databases. User details include courses, user groups and experiments they are required or eligible to operate.

Some of the largest and most widely used RAL systems are studied and analysed for different existing features because these have been developed and used for several years for example iLab [9, 13] from MIT's Media Lab which was one of the first RALs deployed, SAHARA [9] developed by a consortium of Universities through Labshare in Australia and Weblab-Duesto [11].

2.1.1 System Architecture

The iLab has a three layered architecture called the iLab Shared Architecture (ISA). Users connect with a service broker server, which in turn makes a connection with the actual laboratory server. The system architecture is heavily dependent on web services [9]. iLab has also been used to implement extensions such as iLab-MIT-Africa [38] in African nations and some universities in Australia [14]. ISA is currently the architecture employed by most laboratories globally. Experiments in iLab have been categorized into three different delivery methods: batched, interactive and sensor [9]. Other RAL systems offer more straight forward connections that follow a client-server architecture, where all experiments were hosted at the centralised laboratories, and

accessed upon request by remote users. In this design, the lists of experiments are stored by the central server, which is also responsible for other operational aspects including running the RAL, scheduling, and operating the rig.

Recent developments in RLMS have moved towards grid architecture, but mostly within partner institutions. A recent trend is the federation of remote laboratories where the several institutions collaborate to share experiments. These institutions possess the experiments including the hardware and the supplementary learning materials which may not follow a standard in programming language or hardware. Federated remote labs use a protocol among themselves to inter-connect the RLMS and enable access to the experiments among each other. The federation approach of inter-connecting labs [39]:

- enables transitive properties by allowing resource sharing in transition
- supports distributed load balancing by redirecting students to different remote sites as per the network traffic at a given point of time

The federation allows large scale sharing of the instruments, but the experiments are still part of the institutions' domain. Unlike *makers* in the P2P RAL, these providers are efficient *producers* while hosting experiments.

It has long been realized that due to resources being scattered throughout a geographic area, a multi-tier distributed architecture has to be used to connect resources to allow remote laboratory services [40-44]. Initial attempts were to create an efficient brokerage between several physical labs across a wide geographic region. These systems give the users a variety of experiments across multiple laboratories and universities manage their local resources optimally [40].

Sharing laboratories has been suggested in [43] and a smooth interface between the physical laboratories is said to be crucial to determine the extent of sharing. Labshare and LiLa are collaborative projects of consortiums of laboratories that work in this direction. A flexible architecture that connects and deploys hardware from different physical laboratories into an experiment has been proposed in [42]. The main obstacle identified was the service-oriented architecture (for example SOAP) which is difficult to manage across heterogeneous networks and socket based communication is suggested as an alternative. The problem of inter-hardware communication has been

eased with the advent of HTML5 and the new capabilities of JavaScript and WebSockets [45]. These technologies can work in bidirectional full duplex mode and in real-time.

In most cases distributed technology and resulting benefits are aimed at the service model i.e. the universities, RAL developers and administrations. The overall architecture of the system remains the same client/server where the user can only view and perform a set of instructions and then acquire results.

Laboratory as a Service (LaaS) has been proposed that views laboratories as independent component modules [46]. Recently, there have been attempts to standardize the RAL command and data exchange based on this concept. The aim has been to encapsulate the exchange of commands/data into a particular set of web services or web based methods that can be incorporated for multiple experiment sites. This aim of the approach is to enhance the federation architecture for RALs by allowing a cloud based service provide LaaS [47-48]. These standards will make it very easy for institutions to share their equipment.

However, in context of P2P RAL, the rigs are to be built by individuals and they must be provided support in this regard. The standards of LaaS web services do not allow ad hoc rigs to be controlled with a generic interface. The actual commands/data exchanged for a specific experiment are often encapsulated in a higher level structure such as XML preventing them to be seen by external sources or the governing RLMS. This does not suit P2P RAL where the aim is to process the commands for experiments and provide supporting tools accordingly.

Also, these are based on Web Services which are slow [49] and more importantly, these rigs are not flexible enough and no universal approach is provided for students to build them. These are an organized approach for sharing existing remote laboratories among institutions. From a user's point of view, the system architecture remains in the service oriented model.

Web Instrumentation is the practice of controlling the actions of an instrument through a network environment. This methodology is popular in RAL systems [50]. Web instruments use a set of web services associated with the components of the instrument to operate them by calling the respective web service. This method is slow

as it initiates HTTP like connections procedure every time a web service is called and also too complex, involving acute understanding of object-oriented programming, creation of objects and attaching and mapping of methods. This makes it unsuitable to be implemented by individuals, particularly students and school teachers.

The notion of devising a common hardware platform that is able to integrate multiple experimental rigs potentially increasing collaboration between institutions and lower design costs have been explored in [51]. This approach uses FPGAs based on the IEEE1451.0 standard to attain a modular architecture for RAL. With respect to the current context, drawbacks of this approach include the complexity and the use of a separate micro-computer to intermediate between the user and the FPGA. The proposed approach implements the control unit of the experimental rig as a 'ready-to-go' component that can be directly plugged to the Internet. Personalized environments can improve the learning experience of the users [52-53]. In [52] the monolithic user interfaces such as the Java Applets are replaced by a set of even smaller applications - the Web Widgets. This method allows the users to rearrange the UI as they wish. However, this approach still does not allow the users to handle the actual rigs or configure the instruments which are required in the context of this research project.

More recently, desktop sharing technologies have been used to share laboratory experiments between users of different laboratories. A Relay Gateway Server (RGS) architecture has been proposed in [54], where it is used for connectivity between students, instructors, and experiments. The architecture consists of a publicly accessible RGS which acts as an intermediary and pass information between the users and the laboratory setups. In this system, the users conveniently access remote labs in web-browsers using Java and Flash platforms.

The IEEE Networked Smart Learning Objects for Online Laboratories Working Group (NSLOL WG) aims to develop an IEEE P1876™ standard for smart objects to be used in Online Laboratories. The purpose of IEEE P1876 is to enable providers to create remote laboratory experiments that have similar structural and operational properties/capabilities. This in turn is expected for easy integration into larger federated RAL systems. The P2P RAL follows a similar concept but needs to provide more specific tools that can be used by 'individual' makers.

Whereas the systems discussed in this section allow experiment access via a common

portal and shared between institutions, these service oriented approaches are not flexible enough to allow for individual experiments sites without extensive infrastructure requirements.

2.1.2 Experiment Scheduling

There are two major methods of scheduling users: time slotting and queueing. Combinations of these methods have also been proposed [16, 55]. Time scheduling is directly related to the nature of the experiment in terms of how much time it takes to complete one experimental activity event before the users have to provide further input. Some experiments are dynamic and 'live', requiring constant vigilance on the part of the user, whilst other activities may take a considerable amount of time to operate after the user has provided a set of inputs.

2.1.3 Interactivity of Experiments

Based on the level of interactivity experiments can be divided into three types [9]: interactive, batched and sensor experiments.

Interactive experiments take multiple inputs over a session and process them immediately. An interactive experiment provides rich user experience and allows the users to have greater control of the experimental rig. But due to the high rate of data exchanged, these experiments are dependent on the condition of the network for good user experience. Also ensuring the safety of the rigs becomes more difficult as it requires real time monitoring of each commands coming from the user.

Batched experiments ideally take only one set of inputs in a particular session and process them. The commands may not be executed immediately depending upon the length of any queue for users. Batched experiments may take a considerable amount of time to complete and usually generate large amounts of data [56]. Batched experiments are safer to control as the commands need to be validated only once when it has been issued and there is no need to execute it immediately.

Sensor experiments do not take any input from the user and are only about collecting and analysing data.

Any proposed RAL architecture must address the issue of interactivity of the experiments with respect to the nature of the experiments provided.

2.1.4 Deploying New Experiments

In all major RLMS new experiments are chosen by the administrators based on the university curriculum and educational needs according to the subjects being taught. The instruments used are typically of high cost featuring complex functions. Due to the nature of the experiments, these systems have to be developed within the laboratories of participating universities. The experiment configurations are generally composed of several experimental apparatus operated by a high level language, and typically involve a PC computer based controller. The user interface for the remote laboratory is also typically created by the laboratory staff. The scheduling aspect is easy to implement for instance as in the SAHARA software. These features allow developers to implement their own laboratory management systems.

Some RAL hardware for example VISIR used in various RLMS, provide a workbench environment and set of experiments, which is flexible but still limited to the number of experiments that can be performed with the given restricted component set [57].

2.1.5 Nature of Experiments

Remote access laboratories have been successfully used in teaching in fields from education [58], business, nursing [59, 60], and geographic information systems [61] to hydraulics and power engineering. This has been possible by extending the traditional definition of remote laboratories from merely controlling hardware remotely to a conceptual space of conducting experiments remotely [62].

Within iLab the experiments are varied in nature and maintained by different laboratories at MIT with different experiment focus. The micro-electronics laboratory for instance is the most prominent one. In addition to this, there are other laboratories for control theory, circuits' laboratory, micro-electronics and physics. All laboratories are built with a key focus on the required laboratory experience for undergraduate and graduate courses.

VISIR is restricted for use with analogue electronics basic experiments [63]. The UI for experiments may feature considerable flexibility and intelligence. The users can assemble and measure currents, voltage and other properties of serial and parallel circuits. The environment can detect and immediately inform users making incorrect

connections such as short circuits. This increases the students understanding about what can go wrong while designing a circuit. Although VISIR provides more definite sets of experiments, it is restricted to core electrical and electronics education for undergraduate students [63].

The experiments are all hosted at the university site and have been designed by academics. Some RAL systems use the remote desktop sharing as an experiment access paradigm where sessions are authenticated via a booking system that integrates with the institutional LMS. It allows users to view the experiments and the interface by directly transmitting the desktop image from the university servers to the user's desktop. A lot of different equipment can be run out-of-the-box using this approach making it very easy to implement any experiment quickly without much expertise. These systems use native programs of the rigs to operate them.

Traditional RALs often offer a static experiment environment with a fixed set of experiments with students having limited operational control. There are few examples of RAL experiments where the user plays a major role in deciding the design and operation of the RAL rigs. For example, in [64] using remote laboratories that shares equipment for research applications is described. It allows user defined programs for controller in an automatic control laboratory. This still does not allow the students to create the experiment setup. However, even if the students are able to reconfigure some parts of one particular experiment, the list of experiments available remains static for a given RAL system.

Go-Lab follows the federation approach of combining several online labs composed of simulation, real equipment and data sets for large-scale use primarily in STEM education [65-66]. The Go-Lab enables Enquiry-Based Learning (EBL) promoting interest and learning of deep conceptual domain knowledge and inquiry skills which are required in STEM education.

The Go-Lab is primarily focused on providing a rich educational experience in the online learning system termed as inquiry learning space based on EBL methodologies. It provides the pedagogical foundations of EBL in terms of RALs and corresponding online tools that engages both teachers' and students in creating digital material and learning process of the concepts of STEM. However, Go-Lab largely ignores the problem of providing any form of *hands-on-experience* which is vital for STEM

education. Hands-on-experience is the term used to refer to the skills acquired though physically setting up any experiment before using them. The greater impact of 'hands-on-experience' has been established in multiple cases [67]. Both in primary and secondary schools, such options increase interest among the students in participating in the activities. Many users and teachers regard it as a fundamental part of the learning experience in STEM education [67].

Another difference between the proposed P2P RAL here and Go-Lab is that the authentication of users into the system is the responsibility of the respective institutions while in P2P RAL, the authentication is done with a single database in the centralized global management.

2.1.6 Features and Trends of RLMS

Most laboratories have their origins in addressing problem of inaccessibility of equipment (i.e. more students and limited instruments) including iLab, Netlab (UniSA) and WebLab-Duesto. Some laboratories were developed to offer more expensive and hence higher performance instruments than the ones being used in the regular laboratories [9]. Later Labshare and LiLa were initiated to share resources among different institutions in Australia and Europe [68]. Some RAL systems were initiated to provide knowledge of the difference between simulated data and real experimental data on a computer.

Further to the original aims of RALs, of providing access to the instruments over the Internet, i.e. that users be able to access the instruments from their computers, several innovative steps were introduced that can be used to enhance the student learning.

Co-operation between students in experiments: Operating experiments via the Internet also allows for co-operation and collaboration between different students interacting, watching or lurking within the same experiment simultaneously. All of the 3D environments stated above already allow multiple users to access the experiment at a given time. In these instances, the users are represented by their avatars. Should it be desired, a multiuser interactive collaborative environment is required to allow concurrent users to have control over the entire experiment simultaneously.

Collaboration skills can be acquired by conducting projects with an embedded remote experiment and working as a part of a team. The RAL system NetLab gives students

the ability to form groups and negotiate time periods during which they can perform an experiment as a group [69-71]. Collaboration refers to the practice of creating small learning group of students where the group members actively support the learning processes of each other. Each group member can have a different perspective of the experiment and have different ideas for changing parameter to obtain the accurate results [71]. While any one of them set the parameter and runs the experiment, others can observe the result. Students feel the best utility of RALs is that it gives the opportunity to perform experiments repeatedly [70].

This collaboration is however only while running experiments. The students do not have the opportunity to the build the experiments together. The groups are focused on a particular set of experiment and usually come from a specific cohort e.g. classmates from a course. The P2P RAL aims to establish collaboration at a much higher level. The participants are not only able to perform the experiment in collaboration but they can also make it in collaborations. Also, one group of students can make something and publish it on the internet which can be used by another group. This also includes sharing the corresponding design of the experiments the program codes and any kind of experiences.

Dynamic Components Assembly: VISIR system employs a relay based dynamic circuit assembly system to allow students to build and test circuits during sessions by using micro controllers through a computer server. The Netlab system also follows a similar approach to connect several instruments together dynamically to form the experiment. Other systems have implemented this technology [72].

Reconfigurable Laboratory Kit: One general drawback of RAL systems are that they provide only a static set of experiments and the users never actually set them up. There have been some efforts to create low-cost reconfigurable laboratory devices that may be used by individuals to create and test experiments. An adaptable model of remote laboratory platform that can be easily re-assembled/configured for electronics laboratories allows large number of reconfigurations has been reported [73]. The WebLab has also created one such device.

These features of dynamic assembly and reconfigurable components to create RAL experiments are vital for the makers in a P2P RAL.

Lab on Mobile Platforms: Several RAL systems have tried and tested experiments from mobile devices [74] like smartphones. Mobile Devices pose a problem of being too compact and short on resources like Internet speed and computational power. So it is difficult to recreate the same effects as that of a PC. Several technologies like SMS, HTML5, Java and Adobe Flash have been used to implement different prototypes of experiments, but this method of distribution is still not very popular and majority of experiments are done through the PCs. This aspect of accessibility of the experiments and the RAL interface in multiple platforms is important for any RAL system.

Virtual 3D Environment: Several RAL systems have used 3D interactive and immersive environments to simulate the real world experience in the virtual world. The RemoteElectlab (Porto) has presented a case study for accessing a digital multi-meter through a 3D immersive environment [75]. iLab have created the TEALsim system to provide interactive physics experiments on magnetism [76]. REXLab has implemented a Young's Modulus experiment in a 3D virtual laboratory environment [77]. WebLab also introduced the most significant of these 3D systems, SecondLab, which is based on the SecondLife virtual world environment [78].

2.1.7 Pedagogy

RALs have been traditionally seen as replicas of on-site laboratories and every effort has been made to make these activities look exactly like traditional laboratory experiments. Some RALs accurately replicate the actual instrument panels on the web pages [69] while others use simplified interfaces and in some cases an enhanced version of the experiment. For example in a 3D experiment interface that shows the experiment action with additional simulated elements (the magnetic fields) otherwise not possible in real laboratories [76] as a form of augmented laboratory reality. However, as mentioned in [2], "It's probably a safe bet that few, if any, engineering programs implement remote labs for pedagogical reasons…" RALs usually do not carry any additional pedagogical values. iLab and Labshare developers have studied the factors affecting the convertibility of laboratories and experiments to RALs [9, 79-80]. Students learning outcomes [81-82] with RALs have also been studied and found to be adequate. Although there have been recent projects such as Go-Lab that have deviated from providing the instruments only to a much more comprehensive pedagogically driven RAL.

2.1.8 Common Advantages of Centralised RAL systems

Traditional RAL systems have been successful for many years and the systems have some key advantages. The experiments are designed with keeping a particular course and curriculum in mind. In other words, the lists of experiments are equivalent to that of an on-site laboratory. Since they are hosted by universities, there are qualified personnel to maintain update, modify or add new experiments.

All of the leading RLMSs have been used for teaching at in several courses. Each one has been used by more than a thousand students over several years. This suggests that these laboratories have been successful in providing an alternative platform [2-3, 79]. Centralised RLMSs have good technical support and are available as and when needed.

2.1.9 Characteristics of RLMS and their Suitability for STEM

While developers have improved and worked on different aspects of the RALs such as user interface and experiment pedagogy, the core architecture has remained the same.

The current trends for developing RALs allow only experienced and expert developers to create an experiment. As a result, the experiment variety is limited and concentrated in particular fields of higher education.

The instruments and devices used are mostly costly and complex to build and operate [83]. They directly use industrial standards such as GPIB, LXI [84] and PXI to connect the hardware to the computer servers. High performance software for engineering such as LabVIEW, VEE and MATLAB are also widely used to implement these experiment setups. Thus rig operation remains a matter of high complexity in all RLMSs.

Laboratory management systems are predominantly client-server in nature. All users need to log into a web address and provide user credentials to authorize access, select an experiment before utilising it. Any grid technology implemented is essentially limited to the server side of the architecture. The experiment configuration is also centralised and maintained under high-end laboratory conditions. All laboratories are designed to be operated for long periods and available to students all the time.

There is very limited scope for collaboration among students in different geographic locations, and not typically available in RLMSs except for forums [10], although this issue has been given importance in some systems [69,85-87]. There is also a trend to incorporate 3D user interfaces for collaborative learning purposes [68-70]. There have been multiple reports of 3D UI in various laboratories using different platforms, but it is not clear how many students have used these systems, although the positive effects on learning outcomes have been reported [77].

The experiments are mostly concentrated on providing for engineering courses in undergraduate and graduate degrees. There appears to be little attention directed towards school level science education, which is rapidly becoming an important area for development using enquiry based learning methods.

The enquiry based learning methodology [88] in STEM requires students to analyse problems and find solutions through the application of practical knowledge and implementation to understand the concepts. As such there can be an infinite number of different setups of rigs and devices that may be used for designing different concepts. Moreover with the school systems, it is the teachers and students who are closer to designing an experiment setup than experts who are already providing pre-setup rigs. But, with the above stated features for creating new laboratories, experiment setup is difficult for them.

There have been recent concerns on the slow adaption of remote laboratories with teachers [89] for their students. Faculty resistance to incorporate new technology in teaching and technical support issues have been cited as main reasons behind underutilization of remote laboratory technologies. These reasons become more prominent if the rigs that are supposed to be used by teachers are actually designed by people other than themselves. Another study in Europe concludes that schools and teachers are very interested in remote laboratories, but are unsure how to integrate them into school curriculum [90]. This is mostly because they are incapable of fulfilling computational requirements in RAL implementations and applying the relevant pedagogical and technical concepts.

Since RALs are considered as extended on-site laboratories, their curriculum and structure closely resembles the onsite laboratory. This is perfect for higher education where experiments have a fixed nature and done with specific equipment. On the

other hand, in STEM education, while the list of objectives may be static, the physical system on tends to be very flexible. The same kind of activity may be done with various setups to understand the STEM concepts behind it. These setups need to be built and used by students for effective learning.

2.1.10 The Peer-to-Peer Architecture

The proposed *pedagogic* solution for employing RAL experiments in STEM areas is a distributed or P2P RAL system where participants may be both creators of experiments (*makers*) or share them with others and be user of others' experiments (*learners* or simply *users*) creating more flexibility on the laboratory provider side. School level children are capable of participating in this kind of activity as evident from recent initiatives taken to incorporate RAL activities into schools such as the robot-RALly project [27, 91]. A project with RAL at University of Southern Queensland was used to create enquiry-based learning activities and facilitated collaborative learning between elementary school children from Japan and Australia [91]. The study indicated that such technology can thrive in school environment also but will need transition from the client server to a peer to peer architecture where students can directly interact with others and their experiments.

This change in architecture provides a potential solution to the incorporating STEM and RAL, but requires a number of technical challenges to be resolved. The fundamental challenge is the shift from a predominantly client-server RAL architecture which is successful in terms of technical and operational capabilities to an open ended architecture that would allow multiple users to participate in creating the experiments. This may be done by drawing parallels between RAL systems and Internet of Things applications.

2.2 Internet of Things

The Internet of Things [92-93] aims to create a network of regular objects used by people in common everyday tasks with capabilities such as identification, sensing and data processing. These objects (or devices) operate collectively over the Internet to accomplish given objectives. By its very nature IoT applications rely on distributed processing at least partially.

The IoT is composed of and dependent on a vast and heterogeneous set of objects,

each one of which provides certain specific information and functions. Each instrument can be accessed through a particular set of instructions corresponding to its platform.

For the over-arching application and its output interfaces to procure and display the correct data there must be an abstraction layer capable of harmonizing the control of each device in the system for example a common language [94] or a device must offer discoverable services on a network.

The IoT is described as a convergence of three related areas [95]: The *Internet* and how the devices such as personal computers, servers and mobile devices co-operate with each other to exchange data. *Things* or small-embedded devices that are usually capable of low level computing dedicated to a particular set of operations. *Semantics* or the method to establish meaningful conclusion from a vast amount of gathered data by parsing or analysing using computational techniques.

The advent of low cost micro-controller devices such as Arduino which are available as consumer electronics devices has opened the door to a large number of possible ways to create and configure the devices in IoT systems. These devices are not as powerful as personal computers or even mobile devices, but their ability to operate with multiple sensor and actuators makes them ideal for creating 'end-nodes' in an IoT system. An end-node in the IoT system collects the data and sends to relevant destinations for further processing. These devices are capable of connecting to Internet and using full TCP/IP stack [93].

In this context the P2P RAL can be described as an IoT application with respect to the communication and the end-node paradigms. However, where the P2P RAL adds or improves upon the IoT is in the semantics by involving human user to a large extent in parsing of data for learning purposes. It even requires the users to create and dynamically add to the semantic processes of the IoT. The P2P RAL system is also a changing or volatile system i.e. the end-nodes may not be continuously available for service.

Normally, the interaction in RALs is one-to-one communication between the student and the instrument. This is also true for P2P RAL, but there can be connection where a single user may connect with multiple instruments at different sites (one-to-many)

and multiple users at different sites may share the same instruments (many-to-one). However, the major focus of collaboration among STEM students is to work in a group and to collect and analyse data. This is during the experiment setup phase and subsequently in running the experiment locally and remotely. This can be achieved with the technical implementations of both one-to-one and many-to-one. But from an IoT perspective the connections will involves relatively fewer number of end-points.

2.2.1 Common Components of IoT Applications

Hardware: The *hardware* in a IoT is heterogeneous and run on various native platforms and software. Many devices are based on RFID for tagging and location estimation **[**93] of objects, or sensors and actuators to collect data and alter certain physical system setups. The P2P RAL also has to enable the use of multiple devices for the experiments controller. All of these devices must be able to communicate with their corresponding learner nodes. These devices can be programmed to be smart i.e. identify patterns in the incoming data and make decisions.

Middleware: Any IoT system is expected to operate on a vast numbers of devices with heterogeneous interfaces. These generate enormous quantities of complex data. A middleware is used to create a homogenous set of processed data streams from these raw data from the IoT hardware which feeds to the overlying applications for users. To enable exchange of ideas and experiment related design and experience data, a middle ware is also required in the P2P RAL. This middle ware would essentially convert the data from all the underlying devices into a common format such that the makers or the learners can access the system with a uniform user interface for both making and running experiments.

Search and Discovery: In a typical IoT system with a very large number of objects, it is necessary to search for objects. Searching involves not only stable contents such as identity of the objects but dynamic properties of the objects. It has been suggested that special web browsers may aid in this operations. In case of P2P RAL, the number of objects may not be very large, but the objects have variable properties and functions depending upon what the owner of the object wants.

The Internet allows for the communication between devices. Some of the technologies include RFID, Wireless Sensor Networks and ZigBee. The network between the

devices may be highly heterogeneous consisting multiple protocols and medium.

A cloud computing model based IoT system to share social devices has been proposed in [96] which is part of the clouT project [97]. It provides a virtual execution environment in a decentralized manner with high reliability without any space or time constraints. This approach allows for easily reusing distributed IoT resources with an enhanced homogenous service layer on top of their individual heterogeneous services. Any consumer applications can be created by integrating those services and deploying a package into a global service platform distributed in form of a cloud. This enables secure exchange of data among the device connected to the cloud platform.

This work uses a three-layered architecture with a gateway as the middle layer. The middle layer translates the heterogeneous services from the various IoT devices into homogenous consumable web services as in REST or JSON format of data. These data can then be consumed by the devices. Thus devices that would otherwise not be able to communicate with each other can share data through the cloud based gateway.

A Semantic Gateway as Service (SGS) has been proposed to allow translation between messaging protocols such as XMPP, CoAP and MQTT with multi-protocol proxy architecture [98]. This also proposes to create a middleware to convert data to be processed in a cloud based environment.

While the concept of a cloud based gateway for translation between heterogeneous services is applicable for the P2P RAL as well, it is not suitable for translating the service online in case of P2P RAL. The P2P RAL uses MCUs as the core of the experiment rigs. Each MCUs as a part of an experiment can be programmed differently. Thus the homogenous layers are individually software modules based on a common algorithm that are placed on the experiment rigs instead of in a cloud based environment. This allows for quicker processing of the commands as needed for validation, evaluation and guidance.

2.2.2 IoT and Human

Recently, there has been effort to study the Human Computer Interface requirements for IoT. Most research in IoT [99] is generally less concerned with what the hardware components used are, but, more concerned on exactly how computing could be incorporated into the objects. This approach is applied for RALs where the makers are

given the freedom to make anything using a large variety of objects. Although the actual number of such objects is restricted for any practical purposes. Other studies have focused on the ways humans can incorporate new objects in the IoT system with least effort and error [99]. The connectivity between objects is also reported to impact the way the IoT system will be designed and used [100].

Since IoT applications are created to operate discretely and do not require a core centralised server, cloud computing principles can be easily applied for the IoT Architectures. Also, the devices used in the IoT are usually available for a long duration of time [93]. The application logic is not stored to operate on the external interfaces visible to the outside environment, but stored in multiple nodes in the system that communicate with each other and generate the data and operate the other relevant nodes.

More recently integrating social networking concepts into IoT solutions has been investigated [101]. It can support novel applications and networking services for the IoT in more effective and efficient ways [101].This approach takes a non-traditional view of Internet of Things (IoT) based on the concepts of opportunistic IoT. Instead of connections between the physical devices in a global infrastructure only, it allows for ad hoc, opportunistic networking of devices. The concepts of opportunistic IoT closely tie the human element with the operations of the IoT devices. However, the main focus of this work is data sharing that has a major impact on the underlying service of the IoT systems concerned. The opportunistic IoT aims to send and share data among suitable nodes in the network such that the information reaches the correct nodes resulting in consumption of some resource that is represented by the data. The end nodes for example smart phones or smart vehicles etc. are closely related to their human owners who impact the sharing process.

In terms of P2P RAL, on a larger sociological context, human makers can impact the learners with their presence in the system and change their practices in learning STEM subjects. Also human users run the experiment and they must run it according to some constraints set up in the experiment thus impacting their behaviour.

The most important concept that is also applicable in case of the P2P RAL is the opportunistic or ad hoc communication between any pair of nodes. However, the major aim of this thesis is to allow a direct one-to-one exchange of commands

between two node with an experimental rig at one and it's controller on the other end and not data sharing based on personal devices such as mobile phones.

2.2.3 P2P RAL and IoT

RAL experiments based on IoT concepts have been developed before [102]. This research focused on using Arduinos as controllers for an experiment that can facilitate collaboration between different schools so that each can have access to laboratory resources in the other. This work however, did not provide a peer-to-peer service model for the RAL system or any generic model for the experiments that can have a common programming platform. This work had a centralized approach where the Arduinos were set up by experts for a fixed demonstration experiment and hosted in the schools.

The Web of Things [103-104] is a newer concept that builds on the application layer of the Internet of things. The Web of Things aims to use existing technologies into the smart devices to create the web using web technologies i.e. Web Services or WebSockets that are already available, instead of creating new low level protocols or hardware for customized communication in IoT Systems. The advantage of WoT is that it is easier to integrate into the existing Internet infrastructure with the need of separate network capabilities.

In the Web of Things, smart devices could run web servers and provide and consume services as any other fully capable computational device [103]. The functionalities of the web servers will be limited to what is needed for the system. The P2P AL follows this paradigm of IoT in particular. In P2P RAL, the MCUs provide the functionalities of being the link between the users and the sensors and actuators. They can host web servers and other tools to process incoming data which must follow the requirements of the P2P RAL system.

In WoT resources or end-nodes can include physical objects such as temperature sensors or abstract concepts such as collections of objects which must satisfy a number of constraints [105]:

1. Resource identification by using unique strings such as URI
2. Uniform Interface with well-defined interaction semantics
3. Self-Describing Messages such as the XML or JSON that contains the

metadata along with the data

4. Hypermedia Driving Application State that allows the exploration of services once the resource has been identified
5. Stateless Interaction requests as in HTTP.

Constraint 4 is not required or considered in the P2P RAL architecture. The MCUs or experiment controller needs to have a static logical flow of operation for a given experiment and all of it must be exposed through the P2P RLMS. The client in the P2P RAL does not need to explore and find out about the services themselves.

The new P2P architecture requires several automated features in the RLMS in order to aid the makers to create an experiment. This requires a generic model of the end points in the system. This model of the two end-points one controller interface (master) on the user side and one controller unit (slave) on the experiment side requires a communication language that can be used to govern a wide range of experiments. This communication language forms the new layer in semantics in terms of IoT.

Thus the P2P RAL can be described as an IoT system due to the following characteristics:

- Large number of devices interconnected to share data;
- Each devices being capable of collecting and processing data to at least some extent;
- It is based on TCP/IP and devices are uniquely addressable; and
- It uses potentially intelligent devices capable of making decisions individually and in groups.

However, there are also some unique aspects in the P2P RAL system as well. The P2P RAL is designed to support human use i.e. directed towards human learning systems. P2P RAL IoT incorporates two types of *end-nodes* - the experiment and the *participants* (both makers and learners). The experiments are similar to any normal node in IoT i.e. contains smart devices, but the user nodes are different. The user nodes consist of a computing device such as PC or Mobile phones that runs the experiment. The experiment is run from with an online environment accessible through browser on the user node, which parses the human inputs to commands

suitable for the experiment. Hence the user nodes do not have sensors or actuators on them, but still have to be smart enough to interpret the human user interaction.

The communications between the human and the nodes are segregated. Multiple human users can connect to a single node and vice versa, but the ratio between human user to devices i.e. the number of devices accessible to human users at a time is very low compared to general approach IoT system.

2.3 Summary

RAL technologies have been largely confined to replicating the experience of on-site laboratories. The focus has often been on the accuracy within a remote online environment to maintain equivalent learning outcomes. These laboratories largely focus on the fields of higher education, but lack the capability of infrastructure support for STEM education and related physical activities. The resulting online learning tools mainly aim to resolve the resource constraints of universities. STEM education has other needs. Collaboration and hands-on experience of creating and running experiments are key requirements. The current features of RAL systems are complex and mark a barrier for individuals in schools with little experience in networking, computer systems and instrumentation. By using newer web technologies and the peer-to-peer access paradigm based on IoT principles of distributed network of devices, RALs could provide much richer environments and experience for students remotely interacting with experiments and collaborating in joint activities in the context of STEM education.

3

P2P Remote Access Laboratories – Research Questions and Methodologies

> *This chapter presents a general description of the proposed P2P architecture for RALs, the corresponding research questions and methodologies.*

The client-server architecture and different technologies that support RALs have been previously investigated in detail [106, 107]. Hardware and RLMS are generally hosted by universities. The RLMS is usually also responsible for authentication and scheduling of users access. These systems employ the notion of a service provider that provides experiments at the server side. This architecture allows for little operational autonomy in regards to the physical location and the design of rigs. This limits the pedagogies that can be employed in the remote laboratory space, as students are generally not involved with the design of experiments [28, 108]. Many student activities focus on outcomes, and experiments are used to collect results.

P2P RAL is a new concept introduced in this thesis that aims to enable students to create their own experiments. Once an individual for example a STEM subject student or teacher can create and share an experiment, other users can use that rig to learn. They can also possibly modify on that design or create a new experiment based on the available rigs. For experiments in STEM education, several students were involved in a P2P approach of RAL in previous work, for example, Robot RAL-ly [108, 34, 49] which has demonstrated the feasibility of an approach with users being able to setup experiments. Thus the focus of this work is to develop tools to enable the students to create these experiments using a common platform and share them through the Internet.

This chapter first describes general experiment components in details in Section 3.1. This section identifies unique components that are common to majority of the experiments in RALs and relevant to the new P2P architecture. The notion of a distributed remote laboratory system is discussed in Sections 3.2 along with two unique strategies for control with respect to commands that are passed during the experiment. This section also defines three major requirements of the RAL system that needs to be applied in the new architectures as well. These three requirements forms the basis of research in the next chapters. Section 3.3 describes the extended Peer-to-Peer architecture based on the distributed RALs and their general properties. Section 3.4 describes the two distinct activities of the P2P RAL - *making* an experiment and *running* an experiment. The technical requirements and how they can be addressed are described in Section 3.5 followed by the resultant research questions in Section 3.6. The original contributions of the chapters are outlined in Sections 3.7 and the methodologies followed for the research in discussed in Section 3.8.

3.1 General Experiment Components

An experiment requires multiple components. In the context of this work, the main components of an experiment are:

- Measurement Unit (MU);
- Controller Unit (CU);
- Remote Laboratory Management System (RLMS); and
- Controller Interface (CI).

Their relationship is shown in Figure 3.1. The user is depicted on the left hand side accessing an experiment through the Internet. The MU encompasses the actual experiment measurement and control instrument. It consists of a combination of sensors and actuators that cause actions and collect experimental data. The MU receives requests and responds with data or error information. The CU is the component that connects the MU to the user of the activity through the RLMS. RAL environments rely on the TCP/IP based Internet to establish connections between the users and experiments. The CU is a networked computing device that hosts the corresponding drivers to control the MU.

Experiment users control the system using the CI that contains Control Program Logic (CPL) and user interface both created by the maker of the experiment. Commands to

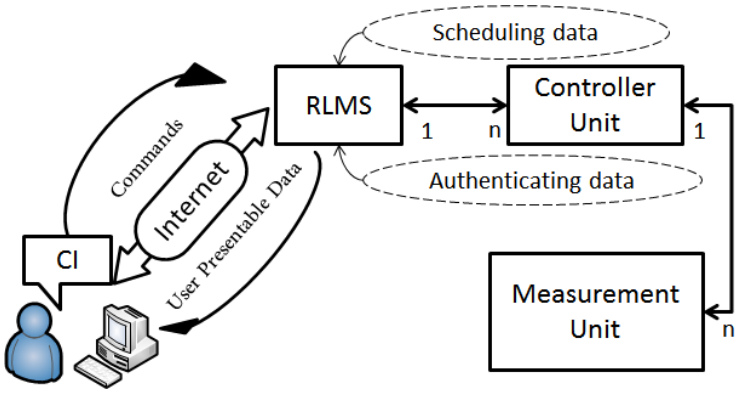


Figure 3.1. The RAL experiment components.

the CU are issued in response to user interactions with the UI according to the predetermined CPL. Any corresponding outputs from this interaction are generated by the CU and returned to the CI. The RLMS stores the CI which is downloaded to the client at the start of the session. It mediates between the CU and CI during run time [28]. It also handles authentication, access control and scheduling of users. The CI is provided by the creator of the experiment but run at the client site.

Undertaking an activity with an experiment involves the students interacting with the UI and giving inputs which are then processed by the CPL to create commands for the CU controlling the MU. Results are then returned to the UI. In relation to these interactions, three critical aspects of implementing an activity include:

- validating commands to ensure safety of the experimental rig;
- evaluating student performance to ensure proper learning by determining whether the users have performed certain acts and obtained the corresponding results from the experimental setup; and
- enhancing user experience to ensure support is provided when needed.

3.2 The Notion of Distributed RALs

The distributed RALs aims to decentralise the location of the experimental resources such as experiment hardware and learning materials. In a distributed RAL, multiple experiments are available throughout geographically separated locations. Each experiment is available for integration into multiple learning activities as required in a given learning context. There can be two broad ways to implement distributed RALs that we define as *black box* and *white box approach*.

For the *black box approach* the RAL systems and the control mechanism are

considered to be black boxes i.e. their internal mechanisms are not transparent. This method is concerned with the end-to-end control functionalities. It does not focus on the actual experiment control mechanisms. The RAL system is not concerned with commands or their structure. Communication data is encapsulated and relayed between the controller and the experiment. Experiment design and user inputs are specified by the creators of the experiment.

The advantages of this approach are that it is simple to implement. Existing resources can be easily geared to become available for integration into the RAL system. However, it is assumed that the creators of experiment are able to implement the common requirements including validation, evaluation and support. This limits the number of experiment creators [109-110].

Go-Lab [111] and OnlineLabs4All [112] are two examples of Remote Laboratory projects which may be regarded as largely following this black box approach. Go-Labs provide an online environment to create a learning space customized to the teacher using the system to create the experiment activity [109]. The learning contents are presented in a customized manner and a set of tools to aid learning are provided which the teacher can use to evaluate the learning outcomes. However, this process is not automated and depends largely on the teacher creating the experiment activity. In OnlineLabs4All a new approach is adopted in which queuing, lab data storage and deployment are offered as a service for experiment owners, allowing the lab specific part to be loosely coupled with the RLMS and lab server. An experiment must follow a set of specifications set out by the RLMS for ensuring accessibility. These specifications focus on checking availability, booking, passing on the messages for control and results. The specifications do not govern the control of a given experiment.

Using the *white box approach* (WBA) control mechanisms are at least partially known, i.e. the structure of commands are fully known or can be derived from a known set of rules. Learners in the system are encouraged to take responsibility for creating experiments. Rigs are created by students for students. This enables wide scale collaboration between participants. This approach enables the RAL system to implement the requirements of the experiments automatically by analysing the performance of the students with the experiments. This also allows for novice users to

become makers of experiments without having to learn in-depth programming and automation skills.

The white box approach allows more hands-on-experience for learners who don't possess the necessary skills. Once the participants have become used to creating rigs, they may progress to a black box approach, whereby they can implement control mechanisms beyond what the RAL system can deduce or provide support with.

The white box approach is the main focus of this dissertation, where the participants - makers and users are considered absolute novices with very low experience and with low quality resources at their disposal.

Figure 3.2 depicts a typical example of an experiment in WBA distributed RAL system. There are two sides for communication in every experiment session - user and the experiment. The RLMS establishes the connections between the two sides based on certain predefined functions that are implemented in the experiment CU.

This research focuses on finding a generic control model for the CU (Y = F(c)) with respect to a generic CI or user interface based on a fixed set of commands. The RLMS defines the specific commands for a CU hardware. Typically, the commands are same for every CU hardware, but implemented with different software depending upon the actual hardware. This provides a universal set of basic commands. Obviously, more complex commands may be created which are specific to a particular hardware or even experiments derived from these basic commands. The experiment makers are not required to have knowledge about the implementations of the basic commands on hardware or the communication establishment between the nodes, all that is taken

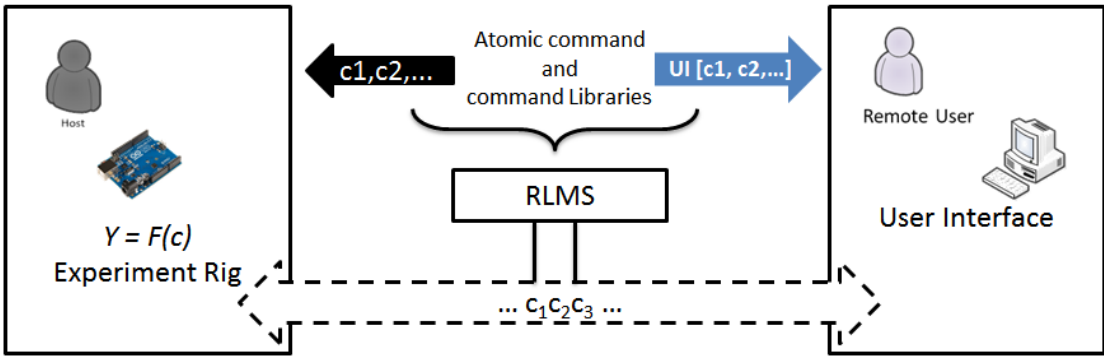


Figure 3.2. The WBA command based RAL experiment architecture.. The same command library is used to create the CPL in maker's process and used by CI to send command to the CU in learner's process

care of by the RLMS.

However, as the RLMS is now aware about each command for the experimental rig, it can automatically create a model of each individual experiment setup. The RLMS can monitor the exact command that are being exchanged and determine the quality of the session and provide the services of validation, evaluation and support automatically.

3.3 The Proposed Distributed Peer-to-Peer RAL

In this thesis, a new architecture namely the Peer-to-Peer architecture for distributed RALs is introduced. This P2P approach follows the WBA i.e. support the participants by monitoring commands.

The proposed distributed P2P RAL is a network controlled system driven by human participation where the equipment and their users are distributed geographically. The Internet is used as the medium of communication between users and the instruments. The nature of the system is peer-to-peer, i.e. connections are established point-to-point between users and experiments. Participants are responsible for creating and managing experiments on the Internet. The distributed RAL system aims to incorporate both experiments building and running experiments into the curriculum. The entire system is to be run by users or the 'maker' community which includes students as well. Once the maker has created and tested the equipment successfully, the experiments are published online for others to access. The instruments at the experiments side are operated from the Internet by the users.

Figure 3.3 shows the typical P2P RAL scenario with two end-nodes on the left and right that are supervised by a cloud based repository and authentication system (centre). This RLMS supporting the P2P RAL is responsible for creating the link

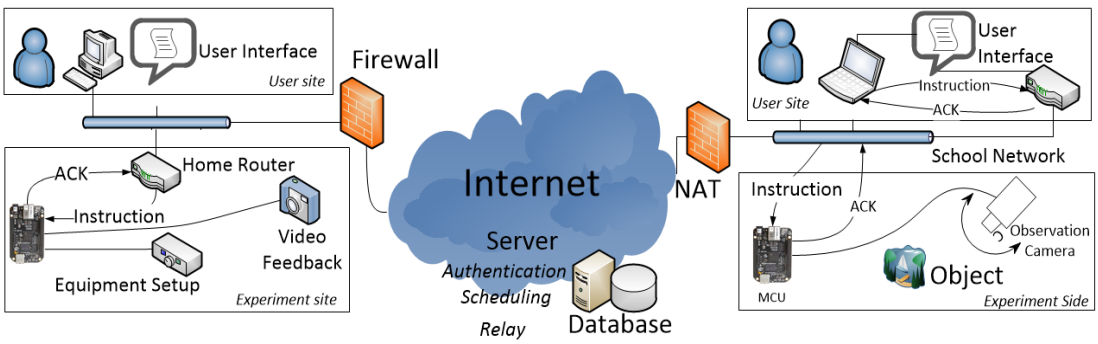


Figure 3.3. The distributed architecture of the proposed RAL system

between the users at the organisational, user and communication level. Such nodes can be behind firewall and NATs and have variable network capabilities. Each end-node is either an experiments rig consisting of all its parts i.e. the controller and rig connected to the Internet or the user and the corresponding learning device, for example, PC or mobile device connected to the Internet.

3.3.1 Differences between Centralised and P2P RAL

Typically RAL systems are catered using a centralised system [28], where experiment rigs along with their CI are created and maintained by limited number of service providers, such as universities. The CI, CPL, and UI are created specifically for a particular rig and integrated into the RLMS by these providers. This allows for each individual rig interface to be equipped with specific tools to monitor and validate the interactions. One of the major shortcomings observed of many centralised RAL systems [28] is the lack of direct hands-on-experience. The students are generally provided with ready-made experiments for end use only. This is due to the high cost and expertise required to construct traditional instrumentation experiments, and not having a published RLMS protocol or flexible middleware.

In contrast, in the proposed P2P RAL system an 'institution' is no longer required to conceive, build, or maintain experiment rigs. In this model, individual makers can conceive, design and build experiments including the corresponding CPL/UI that can be then used remotely by other users. However, unlike *institutions*, an individual *maker* cannot include measures for monitoring and validation in their CPL/UI.

> Individual makers, although hosting RAL experiments, are still *consumers* of the RAL system unlike institutions.

In P2P RAL, makers start with only a small set of development tools (for example, the communication protocol and the interface development tools) provided by the RLMS. Makers then complete the experiment design through creation of the CPL and UI for their rigs, thus allowing for varied open experimental sites. The RLMS, or its administrators, have no direct control over the activity or experiment added to the system.

As such, there are three general "roles" involved in P2P RAL:

- the *administrators* who are responsible for the RLMS features,
- the *makers* who design and build the experiment with the CU and MU (i.e. the rig) along with the CPL and UI, and
- the *user*s who interact with experiments for learning purposes

An individual maker is not expected to implement tools for evaluation, support, commands validations or any other features that would otherwise be specifically developed for each experiment if implemented by institutions. Thus the P2P RLMS must be able to provide all these tools without the individual makers having to implement it specifically. The P2P RLMS is just a *supervisory unit* that must monitor and validate the user interactions with the instruments with universal tools based on a generic model.

One distinct property of the P2P RAL is that the resources i.e. experiments are not required to be ever-lasting as in a traditional RAL. On the contrary, several participants can create experiments and then re-use the components to make another experiment after some time. However, the tools of P2P RAL can enable teachers and their students to create rigs that can be operating for a substantial amount of time, before another group of new participants subsequently pick it up and create new rigs for the same experiment, replacing the older ones in the P2P RAL system. Obviously, a well-built rig may be kept online for a very long period of time if so desired. Thus the P2P RAL provides great flexibility.

The differences between a centralized and a P2P RAL have been summarized in Table 3.1.

Centralized RALs	**P2P RAL**
All resources i.e. hardware and software are concentrated at a particular place and owned by a single entity.	Resources are distributed and owned by individuals unknown to the RLMS.
Available for 24x7	Availability may or may not be 24x7
No Hands-on experience for setting up the experiment	Full scope of hands-on experience for setting up experience

Collaboration is limited and only possible in running experiment in small groups	Collaboration is possible in running an experiments and in sharing the experiment creation experience
The resources are expected to be ever-lasting	The resources are not required to be ever-lasting
No Re-usability of experiment components	Wide scale re-usability of experiment components is possible
No special support is needed	Makers/Developers needs support while constructing the experiments
All resources are available at a given location, both in the network and geographically	Resources are scattered over a large geographic region and network addresses must be allocated dynamically

3.3.2 Properties of the proposed Distributed P2P RAL

The proposed distributed P2P RAL approach aims to expand the one-to-many approach, where a single or a collection of few central laboratories serves many users, to a many-to-many approach with many users using multiple equipment setups provided by different makers. In a distributed RAL, experiments are to be created and hosted by individuals [113]. Users are all scattered in the network and anyone can connect to anyone. In this model of RAL, the experiment module is no longer a part of the RLMS as in a client-server model. This results in two types of modules, the experiment modules containing the actual experimental setup including the hardware and the software related to it and the user modules which remains the same as a centralised RAL i.e. just using the interface of the experiment.

Designing an experiment will include assembling an equipment setup, programming and run experiments locally and sharing the experiment with others by putting it on the Internet. A distributed architecture has two characteristics: modularity and high scalability.

A modular design consists of individual modules or entities, such that each of them can operate independently as well as work together towards a larger goal. It allows users to combine separate experiments to create the workbench without the need of integrating it to a larger structure. In a modular design, new and improved

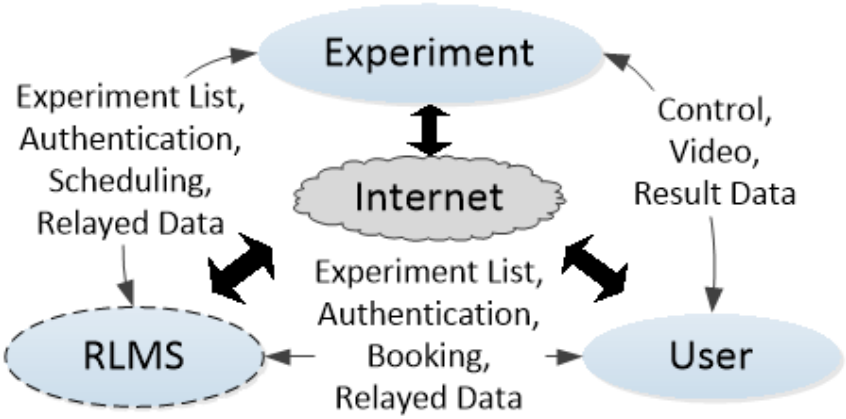


Figure 3.4. The modular nature of the distributed RAL

experiments setups that are built subsequently may replace older ones. It is not necessary or expected that any of the experiments will be hosted for a long period of time. This will enhance collaboration as several makers can work on individual experiments at the same time, and then combining them together for a bigger project.

The experiments repository may extend without bounds with users adding their creations to the system. New experiments could be directly added and made usable to others students. This gives the creators full liberty on design and operational paradigms. Any experiment can be added or removed from the system without having to change the rest of the system.

Figure 3.4 shows the structure of the modular design with three modules and the data that flows between them. The experiment modules and user modules are the two end points in the architecture mediated by the RLMS. The user node goes through time scheduling (for example, time slotted or queuing) with the RLMS. The experiment node then authorizes the access at the appropriate time allowing the user node to start issuing the instruction commands. These commands and the corresponding result data are exchanged through a Virtual Private Network (VPN) or an underlying overlay network as part of the RLMS on the Internet.

3.4 The process of creating and running experiments in the proposed P2P RAL

The P2P RAL contains two major roles – the makers and the learners. Figure 3.5 shows the making procedure in a P2P method. It needs to have the following steps:

i. Select: Selecting a STEM topic and looking through the RAL system if there is any experiment of that nature.

ii. Create: Selecting a suitable CU hardware platform for the experiment. This will depend upon the form factor, power capabilities and the number of types of sensors and actuators it can handle.

iii. Program: Once the experiment is setup, it needs to be programmed. This must be using a similar library for each experiment for enabling wide scale sharing. The libraries are discrete set of basic commands that are provided by the developers of the P2P RLMS protocols. The commands can then be used to create the CPL and UI for each experiment specifically.

iv. Train: The maker can then train the experiment to create control models specific to the experiments locally. These models can be used for the purpose of validation, evaluation and support. The control model needs to be based on a basic generic model that can be extended for any experiment. The training may include several intelligent tools which are based on Markovs decision process or clustering.

v. Publish: The maker then creates the experiment webpages with its descriptions and aim and other learning related materials and publishes the experiment on the internet. When published, the experiment hardware is uniquely identifiable in the P2P RAL's network system with a set of links.

The technical aspects of enabling the Create, Program and Train phase with appropriate tools and software are the main issues addressed in this thesis. This includes:

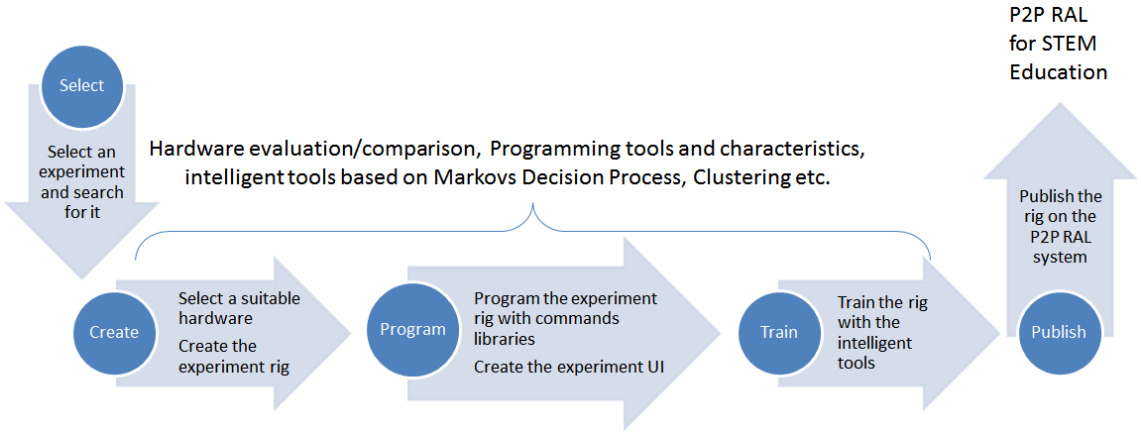


Figure 3.5. The experiment creating procedure

- Analysis of suitable hardware platforms for the experiments.
- Analysing programming tools and characteristics and defining basic commands
- Creating intelligent software tools based on Markovs Decision Process, Clustering etc. to automatically analyse the user's inputs.

Figure 3.6 shows the learners procedure for accessing the experiments. It is as follows:

i. The learner logs in to the system and receives a list of online experiments and selects that. The learner may book the experiments immediately or for a later period of time.

ii. During the experiment session, the P2P RLMS authenticates the learner's node with the remote CU of the selected experiment. The RLMS then supplies the CI common to all experiments and the CPL/UI and any other models or learning materials specific to the particular experiment to the learner node. The learner node then receives the links to the remote CU and starts to send the commands to it.

iii. During a session, the commands can be monitored by the RLMS or in particular the CI and associated tools on the learner node. The learner node can use the models to identify any wrong commands that are passed and evaluate the performance of the learner based on the inputs by comparing it with the models.

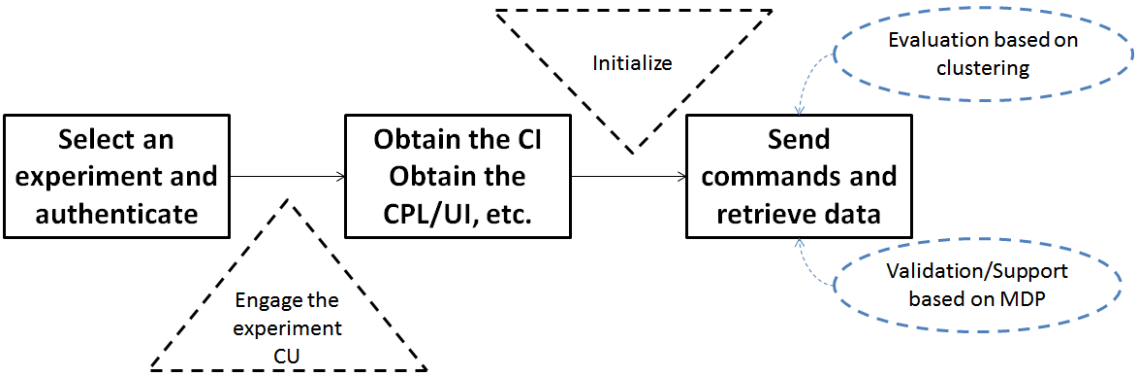


Figure 3.6. The experiment running procedure

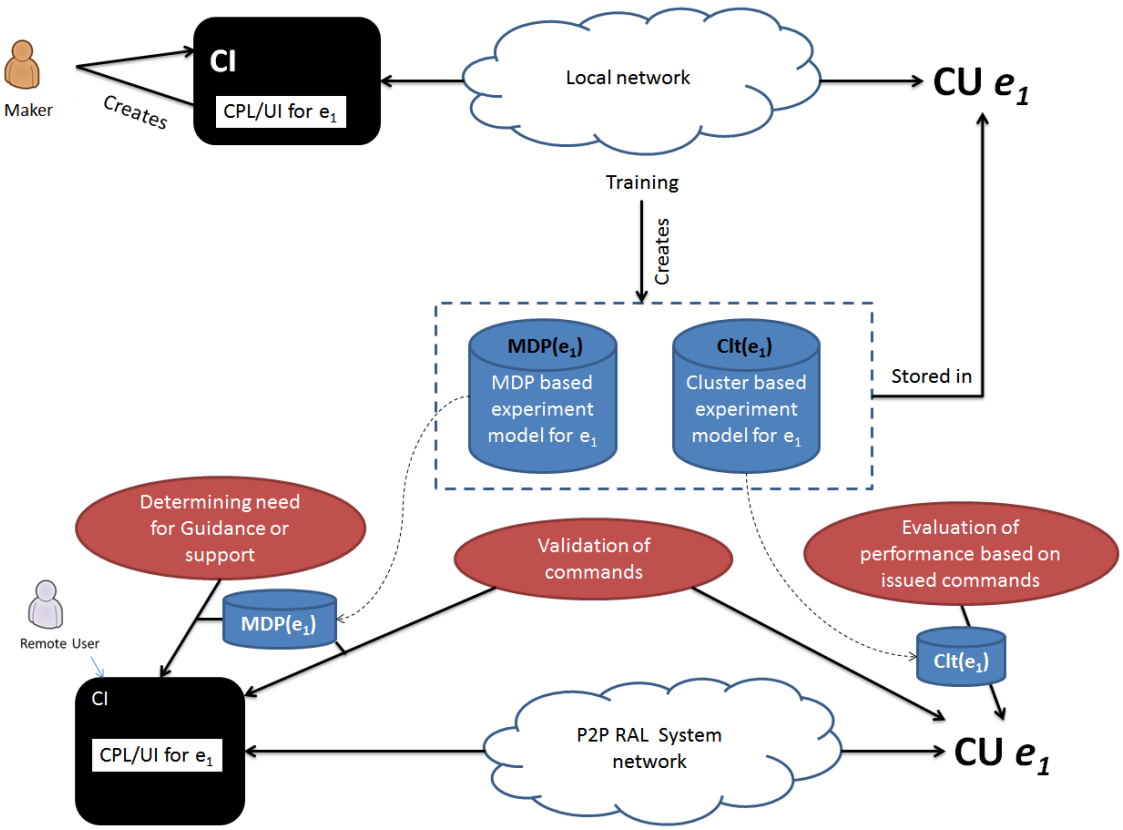


Figure 3.7. Maker and Learners in the P2P RAL

Figure 3.7 shows the process of creating experiments and using them with respect to the internet. It shows the two different scenarios when the makers make the experiment in the top half and the second scenario of users using the experiment in the lower half. When the makers initially make the experiment they can use the local network as LAN which will have negligible latency or they can use the P2P RAL network which will have greater latency. The makers create the CPL specific for experiment e_1. While makers use the experiment the specific models for the experiment based on Markovs Decision Process - *MDP(e_1)* and clustering - *Clt(e_1)* can be created automatically and stored in the CU. When the remote users run the experiment the global CI loads the CPL and the other models. The CI processes the users' inputs according to the CPL and sends the commands to the CI for the experiment e1. This is always through the P2P RAL network on the internet. The CI and CU can then collectively provide the services of validation, evaluation and guidance based on the MDP and the clustering models.

The proposed method to create and use the MDP and Clustering algorithms assumes a general network like the Internet as the medium. Thus the actual architecture of the

P2P network system does not disable the CI-CU command exchange. Obviously, the better the structure of the P2P RAL network, with lower latency between the CI and the CU, the higher the quality of learning experience.

3.5 Technical Requirements of the P2P RAL

The proposed P2P RAL system needs to meet the following technical challenges:

1. A generalized hardware base platform that is extendable to implement multiple experimental rigs.
2. A generalized programming platform that build on top of the hardware which can be used by students and teachers at schools.
3. A network of devices that can minimize the network latency to provide best learning experience.
4. Allowing sharing of experiments among students maintaining system integrity constraints such as reliability and availability.
5. Methods to measure the quality of learning experience and provide support to students on the generalized hardware and software platform.

The first three requirements are addressed by using the proposed generic model and a protocol presented in this thesis. This serves the needs of the distributed system at various levels of communication routing, instrumentation programing and on-board instruction exchange and execution. It is presented to the user in a transparent manner and makers have no role in designing the overall system that enables communication between end points. The proposed instrumentation tools may be structured in multiple levels.

At the lowest level, it requires specifying the exact format of commands that are exchanged between the devices and the user interfaces. Then, once the specifications are established, it requires implementing flow control and queuing mechanisms of messages with respect to the hardware capabilities in the P2P RAL system. Finally, it must allow creating and supporting the actual thread of commands and co-ordinate the input of the user and the output of the experiments online.

Apart from this, there needs to be search and discovery mechanisms as well as optimised schemes for routing the commands and the data between the devices and

the user interface. As the P2P RAL is designed as an IoT system, the physical layer or the communication medium can use a mixture of a variety of technologies. It is considered that the system will use any such technologies available to connect to the Internet such as Ethernet or Wi-Fi or some other technologies to communicate locally.

It may be noted that the underlying network architecture to enable Peer-to-Peer remote experimentation, the actual network setup may be not a true P2P network. Peer-to-Peer Systems can be defined at two levels - *conceptual*/*service* model and *implementations*.

At a conceptual level a P2P system has multiple nodes that can connect to each other in a stochastic manner. The system does not know when and which set of nodes will communicate and for what purposes. The system must establish the communication without any need for a service provider. However, in an implementation level it is not always necessary to not have any centralized node. These centralized nodes are transparent and provide minimal services in setting up the communication.

A true P2P system such as Chord, CAN, Tapestry etc. are both conceptually and implementation wise P2P [114]. However, P2P mechanisms such as *torrents* use a *centralized* model to implement the P2P system. Torrents have been widely classified as P2P in the literature [17, 115-116]. There is a central node that helps with the initial finding of the peers and authenticating them, but henceforth the communication is P2P. The torrent servers essentially keep a list of peer nodes that hosts the corresponding files. The reliability of such systems is guaranteed by keeping the central node in the cloud and keeping parallel computers for it.

In a similar nature, the P2P RAL system is conceptually P2P as:

- From the P2P RLMS perspective, any two nodes can appear at any time they want to connect to each other. The P2P RLMS must confirm that the nodes are authenticated to do so and provide the communication links to each other. In case of P2P RLMS it also provides some additional files only initially, which is part of the authentication.
- Any two users of the systems, one maker and one learner can communicate ad hoc without the need for the experiment hardware being hosted at a centralized location. The maker and learner can communicate as they want.

The fourth requirement of ensuring availability can be ensured by new scheduling

mechanisms.

Finally, the last requirement of providing support to the makers/users is addressed with an enhanced form of the *smart devices* paradigms [45]. This aspect of P2P RAL is most important in context of the White Box Approach adopted in the RALfie system. A set of tools in form of algorithms and procedure are described here which is based on the generic description of the P2P RAL CI and CU.

3.6 Research Questions

Following on from the observations above, the following key research questions are being addressed in this thesis:

Q1. What is the most suitable end-node architecture that incorporates:
   a) Control of experiment rigs with transmission and execution of instructions in a transparent manner (including flow control and queuing of instruction messages)
   b) A flexible architecture that can be used to implement several experiments that adhere to the protocol a common hardware and software platform.

Q2. What are key intelligent tools required for the P2P RAL?

Q3. What are the key QoS parameters in the P2P RAL system design and how can these be optimised?

Q4. What is the ideal scheduling scheme for peers given that access to resources is limited in an RAL?

Q5. How can the usability and reliability of such a system be verified?

3.7 Contributions in Detail

To address the above research questions, a comprehensive research program was undertaken. Figure 3.8 and 3.9 show the research aspects of developing the P2P RAL architecture in more detail. It includes problems, corresponding solutions and the contributions of this thesis. The core themes are shown in the black boxes and detailed below.

i. ***End node design:*** The end nodes architecture describes the way an experiment controller need to be constructed and how it should be communicated with. The controller structure and the communication methods are used to establish an environment to program multiple experiments.

The contribution is a Finite State Automata (FSA) based architecture that is directly implementable with MCUs such as Arduino, LEGO Mindstorms etc. This architecture is termed as *twin-FSA* model. This type of end-node design can be implemented as a generalized hardware platform. The automaton also provides the basis of the programming language required to create the experiments. It uses new and specific message formats and transmission techniques to control these low-cost open source MCUs to control them through a user interface, based on the underlying Peer-to-Peer network architecture. This contribution addresses research question Q1 and is discussed in Chapters 4 and 5.

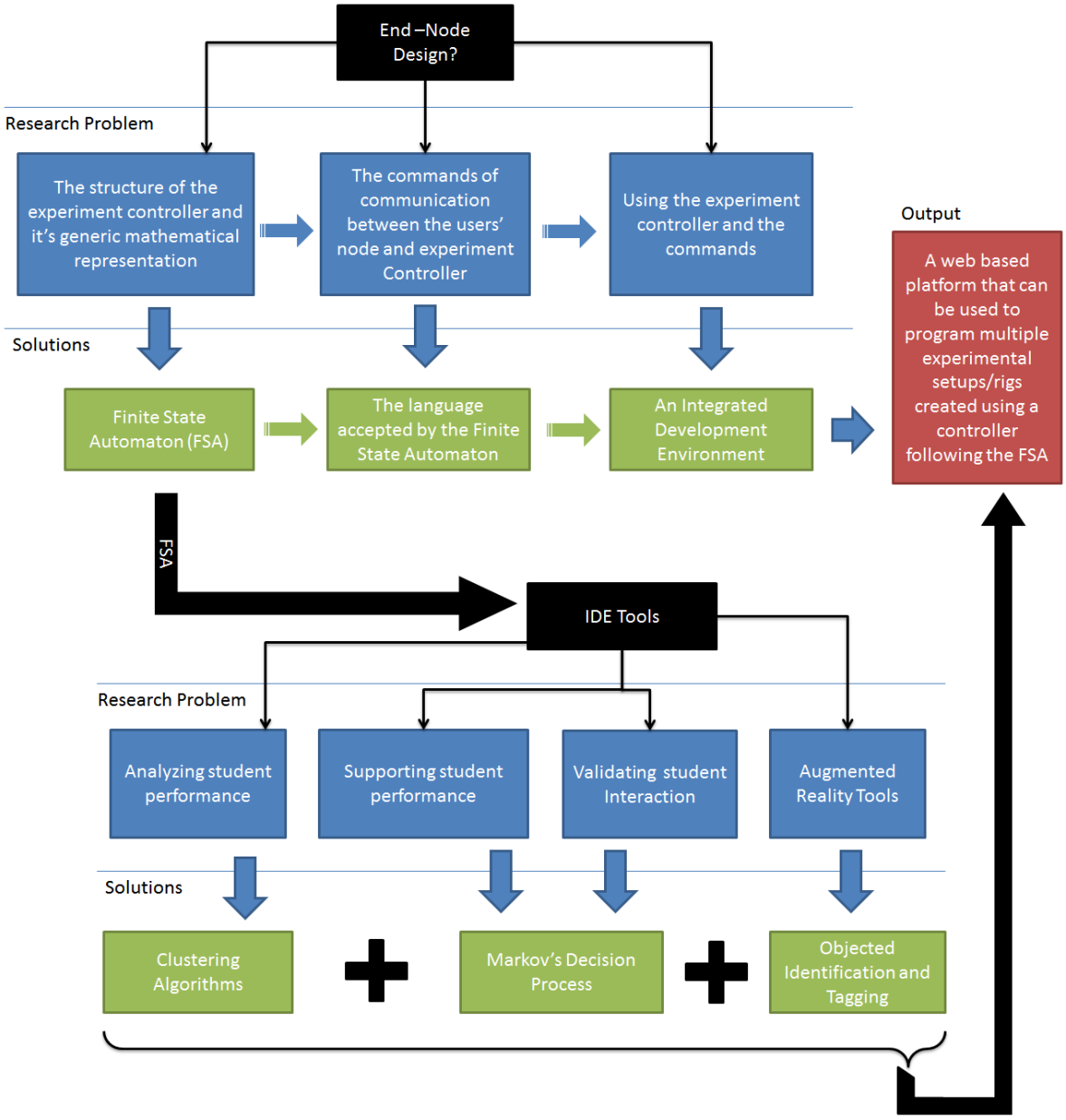


Figure. 3.8. The research aspects of the P2P RAL system with regards to end-nodes architecture.

ii. ***Intelligent IDE Tools:*** The FSA model is extended to include smart capabilities in the MCUs so that the experiments can themselves analyse, support, validate and enhance users' experience.

The new contributions include a clustering algorithm to analyse users' interaction, a Markov Decision Process (MDP) model to validate and support the users experience and objected identification and tagging procedures to enhance users experience with augmented reality. With these tools the RAL experiments can identify user's behaviour and support the learning. It can also be used to make correct transitions in the rig to make them safe to operate. This also helps in identifying certain usage patterns in the system. This contribution relates to Q2 and is discussed in Chapters 6 – 8 and Chapter 12.

These new contributions are then implemented in a web-based platform as described in Chapter 11.

iii. ***Network Performance:*** For RAL networking the Quality of Experience (QoE) objective parameters is *round trip time* or *latency*. New clustering algorithms are created in order to provide good quality learning experience. They were tested with simulations in laboratory on computers to minimize these parameters. This contribution addresses Q3 and discussed in Chapter 9.

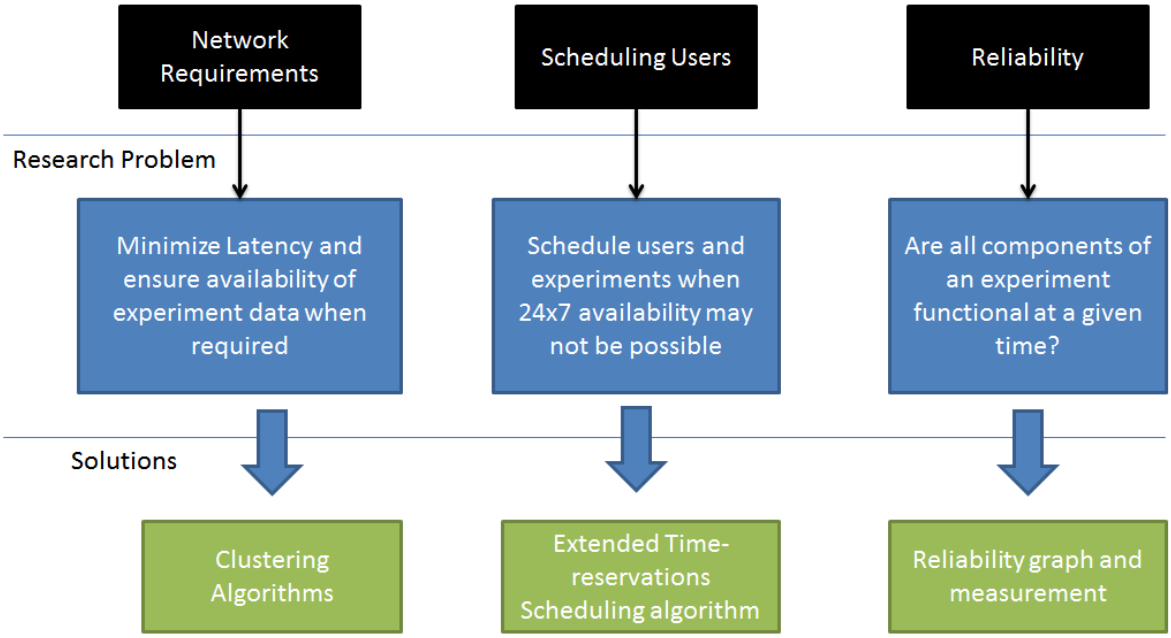


Figure. 3.9. The research aspects of the P2P RAL system with regards to network, scheduling and Reliability.

iv. ***Scheduling:*** A new scheduling algorithm for P2P RAL where availability is not expected to be absolute and makers can take their experiments on and off the system. This contribution corresponds to Q4 and discussed in Chapter 13.

v. ***Reliability:*** A method to measure the reliability of the P2P RAL system and compare it to the centralised systems. This contribution is with regards to Q5 and discussed in Chapters 10.

The first two research aspects of end node design and intelligent tools are aimed at creating the fundamental architecture of the P2P RAL system that can be used for STEM Education. The other three are for enhancing the performance and additional features of the basic P2P RAL architecture.

3.8 Methodologies

Evidently, answering each research question requires in-depth literature review into the state of research in each corresponding fields of machine learning, data mining, reliability theory and *enquiry based learning* in STEM education to formulate the solutions or new contributions of the thesis that help the P2P RAL. As such, each Chapter addresses a unique aspect of the P2P RAL. The literature review corresponding to each research aspect e.g. machine learning, data mining and STEM education have been discussed at the beginning in Chapters 4, 6-13.

Certain QoE Parameters for end-nodes that are subjective, including user-friendliness of the UI, the number and types of devices supported and the performance of the devices were evaluated on the user feedback and from device logs. Note that the aim is to prove the usability of the architecture rather than measure the real impact of the tools. The three main methodologies that were used include mathematical modelling, simulations and test-bed implementations for testing network performance, scheduling and intelligent tools.

The main original contributions of this thesis are

- A number of Algorithms 6.1, 6.2 for performance evaluation and validation, Algorithm 7.1 and 9.1 for clustering, and Algorithm 13.1 for scheduling.
- the CI-CU model and corresponding performance analysis of MCUs

The following table summarizes the broad method of obtaining and analysing the results for the above:

Contributions	**Validation and Testing methodology**	**Purpose**
Algorithms 6.1, 6.2	Testbed Implementations and Simulation	Intelligent tool for validation and support based on MDP
Algorithm 7.1	Testbed Implementations	Intelligent tool for advance evaluation based on clustering
Algorithm 13.1	Simulation Only	Scheduling algorithm for creating time reservation for users
Algorithm 9.1	Simulation Only	Clustering algorithm to determine Nano Data Centre sites
Performance analysis of MCUs	Testbed Implementations	Testing the suitability of the MCUs as the CUs of the CI-CU model

There is no global data collection. All of these algorithms have been tested by writing *computer programs* and testing them with relevant hardware. In each Chapter 5-10 and Chapter 12-13 a dedicated section states the testing conditions or methodologies used to obtain the results. The test setups are different for each chapter and in some cases it is simulations while others used actual experimental setups.

User interviews are also recorded and analysed to establish the usability of the system. The conditions for such interviews and associated activities are mentioned in Chapter 11.

It may be noted that while some of the questions are RAL specific (for e.g. Q4 about scheduling) as they are driven by the impact of interaction between the human and machines, other questions are applicable in larger context of IoT. In case of reliability measurement, the process includes human factor, but excluding that makes it more general to IoT application. Similarly, the network setup and communication routing can also be used in other relevant IoT applications. However, the design for experiment control is largely based on RAL and apart from the basic automation architecture; other contributions regarding intelligence and evaluation are RAL specific.

4

Peer-to-Peer Control System Architecture

> *This chapter presents an automaton-based model of the experiment and a communication protocol that can be used to control the experiment remotely.*

The P2P RAL architecture aims to increase flexibility in designing new experiments by enabling users to create their own rigs. This involves the control aspects as well as the user interface design. In order to implement a P2P RAL system, a suitable hardware platform must be used to create rigs that are robust, network capable as well as easy to use. Once designed, the rigs have to be programmed to communicate with the system and accept commands and send results, which then have to be mapped to a particular user interface. These core technical aspects of P2P RAL control system provide the context of this chapter. It presents a modular peer-to-peer architecture for distributed RAL instrumentation and control where any user or any experiment can be joined or removed at the users' discretion. The design of the RAL system is centred around the use of micro-controller units as the key motion control and decision making component in an experiment rig.

Each RAL experiment conceptually consists of two node types: a master, the CI at the client side and a slave, the CU on the rig. Both are connected through the Internet. The challenge is to develop and deploy an overall *supervisory unit* that governs the multiple master-slave node combinations. While the *supervisory unit* is not aware of node properties or operational capabilities, it is required to provide access control and authentication across entire the system. Thus it is necessary to develop a generic model for a CI-CU pair that enables the supervisory system to monitor and validate

the interaction between each of these pairs.

The major contribution of this chapter is this CI-CU model that allows the construction of a common web-based platform acting as flexible middleware [117], implementing uniform control method for heterogeneous hardware [118]. While the model is generic and useful for various IoT applications, the focus here is specifically on the application to a P2P RAL system exploiting these advantages. It can be adapted for any distributed network controlled and monitoring system for example, home automation and other IoT Applications.

The RAL experiments can be described as a Discrete Event System (DES) that consists of two Finite State Automata or Machines (FSA or FSM): *S* as the Controller Interface and *Y* as the Controller Unit operating in unison. It presents a generic and flexible model of the experiment rigs, and the language utilized by the two FSAs, which forms the foundation of the web-based platform, the communication protocol and the CPL required to operate the rigs. Low cost Micro-Controller Units (MCU) for example, Arduino etc. are the ideal CU for P2P RAL and the proposed architecture may also be useful for other MCU based applications. Different configurations of the rigs to achieve this modular distributed architecture are presented. The feasibility of existing electronics devices to realize this framework is also discussed.

The usage scenario of P2P RAL is discussed in Section 4.1 and literature review covering different controller and control technologies are discussed in Sections 4.2 and 4.3. The proposed *generic model* for experiments is described in Sections 4.4 to 4.7.

4.1 Usage Scenario of P2P RAL

P2P RAL aims to enable makers with limited expertise to progressively create their own experimental setups with low cost components. This can include repeated attempts, and possibly in collaboration with peers [118-119]. The P2P RLMS is the *supervisory unit* in terms of IoT that must monitor and validate the user interactions with the instruments with universal tools based on the CI-CU model. It provides search and discovery of active online experiments, data storage [118] along with the generic web-based platform in which makers can start their development. Communication is done through a VPN [108] or overlay network, which provides

connectivity between learner nodes and maker experimental rigs. The CPL/UI created by the individual makers are then stored alongside the experiment details in a repository in the RLMS.

In use, the process of creating and sharing experiments for P2P RAL is similar to the centralised RALs. It involves three steps of assembling, programming and publishing. First, the makers assemble a rig consisting of sensors and actuators. Each sensor and actuator is connected to the Controller Unit that is mapped to a unique identifier. Secondly, the makers create a user interface and corresponding Control Program Logic i.e. the CI that would drive the experiment based on the user's inputs in the UI. Finally, once the makers are satisfied with the construction and operation of the rig, it can be published i.e. made available to users on the Internet.

When the experiment is accessed, the CPL and UI created by the maker is downloaded to the user device and run in the CI. Each experiment operates in a one-to-one communication mode. But for P2P RAL there are two main issues of experiment control.

First, different makers come from unknown technical backgrounds and may be unable to create an interface without a standardized mode of communication and CPL/UI design. A uniform web-based platform allows uniformity in the design of interfaces, which is important in an educational setting.

Secondly, when defining a generic web based platform, there has to be sufficient flexibility to enable makers to host various types of rigs. Flexibility can be ensured if the protocol can support a control system with the least restrictions.

This context of a decentralised RLMS in a P2P RAL leads to two constraints for the development of the generic model. First, The operation of the CU cannot be specific for individual experiments. It has to be on a generic open platform. Secondly, the paradigms for experiment control must be independent of the CU platform. As such, only a small set of commands should be defined and executed by all CUs.

The CI can be created and run on any platform as long as it is able to address and follow those paradigms. The P2P RAL, instead of a single remote controller with single control equipment, requires a CI - CU model that is flexible enough to control open-ended experiment designs in multiple configurations with multiple actuators and

sensors.

It may be noted that the generic CI/CU model presented here is also applicable in case of a centralised RAL, provided it conforms to the model described here. However the model is important in context of P2P RAL as a universal set of features based on the generic model are required to create a platform-independent CPL/UI. These features enable the RLMS to support various user-experience and performance-related functions such as activity evaluation, validation and guidance.

These features can be specifically implemented for each centralised RAL experiment, but in a P2P RAL the makers are expected to focus more on the experiment's creative learning objectives rather than the automation overhead with regards to user-experience related issues while creating a rig. Thus a universal set of RLMS tools can enable the makers to design and construct the rig, CPL and UI with minimum deliberation. Also the RLMS can monitor step into any experiment session when required to support or enhance user-experience.

Also in a centralised RAL, portability of CI design is very poor between different systems. The federated RAL systems only allow simple access of the same experiments across different lab systems but no way to share the CPL/UI [120]. The P2P RAL's tools based on the generic model can enable wide-scale sharing and collaboration among the users and makers.

The particular pedagogical needs for a P2P RAL are to share experiments and consequently enhance collaboration in order to increase student's interest in STEM are based on sociological factors [121] which are discussed in Chapter 11 in details.

4.2 Related Work – Hardware and Architecture

This section discusses the related literature on various hardware platforms available for remote instrumentation.

Remote Instrumentation and Grids

Grid computing is the collection of resources at several locations that work towards a common goal. Unlike distributed systems these are loosely coupled i.e. they share no knowledge about other separate resources in the grid. They mainly address the requirements related to computational power and data storage for computer based

applications. Recently instruments have been incorporated as a resource in such grids to enable grid instrumentation. A grid based RAL architecture has been proposed by [41]. It incorporated a three tier setup - an internal serial *remote lab bus* connecting Web-based control units and all other physical components, a *bus protection unit* to authorize access to control units and a *protection unit* to check the validity of the commands executed to protect the instruments.

Grid based network resource allocation optimized for quality of services parameters for remote instrumentation has been implemented in [122]. The GRIDCC [122] project has used simple, straightforward procedures for adoption of the grid technologies to run instruments remotely. Instruments are represented in the architecture as an abstract format called Instrument Elements (IE). IE details are stored in a centralised information system. It uses the web services methods to communicate between the sites. The instrument element design has also been used in [123124] to describe a standards-compliant model for the representation of instruments in a grid and for booking of instruments in advance (time-booking) or immediately (queue). However, grids are complex to build and maintain. Grids are also more static in structure and topology throughout their operational period. Moreover the proposed distributed RAL system needs to operate between independent and dynamic users directly. This is difficult to realize with a grid system.

Remote Control for Reconfigurable rigs

Programmable Logic Controllers (PLC) have been integrated with the SCADA (Supervisory Control and Data Acquisition) system, usually used for automation of manufacturing to create a highly reconfigurable RAL architecture [125, 126]. SCADA is usually designed for monitoring and control of industrial equipment and hence not suitable for peer-to-peer remote control. It requires expensive components and complex setup mechanisms that are unfit for experimental setups for the target users. However, the basic concept of SCADA for decentralised control system such as data acquisition, communication and presentation are applicable here as well. A multi-tiered RAL architecture consisting of remote users using web browsers, a central web server and regional experiment servers with control units is discussed in [127]. But these do not support creating rigs at the user end.

Radio Frequency (RF) based components and communication techniques for

monitoring and control system using micro-controller units has been proposed in [128]. This system focuses on ensuring a low traffic between nodes to increase efficiency. This system is however an automation system, built on components based on close proximity using RF which is different from the peer-to-peer remote control through the Internet. Another example of reconfigurable rigs is presented in [129] where household robots fitted with microcontrollers and sensors are adapted to be used for RAL. A WEB Micro-server has been developed by RExLab [130] targeted for mobile learning. Its functionalities can be expanded to monitor and control other devices. This however requires other devices to be controlled and lacks the support for being a controlled experiment rig by itself.

Thus it can be concluded that multiple ways of creating controllers for instruments and experiments have been successfully implemented previously. For the P2P RAL for STEM education, the controller of the experiment needs to be modelled with a generic architecture which may be implemented with multiple types of controller types as described above.

4.3 Related Work – Remote Control Technologies

This section discusses existing motion control technologies and industrial protocols used in automation to ascertain the required characteristics of the P2P RAL CI-CU generic model.

4.3.1 Existing Examples in RAL

A rapid remote experiment implementation platform has been discussed in [131]. The solution uses an embedded controller, MATLAB/Simulink for creating the experiment control algorithm and LabVIEW for the user interface. This combination of software and hardware allows quick and easy deployment of various interactive remote control experiments. However, these use LabVIEW and MATLAB to control a single type of controller. As the distributed P2P RAL is consumer driven, it should be able to use multiple controllers.

A smart-device-based approach to empower the clients side has been presented in [45] where the aim is to make the remote 'smart device' ubiquitous and autonomous. It outlines the requirements and characteristics of using such devices in RALs. The minimal requirements of the smart device paradigm are also incorporated here,

namely ‘state measurement’ in form of READ instructions and ‘state control’ in form of WRITE instructions. However, the described approach in [45] does not incorporate the paradigms into a common web-based platform for all users and experiments.

4.3.2 Industrial Protocols

There are existing standardized instrument control protocols like LAN extension for Instrumentation (LXI) and Common Industrial Protocols that contains Control Area Network Bus (CAN) [132-133], Highway Addressable Remote Transducer Protocol (HART) [134], Ethernet/IP [135]. However these are not usable for a distributed remote laboratory with individual users as:

- Either these technologies are not based on the TCP/IP protocol (such as CAN and HART) which is needed to connect through the Internet or the MCUs are not compliant with them (such as LXI and Ethernet/IP).
- They are platform and hardware specific and require specialized compliant hardware for operation. Hence they are mostly used by industries and limited in educational uses. For example, Agilent devices are compliant with GPIB and LXI are widely used in RALs, but it is costly to interface it with the Internet.
- They are constructed as client-server application and not optimized for Internet based peer-to-peer operations. The topologies supported by these protocols are not ideal for P2P communications through the web and thus not suitable for the modular architecture of a distributed RAL.

However, the characteristics of these are similar in that they advocate transmitting the smallest amount of information in the quickest time possible, use frame or packets to encode this information, and support a fixed but large number of commands that are passed in the frames/packets and understood and executed at the instruments end.

These are not suitable for operating ‘ad-hoc’ rigs with both motion control and decision making elements, created by individuals with MCUs over the Internet. As the distributed RAL operates on the Internet, many of the features that are reliable in in localized implementation, such as the periodic clock synchronizations, are not effective.

4.3.3 Motion Description Languages and TeleRobotics

Motion Description Languages (MDL and MDLe) [136] give context free grammar that can be used to control continuous systems such as robots using a set of atomic behaviours, timers, and events. This approach works with self-autonomous robotic systems without human intervention and input. This language uses an atomic instruction $\sigma = (u, \xi, T)$, that applies a input u to the robot until there is no interruption ($\xi = 1$) or the Time period T is expired. This process of unit instruction is effective for MCUs in RALs. Context Free Grammar (CFG) has been used to describe the motion of robots [137] where the CFG is used to model provably correct controllers for hybrid dynamical systems with context-free discrete dynamics, nonlinear continuous dynamics, and nonlinear state partitioning. The advantages of using such automata based grammar is that it provides a balance between representative power i.e. to explicitly describe the motion of the robot and computational efficiency. It allows the users to create a generalized architecture that can generate multiple varieties of controllers and robotic apparatus. These can be used to design variable controllers, but P2P RAL requires a specific design that can be used to represent multiple controllers without changing the grammar.

Tele-robotics deals with the control of semi-autonomous robots from remote locations [138,139]. P2P RAL follows the tele-robotics principles and uses the Internet as a medium of communication to exchange control commands. In the P2P RAL humans control experiment rigs based on the sensor inputs from it. In [139] it is shown that the variables associated with teleoperation such as the quality of teleoperator interface and network quality may seriously affect the telerobotic operations and system performance even if a stable system is obtained and maintained, hence the importance of uniformity in CI design and quality. Also, the issues of security and reliability in industrial robotics can be traded off against flexibility in design for P2P RAL.

This work focuses only on tools and methodologies for experiment design in a distributed RAL architecture with its different possible configurations and investigate the usability of the suitable devices (MCUs) using a message based protocol for communication to implement these peer-to-peer arrangements. MCUs are proposed to be used as the fundamental building block in experimental rigs as well as core control components of the real-time system where the remote instruments must respond to all

users' input within a given deadline.

4.3.4 Standardization and messaging protocol for distributed control

Standardization efforts have been made for Internet-based distributed measurement and control (DMC) system before [140]. This has been incorporated into the IEEE 1451 smart Transducer Interface standards. The sensors or actuators developed with IEEE 1451 standard have a physical memory chip component in the device. This memory chip enables self-identification with stored information such as manufacture name, identification number, device type, serial number, etc.,. It may also contain calibration data depending upon the device. This information is referred to as the transducer electronic data sheet (TEDS) which can be upgraded in the system.

In context of P2P RAL however, the experiments needs to be setup by individual users with basic components that may not have any common structure for information. The makers who construct the experiments are not expected to create similar data sheets. The IEEE 1451 does not provides a generic control systems model of the transducers and it will be difficult to maintain a standard common interface to the programming interfaces for the RAL experiments at the makers site with different hardware. This is because even if the hardware follows the IEEE 1451 they do not have the common set of commands which is vital for educational purposes is it is to be incorporated into a curriculum for teaching and sharing the experiments. This also prevents the RLMS to identify the commands and provide automatic analysis and services based on the learners' inputs as required by the P2P RAL.

A platform based on the Extensible Messaging and Presence Protocol (XMPP) has been proposed in [141] with the aim of development and provision of services for highly distributed infrastructures with heterogeneous devices. XMPP was proposed as a suitable protocol to provide real-time communication. XMPP is XML based protocol for fast and efficient exchange of data between devices. The XMPP has gained wide acceptance as communication protocol in the IoT systems [141-142]. It has been standardized by the IETF and several computer languages incorporate the XMPP protocol stack [rfc6120]. An IoT like architecture had been suggested as a possible future direction for large scale deployment of RALs [143]. This was based on the idea of using XMPP to exchange commands and data. Although XMPP has not been widely used for RALs yet, the proposed idea of encapsulating the commands and

data for inter-communication between labs have been used in federated type lab infrastructure [144]. Also, IoT Systems with private/public IP systems using XMPP has been proposed for IoT Systems [142]. This means that a device does not have to possess unique public IP address on the Internet. As long as it is directly addressable with a unique URI (web link) and specific commands can be send to it, a device can be part of an IoT system.

The concept of having a message based middleware is used in context of P2P RAL as well. However, XMPP is primarily designed for exchanging message which has been customized for several IoT applications [141-142]. The P2P RAL on the other hand requires a messaging system that can be used to handle a generic control system that can be extended to various experimental rigs. It requires semantics that can be used for controlling a robotic apparatus based on the generic model and that can be further processed for validation and evaluation purposes.

4.3.5 Automaton and DES Controllers

Automaton has been previously used to express control systems. An evolutionary methodology to automatically generate Finite State Automata (FSA) controllers to control hybrid systems has been proposed in [145]. The transitions are described as specifying the new states corresponding to the input. This approach however, requires training periods to find optimal controller policies and also requires the developers to accurately create the bond graphs of the rigs.

The states of a mechanical system have been analysed in symbols/language generated in automaton form in [146] for finding erroneous behaviour. In [147], a discrete-event-type controller is proposed to meet particular specifications, designed as FSA and implemented on FPGA platform is reported. FSAs have been shown to have the greatest potential for sequential DES control. Supervisory Control Systems (SCT) is used to control DESs and make sure that the performance is in accordance with specified expectations [148]. The DES is described as an automaton process,

$$\mathfrak{Y}_x = (X, \Sigma_\Lambda, \delta, q_0, X_m) \qquad \ldots 4.1$$

in the uncontrolled model, where X are various states of the system and X_m is a marked or final acceptable state i.e. end of a given task. $\Sigma_{\mathfrak{Y}}$ covers all the events that are possible in the system. $q_0 \in X$ is the initial state and δ is a partial transition

function. Whilst this automaton is able to capture a flexible description for an experimental rig in the P2P RAL, it can have a controller only if the parameters for δ are known. For P2P RAL, the final capabilities for design of an experiment setup are uncontrolled as the users may design activities with greatly varying capacities and functions.

Thus, the specific characteristics that need to be associated with the P2P RAL protocol may be summarised as:

- The experimental rigs must be designed around a controller capable of following the smart device paradigms [45].
- Commands must be precise and short. They must be atomic to ensure best control capabilities as seen in case of MDL/MDLe and industrial protocol.
- Automatons are a suitable structure to model the components of the rig [146] and it's controller which may be used to describe flexible experimental rigs as in SCT.

4.4 Proposed Automaton Based Experiment Control Model

Educational experiments within both RALs and P2P RAL environments can be modelled as DES with the two sides - the CI and the CU. The human at the CI (or *S*) generates action which in-turn generates discrete events at the CI causing its state to change, which is propagated to the CU (or *Y*). The CU responds with a change of its state and a corresponding message to the CI. The experiments are DES as:

- The state space of the experimental rig is a function of a finite set of variables (the actuators or sensors). Thus the state space is a discrete and finite set.
- User interactions with the system lead to transitions in state space i.e. it is event-driven.

Unlike a centralised RAL system, where the makers specifically integrate the experiment with the RLMS using typically customized software and hardware components, P2P RAL makers do not necessarily take such measures themselves and the integration is required to be automatic. Thus from the perspective of a P2P RLMS,

Figure. 4.1. The relation between the two FSAs

the δ (in Equation 4.1) is unknown for any experiment. The goal is to define a set of symbols Σ (= $\Sigma_S \cup \Sigma_Y$) such that Σ may be used in any S or Y and the number of symbols (i.e. commands) in Σ is minimum and finite.

This is achieved by describing both S and Y as two automata that share a language $L(\Sigma)$ with common symbols and strings. A change of state in either of the two is reflected in the other. The changes start with S where the human user starts the session. This leads to a generalized architecture for CI and a working model for CU that can be used to create human controlled semi-autonomous electro-mechanical rigs. The language accepted by both for the communication can produce varying levels of flexibility and complexity in rigs.

The basic system architecture is depicted in Figure 4.1. It shows the two components CI and CU are two automata connected through a network and have the queues J and K. The output of an automaton is placed in its corresponding queue (CI → J & CU → K). This forms the input for the other automaton. The two automatons are depended on each other although both of them are separate, hence the new proposed name *twin-finite state automata*.

A set of ports (R) are variables in S and Y and identical in both. Each port corresponds to an actuator or sensor address on the CU. The change in any port will change the state of the rig and the CI. Any combination of the port values is part of the experimental rigs state space. The ports used for control on the CU act as variables in the CI. The CI changes the state of an actuator variable ($x \in R$) that is reflected in the state of the rig and queries on the state of a senor to get its value. There is also a stack ω associated with the CU that stores successfully executed commands.

A command is the message sent from the CI to CU. Commands contains instructions are executed on the CU. A command may be composite i.e. perform multiple functions with multiple instructions or it is atomic, i.e. it performs only one function with a single instruction. An atomic command consists of one atomic instruction. The

CU responds with reply messages consisting of the resultant data. This includes success or failure of the instruction execution as well as sensor data.

The assumptions made in this dissertation [149] are:

- The delay in the network (i.e. time taken to transfer the message from CI to CU does not have any effect on the stability of the experiment setup. All experimental actions take place on the rig when the instructions are received.
- The network is reliable and is able to deliver the data from the source CI to the destination CU. The delay between the CI and CU could potentially be long.

4.5 Controller Interface Model

This section discusses the model of the Controller Interface (CI) as a an automata. The Controller Interface is run on the users' device accessing the remote rig. It takes input from the user through the UI, makes decision with the UIM based on the CPL and the command library and issues corresponding commands to the CU. The CI keeps the status of each rig components, actuator and sensors, in a corresponding variable array *R*. The CI can be described as a *Finite State Machine S*,

$$S = \{G, \Sigma_S, \beta, \beta_0, E, J, R\}$$

where *G* is all possible functional states in CI, i.e.

$$G \subseteq \{INIT, ASSIGN, QUERY, DISPATCH, PAUSED, IDLE, STOP\}.$$

Σ_s is the instruction set, i.e.

$$\Sigma_S \subseteq \{\, \mathbb{s}, \mathbb{r}, \mathbb{w}, \mathbb{u}, \mathbb{a}, \mathbb{e}, \mathbb{f}, \mathbb{l} \,\} \cup N$$

where $\beta_0 = INIT$ signifies the initial state. *E* is the set of final states, i.e. $E = \{INIT, IDLE, STOP\}$, where the system is stationary. *N* is a set of composite commands stored into a Symbol Table along with associated events related to the CI.

A symbol in Σ_S represents a command, event or operation. An *error* symbol $\mathbb{e}$ indicates either an error message from the CU or a *timeout* in case the rig fails to respond within a time frame. This timeout threshold is determined by the actual latency (ψ) between the MCU and the CI. This is determined at the beginning of the operation i.e. within the $\mathbb{s}$ *(set)* symbol. In case the latency is dynamic and changes over time, this value will also change. However as the users will be interacting with the

rig for a short period of time (10-15 minutes), this is expected to remain static. The initialization command 𝕤 is used by the CI to configure ports at the CU and to set their initial values. The *read* 𝕣 and *write* 𝕨 instruction are issued when user actions leads to an event request values or require changes to port variable values or its value be read.

Set 𝕤, *wait* 𝕒 , *read* 𝕣 and *write* '𝕨' are intrinsic elements of Σ_I. They are generated by user actions on the CI and do not relate to messages received from the CU. *Error* 𝕖, *fail* 𝕗, and *success* 𝕦 are extrinsic elements of Σ_S exchanged in form of messages. They are explicitly sent by the CU. *Error* 𝕖 indicates that an error occurred on the rig and the CI remains in the PAUSED state. *End* 𝕝 signifies that the experiment session has ended either due to reaching the allotted time or due to failure on the CU the experiment session has been closed.

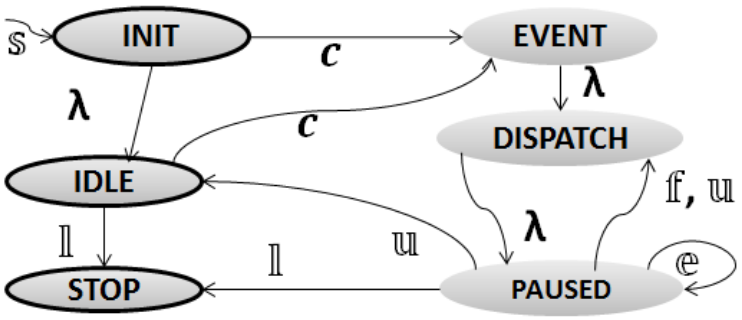
The state transition function β is

$$\beta(q, D) \rightarrow q' \quad where, q, q' \in G \; and \; D = d_1, d_2 \ldots \in \Sigma_S$$

A queue J stores outgoing messages and initially J is empty. Figure 4.2 depicts the state transition diagram for the CI. The different functional states are described as follows:

INIT - The initialization step starts with the local variables at the CI being set to their initial values. This phase is executed at the beginning of an experiment session. This state sends the '𝕤' command to the rig, which initializes the CU. After initialization, the CI starts the CPL where it can go to the *EVENT* state if there is any event from the users' interaction. Otherwise it goes to an *IDLE* state.

EVENT - This state validates a particular command $c \in \{\mathbb{r}, \mathbb{w}, \mathbb{a}\} \cup N$ against the event (such as clicking a button on the UI) by matching it in the *symbol table*. For processing only atomic instructions, this *symbol table* only contains 𝕨, 𝕣 and 𝕒. If



Symbol Table (atomic commands only)

Event	Symbol
Write	𝕨
Read	𝕣
Wait	𝕒
End	𝕝

Figure. 4.2. The state transition diagram of the controller interface (S)

the CI has to send any composite commands they may appear in the table as well. Once the command is validated, the corresponding symbols are placed in the output queue J. Any local variable $x \in R$ is updated. The system then moves to the *DISPATCH* state. If any event cannot be validated, i.e. no valid command exists in the symbol table for the event, the system moves back to *IDLE* state. If the CI wants to end the session, CI can send the 𝕝 message.

DISPATCH: This state sends the content i.e. the command message in the queue J to the CU. *S* enters a *PAUSED* state to wait for the reply message from the CU.

PAUSED - It is an idle state where the rig waits for a response from the rig. In the *paused* state the user inputs are ignored or the user interface is disabled which simply disables any user input an there is no change in *J*. Upon receiving a result $d_i(v_t)$ which could contain the required values v_t, S moves to next states. If a '𝕦' message is received, the front instruction is deleted from the queue *J*, the value of x_i received from MCU is updated on the CI. Otherwise in case of '𝕗', *J* is not altered. If there are still messages in *J* then *S* goes to the *dispatch* state. If an 𝕖 is received, the system remains in the *PAUSED* state until a 𝕗, 𝕝 *or* 𝕦 message is received.

IDLE - When the user does not gives any input and the CI itself does not have any operations to execute.

STOP – If the CI determines after repeated queries, through DISPATCH state, that the remote rig is incapable of returning the required result, it stops the experiment execution. Users are not able to use the rig until the system is manually reset. The CI also reaches this state at the end of the session.

State changes of the FSA, discussed above, are triggered by user inputs and messages sent by the corresponding FSA that represents the CU. This is described in the next section.

4.6 Controller Unit Operating Model

A model for the operation and the corresponding instructions of the CU are introduced in this section. There is a master/slave relation between the CI and CU. Instructions originate at the CI and are executed at the CU and data is collected at the

CU. As every instruction has to be executed in order, acknowledgements are important. A CU has a processing unit, a memory unit, a network interface and an array of input/output ports R. A port ($x \in R$) holds values depending on what it is connected to. All peripheral devices are controlled by these ports. Instructions executed on the CU alter the configurations by changing port values of the rig or reads data at a given point of time. An experiment setup is controlled by consecutive commands being executed by the CU.

4.6.1 CU Finite State Machine

The CU can be described as a Finite State Machine Y,

$$Y = (Q, \Sigma_Y, \delta, p, F, K, R)$$

where Q is a set working state of the CU i.e. operations that are executed. Thus,

$$Q \subseteq \{INIT, ACTION, DAQ, IDLE, DISPATCH, RESET, STALL\}$$

and the instruction set Σ_Y,

$$\Sigma_Y \subseteq \{\mathbb{s}, \mathbb{r}, \mathbb{w}, \mathbb{a}, \mathbb{e}\}$$

The state of the rig (Y) changes according to inputs from the CI or event on the rig. The state transition diagram is depicted in Figure 4.3. User driven extrinsic inputs are $\mathbb{s}, \mathbb{r}$, $\mathbb{a}$ and $\mathbb{w}$ commands. Intrinsic input $\mathbb{e}$ are triggered by the CU. The input $\mathbb{e}$ occurs when the experiment setup fails to perform any action or data collection on a specified port. In cases where a rig does not have sensors or actuators, some states, i.e. *ACTION* or *DAQ*, are not used. This also applies to Σ_Y as in these cases the $\mathbb{r}$ and $\mathbb{w}$ commands are not required. To be of practical relevance, at least one sensor or one actuator is required. The wait $\mathbb{a}$ command is used to stall the CU for a certain period of time. A *wait* command with parameter v means the CU must remain dormant for v units of time since the completion of the last instruction execution.

The control vector p indicates initial port-value tuples for an experiment. F = *{INIT, IDLE}* is a set of stable control outputs where the CU is in a stationary state i.e. there is no error generated. The state transition function δ is

$$\delta(q, c_i) \rightarrow q' \qquad where, q, q' \in Q \qquad and$$

$$c_i \in \{c_1(x_1, v_1),\ c_2(x_2, v_2) \ldots\} \quad and \quad c_i \in \Sigma_Y$$

where every command c_i carries a corresponding port(s) x_i and optionally some value(s) v_i. Additionally, queue K stores outgoing messages. Initially the queue is empty. Messages are generated by state changes. There is also a stack ω associated with the CU that stores previous successful 𝕨 instructions. The seven states of CU operation are:

INIT - This phase is executed whenever an experiment session begins. All variables that relate to the setup are set to initial values. The *initialize* command 𝕤 is a command that signifies the start of a new control session. This state may involve setting timestamps, starting new log files etc. For a system that supports languages with composite commands as described later 𝕤 may be composed of 𝕨, 𝕣 and 𝕒 instructions to set initial values of the CU ports, i.e. $\mathbb{s} = \{\mathbb{r}, \mathbb{w}, \mathbb{a}\}^*$ but for a simple CU, 𝕤 = λ i.e. empty.

ACTION – Write 𝕨 instructions $w(x, v_x)_t$ are used to control actuators. The instruction $w(x, v_x)_t$ received at time t, moves the CU to the *ACTION* state and alters the value of port *x* to the value *v*.

$$w(x, v_x)_t \triangleq x_t + v_x \qquad \ldots (4.2)$$

where v_x is the value to be written in x at time t > 0. Once a $w(x, v)_t$ command for writing a value on a port $x \in R$ is started, the CU is free to execute any new instruction if available. It first changes to *IDLE* state (even if momentarily) and then to *DAQ* or *ACTION or the STALL state*. If an instruction is completed then a corresponding success message 𝕦 is put in *K*. It can also move into the *DISPATCH* state if an error occurs. If the instruction is successfully executed, it is pushed to

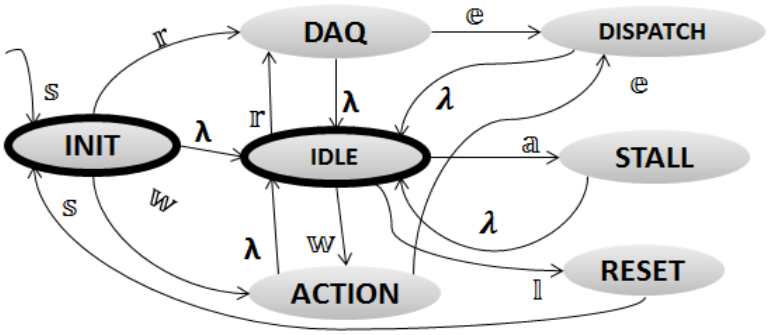


Figure 4.3. A state transition diagram for the RAL Control Unit (Y)

stack (ϖ) along with the time t.

$$\varpi \rightarrow (w(x, v_x), t) \cdot \varpi$$

Changes to a *port* variable result in a change of the state space of the controlled rig.

DAQ - *Data AcQuisition* is the step of collecting data at the rig. The *DAQ* state reads values from specified ports. A read command '𝕣' or *r(x)* will return the values at port *x* to the CI. Once a read instruction is started the CU is free to execute any new instruction if available. Similar to the *ACTION*, the system can go to *DAQ* or *ACTION or STALL state* through IDLE state. After a value is read, a corresponding 𝕦 message is created that includes the values and put in *K*. If there is an error, it goes to the *DISPATCH* state.

IDLE - The *IDLE* state is a passive state that occurs between the *ACTION*, *DAQ*, *STALL* or *DISPATCH* states. In the *IDLE* state, the CU does nothing. The *IDLE* state can be held for an indefinite time i.e. long periods or even momentarily. It occurs when CU is waiting for any input. If a 𝕝 message is received, the system goes into RESET state where the session is terminated.

DISPATCH - This state puts the error 𝕖 of the *Action* or *DAQ* (if any) into the queue K. Depending upon the number of errors in the session or the nature of known errors, this state may put the fail 𝕗 in K. Any messages in *K* are sent to CI.

STALL - This state forces the CU to remain stalled for definite period of time *v* specified as a parameter in the wait command - wait(*v*) since the finish time of last execution (t_e). This deals with any variable time latency between the CI and CU as the actual stall time will depend on when the stall command is received after t_e. If the latency is too large and *v* is lesser than the time passed since t_e, then the CU does not stall at all. Similar to the *ACTION*, the CU can immediately go to DAQ or ACTION or STALL state through IDLE state.

RESET – This state occur at the end of the experiment session. The RESET phase puts the 𝕝 message in K and does nothing until a set 𝕤 command is received when the CU moves to the INIT to start a new session. In this case it is similar to the IDLE state except that no other input that set 𝕤 is accepted.

4.6.2 CU Operation

From (Equation 4.2), for any port *x* connected to an actuator can be altered by sending an input $c_t = w(x, v)_t$ at time $t > 0$. Thus the corresponding system may be described as

$$Y_{t+1} = AR_t + BC_t \quad \ldots (4.3)$$

where R_t is the state vector corresponding to the |R| number of ports connected to an actuator or a servo and *A* and *B* are constant matrices for a particular rig. C_t is the control input vector at time $t > 0$ given by

$$C_t = [c_1 \quad \cdots \quad c_{|R|}]$$

where $c_i = w(x, v_x)_t$ or $c_i = \emptyset$ (null) or 0 if no command is given for x at time $t > 0$. The system also contains the sensors for reading the data, but 'ℝ' operation do not operate or alter the configurations or orientation of the rig directly.

The instructions may be received at any time $t > 0$ from the CI side i.e. $\Delta = t_2 - t_1 > 0$ is not a constant. The system *Y* is however time invariant as delayed arrival (ϋ) of instruction c_i only means that *Y* remains in the IDLE state for a longer period (ϋ). The instructions are executed when it arrives and the state of *Y* is changed at $t + \Delta + ϋ$. Hence,

$$Y_{t+\Delta+ϋ} = AR_t + BC_{t+ϋ} \quad \ldots (4.4)$$

Equation (4.3) and (4.4) is a general equation of control systems [150]. Hence, *Y* can be used to implement any kind of experimental rig governed by Equation (4.3 - 4). It is suitable to implement an experimental setup containing physical motion with fundamental mechatronics elements such as servos and sensors. RAL experiments involving advanced machineries or virtual components are outside the scope of this architecture.

One major difference between the CI and the CU is that the CU runs continuously without any time bound unless it faces an error and require manual intervention to reset it. The CI however starts and ends at definite points of time. Another difference is the way of handling the error. In the CU an error is actually generated due to failure in the hardware or the experiment setup. An error always ultimately resets the experiment setup. However in the CI, error does not immediately cause a STOP and

the CI waits for a '𝕗' reply before it is determined if the experiment has to be aborted.

4.7 Complex Languages

This section demonstrates how a hierarchy of complex languages can be based on atomic instructions and discusses practical implications of using complex languages in the context of RAL activities.

4.7.1 Communication Language

The language accepted by the automata forms the basis of the communication protocol between the CI and the CU in an experiment. Each of the elements in the Σ_S (except 𝕤) is an atomic instruction i.e. each of these can be executed on a CU but cannot be divided into further sets of instruction. Atomic instructions can be joined to form composite instruction that can be called as a single command. The language accepted by the CU is the regular language,

$$L_Y^0 = \{xy : x = \mathbb{s} \ \wedge \ y \in \{\Sigma_Y - x\}^* \} \qquad \ldots (4.5)$$

which means that the CU will only be an acceptable state in *IDLE* and *INIT*. The CI has a language

$$L_S = \{\mathbb{s}y\mathbb{l} : \ y \in \{\Sigma_S - \mathbb{s} - \mathbb{l}\}^* \} \qquad \ldots (4.6)$$

which means that the CI must start with a '𝕤' command and finish with an '𝕝' command. If $\eta \in L_Y^0$ then η is a word or combination of instructions sent in an experiment session in order. The CU after executing all instructions in η is in a final state $f \in F$. If $\eta \in L_S$ then η is composed of all user inputs (𝕤, 𝕒, 𝕣, 𝕨) and CU outputs is (𝕦, 𝕗, 𝕝). The CI after reading all this inputs from η is in a final accepting state $f \in E$. The actual communications to and forth between the CI and the CU essentially involves $\Sigma = \Sigma_S \cup \Sigma_Y$.

For the CU, the symbols in L_Y^0 may be concatenated to form larger fixed set of strings that can be referred by a symbol. This is creating functions on the CU consisting of several 𝕨, 𝕣 and 𝕒 symbols which are invoked by a function name. Thus,

$$\Sigma_Y' = \{\mathbb{s}\} \cup I \qquad \forall \quad I \subset \{\{\Sigma_Y - \mathbb{s}\}^+ . \{\Sigma_Y - \mathbb{s}\}^+\} \qquad \ldots (4.7)$$

and the language accepted,

$$L_Y^1 = \{ \mathbb{s}y : y \in \{\Sigma_Y' - \mathbb{s}\}^* \} \quad \ldots (4.8)$$

Σ_Y' does not support the basic elements of Σ_Y but composite words or strings from L_Y^0 i.e. only composite commands.

From Equation 4.7, Σ_Y' is a finite set of composite commands. L_Y^1 is composed of all words or strings in Σ_Y' that starts with $\mathbb{s}$. Since $I \subset \{\{\Sigma_Y - \mathbb{s}\}^*\}$, I has fewer and fixed number of symbols than can be composed from Σ_Y. A word $g \in L_Y^1$, then $g = \mathbb{s}y$ starts with $\mathbb{s}$ and the remainder y is composed of any combination symbols from $\{\Sigma_Y' - \mathbb{s}\}$ i.e.

$$y \in \{\Sigma_Y' - \mathbb{s}\}^* \Rightarrow g \in L_Y^0 \quad \ldots \text{ from Equation 4.5 and 4.7.}$$

Hence $L_Y^1 \subset L_Y^0$. However, conversely, since I has only a fixed number of symbols, if $g' \in L_Y^0$ then $g' = \mathbb{s}y'$ starts with $\mathbb{s}$ but the rest of the word (y') may be composed of a combination of $\{\Sigma_Y - \mathbb{s}\}^*$ i.e.

$$y' \in \{\Sigma_Y - \mathbb{s}\}^* \quad but \quad y' \notin I \quad \Rightarrow \quad L_Y^0 \not\subset L_Y^1$$

Thus,

$$|L_Y^0 - L_Y^1| > 0$$

This means that there are many acceptable strings in the L_Y^0 that are not present in the L_Y^1 which implies that L_Y^1 is incapable of executing certain sets of operations. This creates a hierarchical level of language with each new level (L_Y^{i+1}) building upon the previous level (L_Y^i) using Equations 4.5-8. This difference in language used can affect flexibility, complexity and network properties of the rig control.

The CI-CU automaton model can be used to describe the relationship between the flexibility and complexity of the experiments. *Flexibility* is a measure of freedom by which makers of experiments can implement the rig. *Complexity* in programming the rig is the number of different instructions that are required to create the program logic and the number of commands that need to be transmitted between the CI and the CU. Flexibility and hence complexity in the design of the rig and CPL/UI is reduced with higher level composite commands. For all practical purposes, a P2P RAL system may involve a language with relatively more number of composite commands. This is done to ease the rig creation procedure at a reasonable loss of design freedom

depending upon the makers experience and expertise.

4.7.2 Types of Commands

An experimental rig built by the users has some actuators identified uniquely. The commands when executed change the state of the rig which produces some output which could be visual movements or other data. The commands will vary in the time it takes to complete depending upon its type - atomic or composite. Each of these commands is generated at specific times the users give their inputs.

An atomic command is one that cannot be sub-divided into any more commands i.e. they are most fundamental of commands [119] – READ to read from a sensor, WRITE to write a value to an actuator changing its state and WAIT. The wait command is used to synchronize the command executions as much as possible.

The program logic, created by the makers, process the learner inputs for the UI to generate the corresponding symbol sets or communication commands composed of a combination of these three atomic instructions. Atomic commands provide greater control *flexibility* but are difficult to implement. A greater number of atomic or lower level commands must be issued per unit time from the CI to be able to successfully operate the CU compared to using a smaller set of high level or composite commands. Also using atomic commands is more susceptible to error depending upon the latency in the network. On the other hand, by sacrificing flexibility, the users can be given a set of composite commands that perform more specific tasks on the rig. This also reduces the complexity of the CPL. Using fewer composite commands per unit time reduces the traffic volume but takes away control freedom from the operator. The exact level of suitable flexibility or complexity required is dependent on the context the CI-CU model is used i.e. the nature of the experiment and the capabilities of maker in the P2P RAL.

Each input given by the users is processed by the experiment individually regardless of how many are sent at any given time. Thus the *control length* can be described as the number of steps or instruction to complete a composite command. More than one atomic command can be joined to provide specific CU/experiment related functions in form of composite commands that will take variable time to complete depending upon its constituents. The actual length i.e. number of steps within a composite command

depends on the parameters passed to it.

A basic composite command is formed from bunch of atomic commands that are joined and sent at the same time with regular time intervals between them. More advanced composite commands are parsed to generate a large set of atomic commands which may involve use of program logic such as condition checking and iterations to generate a set of sub-commands. Thus the constituent atomic commands may not always be the same. The advantages of composite commands are:

- Composite commands being executed on the rig guarantees that the timing between the sub-commands of the composite command is executed at equal intervals for any given set of parameters.
- Composite commands allow creating reusable modules of instructions that may be called without having to specify every single instruction explicitly in the program. The modules are also more easy to understand i.e. human readable as compared to a smaller atomic instructions.

Having specific modules or functions on the CU however makes the CU more akin to certain kinds of experiments that follow the functions, but unable to support conditions that do not conform to the logic or flow of the modules.

For a rig designed specifically for an experiment, the commands can all be composite i.e. specific for the experiment. The UI can send these composite commands depending upon the users' interactions with its UI. Such commands can written in a variety of languages and have safety capabilities to ensure the integrity of the rig.

In a P2P scenario, for collaborations, the makers and users (between makers as well as between makers and users) must all use a common platform to be able to share experiment and maintain a homogenous UI. Thus the set of commands for the common UI and programming platform cannot support a large number of composite commands. As there are no limit of the RAL rig configurations there can be an infinite number of such composite commands, making it difficult to create a finite set of modules to serve all possible rigs. Thus for all practical purposes in the P2P RAL, a finite set of composite command modules are provided at the expense of some control freedom.

4.7.3 Joint Parameters for Parallel Instructions and Toggle

A composite command may resolve to a set of instructions that need to be run in parallel. In such cases, the command is described as $c(X, C')_t$ that may operate on multiple ports in parallel where $X \subseteq R$ and C' is a set of atomic instruction corresponding to each port in *X*. *X* may contain multiple ports but C' can only contain instructions of the same type i.e. multiple *WRITEs* or multiple *READs* but not a mix of any two. Restricting the instructions to be of the same type can ensure that the rig does not try to read values or stall while writing to a port. In this case, the *ACTION* state or the *DAQ* state in the CU will simultaneously operate on multiple ports, but the rest of the process remains same. The outcome of the parallel operations is determined as success if all atomic instructions were successfully completed, otherwise it a failure.

In certain cases *toggle* behaviour is necessary, in which the command requires that the CU holds the value of a port for a certain period of time before resetting it back to its previous state. Toggling is composed of two different *WRITE* commands, but as the time between the toggles is very small and the latency ψ between the CI and the CU could be high, the MCU has to perform the toggle by itself. The instruction will itself specify the toggle property. This is essential a composite instruction $'\mathbb{w}a\mathbb{w}'$.

4.7.4 Inverse Motion

Inverse motion is required if the rig has to roll-back on its states. It is applicable only when an $\mathbb{w}$ instruction fails to complete successfully. If the CU encounters a failure at $c_i^t \in c(X, C')_t$, while executing the composite command $c(X, C')_t$, it cannot proceed with any other already executed instruction in $c(X, C')_t$ and must roll-back. To reach a stable state, it must roll-back all of the atomic commands c_i^t that have been executed and stored in stack ϖ. Any current execution is stopped. The CU starts to pop instructions from ϖ and executes them according to the difference of the time between it and its previous instruction as recorded in ϖ. As the instructions are unique and not relative to the previous instructions, each will take the rig back by one step.

At any state the system is able to successfully complete the all $c_i^t \in c(X, C')_t$, and $c_i^t \in \varpi$, *Y* is again in the stable state. At this point a $\mathbb{f}$ message is generated. If none of the instructions in ϖ can be executed successfully, then a $\mathbb{l}$ message is sent signifying

to stop/abort the session. The role of executing this inverse motion and the messages has to be taken by an external component to the CU. But despite using ω, it may not always be possible to restore the rig to an active state and human intervention may be required to reset the experiment; for example, when a robotic car is overturned.

4.8 Using the CI-CU Model

The twin FSA CI-CU model provides a generic model for experiments in the system. This allows for a common CI and CU software to be used for all experiments. The actual platforms for the experiment can be different i.e. the CIs may be run from browser or stand-alone software and the CUs can be run from many hardware such as Arduino, LEGO Mindstorms etc. These extensions of the CI-CU model are discussed in Chapter 5. Chapter 5 provides the technical feasibility of using the model with MCUs.

The CI-CU model addresses the core issues for underlying motivations of validation, guidance and evaluation as discussed in Chapters 6-8 and 10. In Chapter 6, an MDP is constructed based on the model where every instance of the values of the ports in the CU is a state in MDP. Since the CU has a finite set of ports with finite limits of their values, the MDP has a finite state space. The MDP is further used for validation and guidance. In Chapter 7, a clustering method is proposed based in the temporal locality of the commands in an experiment session based on the CI-CU model. In Chapter 8, the different levels of commands possible as with the CI-CU model are used to describe the experiment interactivity continuum. In Chapter 10, the reliability graph contains the specific components of the experiments as described in the CI-CU model - the peripheral devices (sensors and actuators), CUs, Network system and learners and the four components of the reliability graph. Chapter 11 shows many implementations of the CI-CU model in form of various example experiments.

4.9 The CI-CU Model as IoT

As mentioned in Section 2.2.3, the CI-CU model makes the P2P RAL an IoT system or more specifically a subclass the Web of Things (WoT). The CI-CU model covers all the required characteristic of an IoT/WoT as:

1. Each CU is uniquely addressable on the P2P RAL system. The CI when running from the users' device can connect to CU with web link and send commands through it.
2. The IEL provides a uniform set of commands for all the MCUs.
3. The CPL of the experiments describes the state space and the constraints of the rig. Also the UI presents the ways to communicate with the rig.
4. The constraint 4 is not directly applicable to P2P RAL as all the MCU has the same set of services thus exploration is not requires. The CI is all cases are aware of the IELs capabilities. However, the models created for a particular experiment based on MDP or Clustering as described in the next chapters provides unique model of the experiment to each CI. This can be retrieved and all commands can be validated or evaluated against these models. Using these models can be regarded as explorations new services specific to the experiments.
5. The communications are done using WebSockets and HTTP.

4.10 Possibly Expanding to Many-to-Many CI-CU

The CI-CU model can be extended to a many-to-many or one-to-many architecture in a networked control system. This section briefly discusses the issues that need to be addressed in this regard, although the actual methods to implement it is beyond the scope of P2P RAL defined in this thesis.

Both the CI and the CU contains queues J and K. If one CI communicates with multiple CU in one session, then the queues are filled with multiple messages regarding the CUs. There are also multiple R_i sets corresponding to each CU Y_i in the CI. But the CUs all have only a single set R.

The PAUSE state behaves differently in this case. The CI can totally pause and accept no input at all or the CI can stop accepting inputs with regards to a particular CU that has not responded to the last command sent. The commands can be depended on each other if the CUs are to be operated simultaneously in correspondence with each other. There can be many CI connected with many CUs in the same session. More than two CIs communication with the same CU means they have to coordinate which commands have priority while making a request. There are several issues with these multiple nodes in the same session such as:

• Time delays: If the CUs do not operate in mutual exclusions, then the controlling CIs face problem with unknown network delays or jitters. The CIs must ensure that the commands structure change with the required level of interactivity by choosing appropriate level of composite commands.

• Concurrency issues: Having multiple CI-CUs that are interdependent can result in the 'circular wait' conditions if the two CIs S1 and S_2 tries to operate the same CU Y_1 in opposite conditions each trying to negate the other's command. To eliminate this situation any group of CI with control of the same CU must be in communication with each other.

4.11 Summary

A generic model for RAL experiments have been discussed that can describe the operation of multiple experiments. This generic model can be used for further expansion of common utilities that can be provided as part of RLMS. This model facilitates the development of a P2P RAL architecture that allows for virtually unlimited individuals to create and share their experiments over the Internet. An RLMS based on the P2P architecture with the generic model can seamlessly integrate any experimental rig automatically. The generic model is easily implemented with MCUs such as Arduinos and LEGO Mindstorms as described in the next chapter. The model is also the base for the extensions that are discussed in Chapter 6 to Chapter 8.

With respect to IoT the CI-CU model can be used in any situation where a large number of master-slave nodes exist and the interaction needs to be monitored. A supervisory system with such capabilities can govern a large number of nodes with varying properties and control policies. It can also ensure the security and integrity of the system if required. While the following chapters focus on tools primarily aimed at aiding RAL experiments, the CI-CU model itself can be used to develop many other types of tools to supervise different IoT systems as per requirements.

5

Implementation using Micro-Controllers

> *This chapter describes how the CI-CU model is used to implement the P2P RAL. It looks into the software and hardware examples that can be employed.*

In the previous chapter a generic model was introduced that governs the CI – CU interaction. In this chapter a practical implementation in the context of a P2P RAL system is discussed. The CU evolves to the Instruction Execution Module (IEM) and is the experimental rig designed by makers implemented using LEGO Mindstorms for example. The CI evolves into the User Interaction Module (UIM) interface stored in the RLMS implemented using SNAP as a case study.

The first two sections discuss the control strategies and a basic implementation of the CI CU model in a P2P RAL context. The key contributions discussed in this this chapter include a detailed analysis of different MCU platforms in Section 5.3 and a flow control and queuing method to exchange messages in Section 5.4. An example of an implementation is discussed in Sections 5.5 and 5.6. The contents of this chapter are based on publications [118-119].

5.1 Control Strategies

The interaction of the UI with the experiment in P2P RAL can be divided into two separate steps as shown in Figure 5.1. First, the user interactions with the UI are converted to corresponding commands. Second, the commands generated in the first step are converted to an atomic command i.e. $\Sigma = \{\mathbb{r}, \mathbb{w}, \mathbb{a}\}$.

The first conversion step is done in the UIM based on the CI model and the second conversion can be attached as a Complex-Basic Command Translation (CBCT)

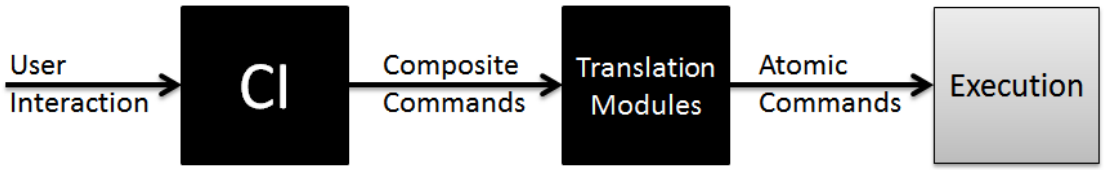


Figure. 5.1. The user interaction to atomic commands conversion process.

component to the CU model for implementation on the experiment site.

In terms of P2P RAL the experiment control strategies can be broadly divided into two categories: *Direct Access Control* and *Undirected Access Control.*

For *Direct Access Control* all instructions originate at the CI of the experiment according to the users inputs. The commands are sent to the CU where they are executed and results are returned in the CI. The CU does not have any decision making capabilities regarding the control logic of an experiment, except when the commands poses a safety threat to the experimental rig.

This kind of access natively supports *interactive* experiments as the users can issue multiple commands in a short period of time. However, batched experiments can be run in the same way, if the users are allowed to give a single input set at the beginning of the experiment session and subsequently the CI issues all commands.

For Undirected Access Control the CU has partial decision-making capabilities in regards to the control logic of the experiment. This requires the CI and CU to be synchronized. The CU contains the specific functions for the experiment that are invoked from the CI and the subsequent results are returned to the CI. This kind of control natively supports the batched experiment and is unsuitable for interactive experiment.

Both of these methods are supported by the CI-CU model described in Chapter 4. In the following sections, a real implementation of this model is discussed primarily focusing on *direct access control*. *Undirected access control* is discussed in Chapter 8. The following section discusses how the model is implemented.

5.2 Software Implementation of the Twin FSA

The P2P RAL rig operation is based on the Twin-FSA and shown in Figure 5.2. It includes three key system components: the RLMS, UIM and IEM.

The UIM is implemented as an extended component of the RLMS. When the system page is loaded in a browser, it is downloaded to the remote controller device. The UIM for the P2P RAL is a static environment providing basic programming tools to create a CPL and UI. The UIM provides a uniform programming and UI platform, which is available to both makers and learners. Learners cannot alter the CPL or UI but use the interface.

The UIM contains the CI and associated structures such as symbol table (or command library), Control Program Logic and the variables representing ports. The command library stores an event-symbol string pair that represents a set of command symbols associated with any event in the UIM from the user's interaction or any other source. The CPL and the UI are created by the maker of the experiment and at the start of the experiment session and copied into the UIM on the learner's device from the RLMS. The UI contains all the buttons, textboxes and display components for acquiring instruction from the users which are then passed on to the CI. The CI in the UIM operates on the CPL and the Symbol Table to determine corresponding commands for given events which is then passed onto the IEM.

At the maker's side, the IEM contains the CU and associated structures like look-up tables and translation modules to parse incoming commands and outgoing messages. It contains the instrument drivers that actually runs the instruments attached to the CU. The CPL is also created by the makers using the same UIM environment available to the learners. The makers design their experimental rig around the CU. There are several options for selecting CUs based on embedded control systems [151] such as FPGAs and micro-controller Units [119]. For the P2P RAL, the MCUs with

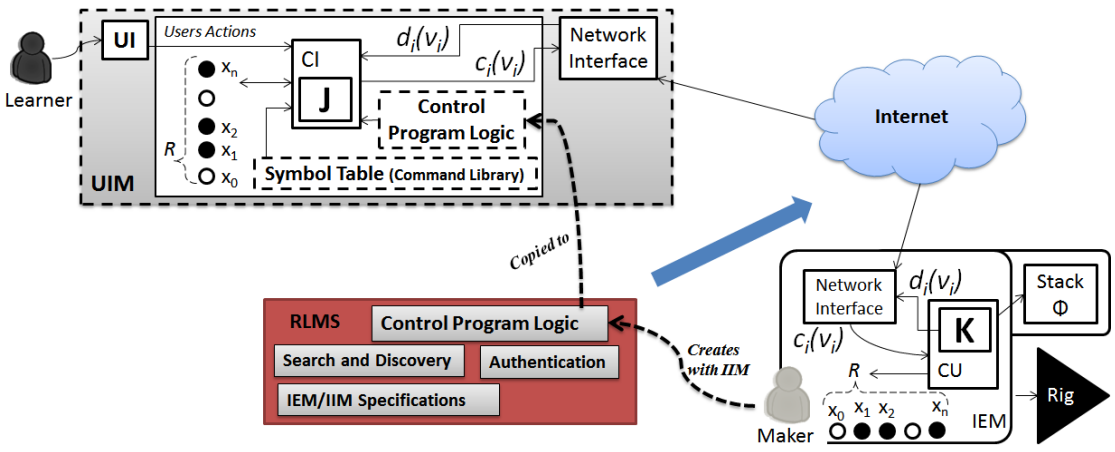


Figure. 5.2. The relation between the two FSAs in the P2P RAL system on the Internet and from Learner end to Maker end.

fixed hardware architectures and microprocessors are used for implementing the CU and associated structures. A number of MCUs are widely available and cost effective [119]. These low cost units have low memory but possess sufficient computational power to perform the tasks as IEM. MCU are discussed in Section 5.3 in more details.

For a given experiment, at any time one learner will be in control of one UIM that is connected to one IEM via the RLMS. The RLMS provides the search, authentication and storage facilities along with the UIM and IEM specifications that are loaded in the learner's devices or the makers MCU respectively.

For different MCUs the IEM has a common architecture. It reads the incoming inputs and processes them accordingly. The IEM has direct control over the peripheral devices and corresponding drivers. The choice of MCU may affect the capabilities of its IEM. An IEM consists of a Complex-Basic Command Translation (CBCT) component as well which consists of the Translation forward and Translation reverse modules to parse incoming commands to the lowest levels of commands. An IEM contains the following components, as shown in Figure 5.3:

- *Translation Forward Module* (T_f) - This will convert the incoming composite commands into a string of atomic instructions by searching through a pre-defined look-up table. This table is the library of commands available to the makers. The T_f will contain any string associated with the set command as well. Multiple atomic instructions of the same type (for example, two WRITEs) may be sent by the T_f, if the instructions are to be run in parallel i.e. one instruction to be executed before the completion of the other. For L_Y^0, or the simplest implementation, commands have only one atomic instruction which is directly passed to and from the CU.
- *Translation Reverse Module* (T_l) - This will convert the atomic outputs of CU i.e. success and failure messages into any complex response messages looking through another look-up table.
- The CU is as described in Section 4.6. The CU output messages, put in K, is parsed by T_l and sent to the CI.
- A *Translation Error Module* (T_e) - In the T_e, if an ℮ message is received, it is passed to the CI through T_l and the T_e tries to take the system to the last stable

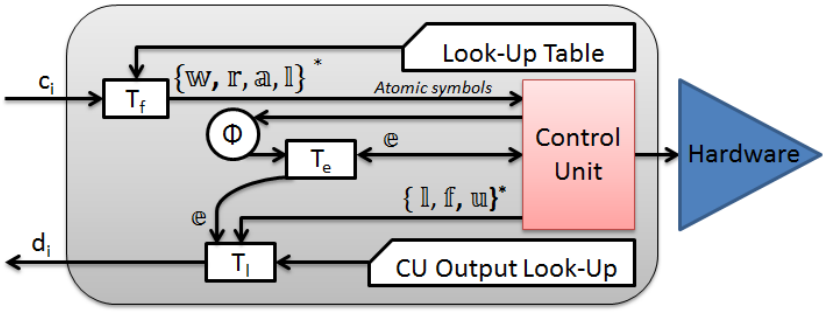


Figure. 5.3. IEM Implementation Architecture

state though inverse motion by retrieving instructions stored in ω. If an error cannot be resolved then the CU remains stuck in an invalid state until the CU is manually reset.

As, the size of the look-up tables has to be finite, only a finite set of composite commands may be supported by a MCU command library.

The IEM allows the users or the programmers who host the experiments to use any programming language to send commands to the MCU to control it. It is not required for them to write any code specific to a MCU. The actual rig setup consisting of the sensors and actuators, creating the CPL in the UIM and the corresponding user interface are all done by the makers of the experiment. A higher-level IEM implies it can support more complex and larger sized look-up tables compared to a lower level IEM.

The UIM uses WebSockets for communication with the experiments as they can traverse NATs and firewalls and can be implemented in all MCUs. They can provide equivalent performance to binary sockets. WebSockets run on most computing devices including portable devices and can be opened from inside web browsers. The UIM opens a WebSocket communication with a particular experiment IEM at the beginning of the experiment session.

For the RALfie implementations, the SNAP (http://snap.berkeley.edu/) programming interface is used as the UIM environment. The SNAP language is a visual graphical language with an exactly similar interface to SCARTCH [152, 153] from MIT that allows the makers to create programs with drag and drop off commands as 'blocks'. A block, as shown in Figure 5.4, is a specific function or operator in the SNAP program. The RALfie system uses several custom blocks especially developed for the MCUs

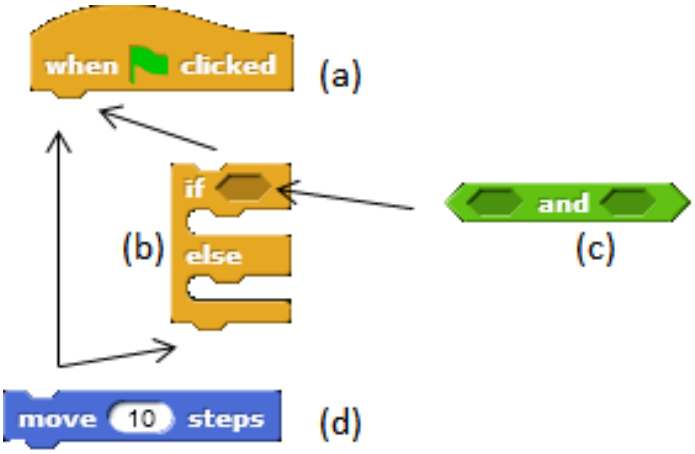


Figure 5.4. Some examples of SNAP blocks (a) a hat block to start a sequence of events by executing the block underneath it. (b) Condiotn Check (c) '*and'* Operator that fits into the '*if else*' and (d) a block that is used for animation of objects.

e.g. rotating motors and turning on ports with high and low voltages etc. The SNAP platform and further details of why this chosen language was chosen for STEM Education is discussed in Chapter 11. It may be noted that SNAP is only an example language platform used in RALfie. Any other platform may be used as long as it implements the corresponding libraries in similar manner.

5.3 Micro Controller Units Alternatives for IEM Implementation

Micro-controller units such as Arduino [154], Raspberry Pi (RP), BeagleBone Black (BBB), Lego Mindstorms EV3 are suitable to control the experiments remotely i.e.

Table 5.1. Comparing MCUs

Properties	Arduino(UNO, Due, Mega)	Raspberry Pi	BeagleBone Black	EV3 Mindstorms
Native Programming	Yes	No	Yes	Yes
Adaptive Programming	No	Yes	Yes	No
Pins	Analog/Digital	Digital Only	Analog/Digital	Custom (I2C)
Network speed	Good	Good	Good	Good (Wi-Fi Only)
Processing capability	Arm 7 (16- 90 MHz – Fair)	ARM 11 (700 MHz - Good)	ARM Cortex-A8 (1 GHz - Very Good)	ARM 9 (300 MHz -OK)
Visual capacity	No	Yes	Yes	No
Control Capacity	Medium - High	Medium	High	Medium (Custom parts)
Community Support	Very Good	Good	Fair	Very Good

they become the CU of an experiment. These MCUs have control ports that can be used to set and reset properties of a rig component like motors and servos. They can also collect various kinds of data from their surroundings through sensors. They also have network capabilities to connect to the Internet with TCP/IP based protocols. They are small, compact, cheap, readily available and the ideal CU for P2P RAL activities as well. Table 5.1 shows a comparison of MCUs with regards to their requirements as a control unit of a distributed P2P RAL system.

Achievable Throughput

Some experiments may require higher bandwidths for proper operation along with transmission of videos, which consumes high bandwidth. All of the MCUs are equipped with Ethernet connections but the maximum bandwidth supported by each of them varies depending upon the computational capacity. To establish real performance parameters an experiment was performed that involved transferring files over the network and the real throughput was measured. Several files of different sizes from 200KB to 95MB were transferred from a PC running Linux to the MCUs. The transport had to be adapted for the MCUs. For the BBB, Raspberry Pi and EV3 (with custom Java firmware) the SCP command was used from PC to transfer files. For an absolute bandwidth test for BBB and RP, the IPERF tool was used and it reported the maximum of 90-95 Mbps. The results of the test are shown in Figure 5.5. It is observed that for sending small amount of information (≤ 1MB), time taken is very low as the throughput decreases with larger files and transmission time. Both the RP and BBB can achieve download speeds of more than 2.5 MBps in a LAN. The

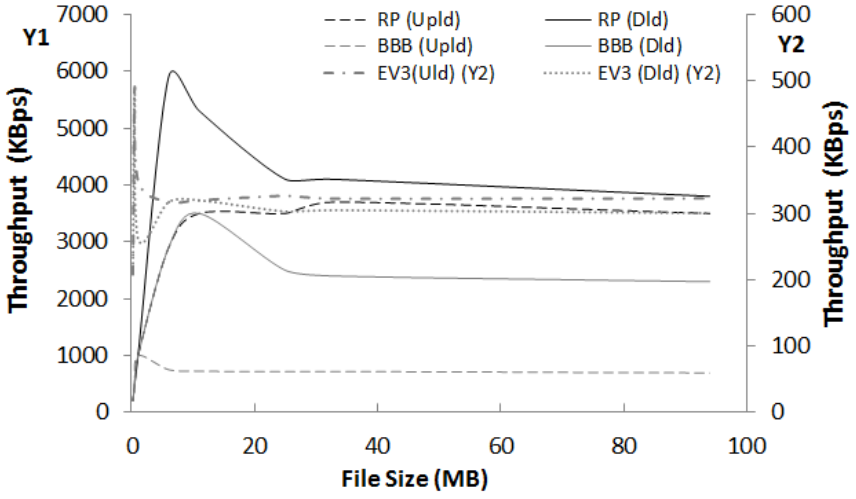


Figure 5.5. The throughput capacities of the MCUs

outgoing speeds of RP are on average 3.5 MBps and for BBB, 700KBps. The EV3 registered average speeds of 300 KBps for outgoing and 320 KBps for download. These values are sufficient for any RAL activities. For Arduino UNO, a web server was used to upload and download files. The speeds were below 10 KBps. This may vary a little with different Arduino boards and implementations, but due to limited computational capacity the speed will remain significantly smaller than the others.

To send video feedback, webcams are used that have in-built encoding mechanisms such as support for high resolution with hardware based H.264 encoding. This encoded stream is directly fed to the MCU which then transmits it over the network. The BBB transmits video at 900 Kbps with the 320x240 resolution. Other devices like IP cameras can be used along with MCUs that do not support video streaming.

A particular communication and control standard ensures consistent functioning, integrity and compatibility of the devices used for RAL. Unlike FPGAs, the MCUs proposed to be used as part of this architecture do not support any uniform standard for communication and/or control such as IEEE1451.x [155] or LXI [84]. Moreover, these protocols do not allow implementing a flexible programming logic required for creating and running variable rig designs. Thus an alternative protocol is introduced in Section 5.4. It has been implemented to investigate the networking and control characteristics of the distributed architecture. This lightweight protocol covers a basic set of commands that are used to control the rig and may be extended as a standard for the P2P RAL in the future.

5.4 Messaging Protocol

The distributed systems consists of three *entities* and each of these *entities* features a number of components - the *experiment units* implements the IEM, messaging protocol flow control and queuing methods along with the UIM, the *RLMS unit* deals with authentication and scheduling in a relay server and the *user units* has the UIM.

5.4.1 Protocol Messages

To exchange information about control, a set of messages are defined that are issued by and interpreted at the nodes. Such messages are unidirectional i.e. an experiment end-node can issue an ACK or EVENT message but not INSTRUCTION message while an users node can only send INSTRUCTION messages. The messaging

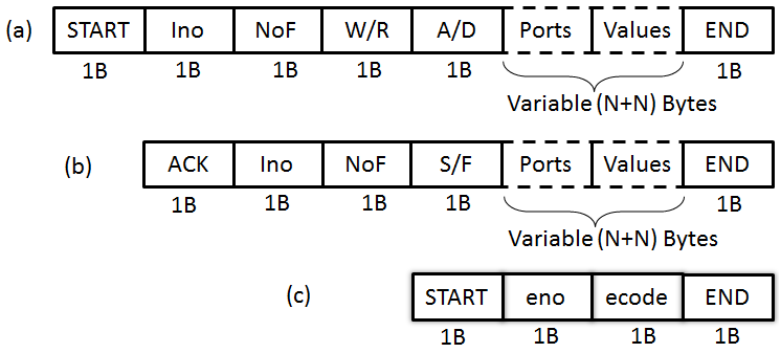


Figure 5.6 (a) Instruction message from CI to CU (b) Acknowledgement message from CU to CI (c) Error message from CU to CI

protocol in this study implements the most basic requirements to control a remote experiment. The main aim in designing the messages was to keep them small.

Network protocols for controlling robots can operate in either object-oriented manner by associating specific functions with the devices or event-oriented manner. For example, the software architecture of SNRP (Simple Network Robot Protocol) as described in [156] is an object-oriented approach. In case of RALs, functional and event-oriented programming is used where the users' action generates messages that represent the event, i.e. a read operation to get the status of the rig or a write operation to change of state of the rig. Instruction messages are executed on the experiment node.

Three messages are defined and their structure is depicted in Figure 5.6: instructions from the UIM to the IEM, acknowledgements from the IEM to the UIM and error messages from the IEM to the UIM.

Two message types based on the most basic operations, *READ and WRITE* are being discussed. In essence these messages read port values or set port values. The UIM sends a series of instructions and acknowledgement messages to communicate between the MCU and the client. The Ino field is the instruction number for the identification of this message. The W/R field can specify whether it is write or a read message. Alternatively this field may be replaced with a number 0-255 for a much larger set of composite commands ignoring the ports and value fields completely.

These messages may specify a variable number of fields by specifying multiple pin numbers in one message. This may be done by introducing a NoF (Number of Fields) block (to specify number of values) and a variable length message (see Figure 5.6).

All of these messages can be built with a few bytes (7-8 bytes for a single message representing a single event) making the information sent very small. The protocol may be extended to include more complex elements to support the different types of ports available on MCUs such as I2C and CAN ports. These can be used directly to control the external peripherals. The START field can include information about the final destination of the message, which can be used for routing it through the overlay network.

For further control an EVENT message is used that originates from the MCU. The Event message may send any information regarding an event that has occurred at the experiment side. The user side may not have asked for this information, for example, when an instruction is not received, the battery power is getting very low or a port suddenly stops operating due to structural failure. The EVENT message starts with a static START block followed by event number (for the client to keep track of events), *ecode* stating a predefined numerical value for the event and finally the END block.

The *queuing* and *flow control* mechanisms are most effective for when using lower levels of commands. The following sections assume that the experiment is using the lowest levels i.e. atomic commands only.

5.4.2 Flow Control of Messages

The instructions sent to control the rig must be executed in the exact order and time interval, hence each message is numbered in a session between users and the experiment to maintain synchronization. However, it is observed that due to the low computational capacities, the MCUs can lose messages i.e. not process every incoming message and skip to the next one, if the messages are sent more frequently. Even if the messages are delivered at the network level, they are dropped by the MCU.

Messages originate and are sent in a particular order. An experiment session can have a set of commands $C = \{c_1, c_2 \ldots\}$ dispatched at intervals $\hat{J} = \{g_1, g_2 \ldots\}$ where $g_l < g_i < g_u$ such that the total time of the entire session $T = \Sigma g_i$. g_l and g_u are the lower and the upper bounds of the intervals. These values are dependent on the nature of the experiment design. For any rig, the commands c_i changes the rig position from one state to another where the change is always deterministic. This is done in unit

operations according to c_i, for example, a servo will always be issued commands with moving up to a certain degree which guarantees that the behaviour of the servo is entirely depended on the input given to it.

If the MCU detects that a message is missing, then an EVENT message is sent back requesting the missing message. But if the client has to wait for each acknowledgment before sending the next instruction, then it could cause significant delays. Assuming that, in general, the underlying network will be reliable, i.e. most messages will be delivered correctly, all messages may be sent directly without processing the acknowledgement in in the CI but as a service in the UIM. This service keeps track of all sent and acknowledged messages. The UIM is *paused* if the significant number of instructions is sent without any acknowledgment. If some request is not received at the MCU then the MCU sends an EVENT message may be sent to the UIM or the service resends the messages again after a certain period of time. This method in general follows the 'Go-back N' protocol. The service also keeps record of the time at which each message was dispatched and when resending them maintaining the exact time intervals.

The flow control is more useful in the Direct Access Control where the number of messages is high as the actual logic or origin of instructions is on the users' nodes resulting in higher message loss compared to the undirected mode.

5.4.3 Message Queuing

Since the MCUs tend to induce delays in the processing of request, it is desirable to send as many commands in one instruction message as possible. However, the message from the UI may occur at random time thus simply waiting for a specific number of commands is not feasible. Thus a queuing methodology is required to optimize the waiting time and the number of the commands to be sent in an instruction.

A simple time-stamp method is used in this study. Every action in the UI generates one or more new messages to be sent over the network. Due to temporal locality, it is expected that during state change on the rig, a number of independent instructions will be executed simultaneously to create the action. These instructions originate with a very small negligible time gap between them. As a new message is created it is

associated with a timestamp immediately. Depending upon the nature of the experiment, a message may be delayed only by a certain amount of time (t). However, this value is extremely small in most experiments.

If any message is created within this time frame, it is joined with the earlier message to create a new combined message. This message retains the timestamp of the first component message. Messages are combined by putting the new port and value combinations into the earlier message and increasing the NoF field. The combined message is then dispatched as soon as the delay reaches the value t, a small value (<10ms) that does not alter the time intervals of the inputs. This way a lot of messages can be accumulated together and the actual number of transmissions can be reduced maintaining the order of instructions. The size of each message varies with the frequency in which the messages are generated. A message may also be dispatched if its size becomes equal to a maximum size allowed. However, since the individual message sizes is 8 bytes, combining them will still not create a large sized message as the time gap t is small and the number of instructions generated during the interval will be low for all practical purposes. This should help in situations where multiple events occur simultaneously and the instructions for each event can be combined together.

With queuing, none of the interval is increases by more than t and g_i has no impact on g_{i+1}. The message creation times and departure times are independent of following messages. Hence the entire session time T does not increase by more than the value of t which is negligible. Considering that there is y % intervals in J where $g_i \leq t$, the entire command set may be reduced by y % at most in the best case if all such messages appear after $g_j > t$. In the worst case, if there is a single sequence of all $g_i = t$, then the number of messages can be reduced by $y/2$ per cent.

A similar approach has been used in previous work to reduce the transmission load by withholding information from the nodes if the previously transmitted data are within a tolerable range [40-41]. These approaches however considered closed-loop control systems, but the proposed distributed RAL is not closed loop, as it does not operate directly on any feedback from the rigs. The rig may gather data from its sensors which is sent to the UI and can influence the next decisions, but a human user actually gives the inputs to the rig. An alternate way to deal with lossy or reduced network traffic

will be to use a predictive system that can estimate the missing instructions according to the expected behaviour of the rig. The MBPNCS (Model Based Predictive Networked Control Systems) is a NCS mechanism that can overcome adverse conditions of data loss and delay in a network by predicting the future control behaviours based on a model of the end user control system [42]. This is however difficult to implement in the current context as the rigs are designed without following any pattern by different users who will require additional tools for creating a model.

5.5 Relay and Remote Laboratory Management System Server

All users will be interacting through the Internet. Hence, the network between the user and the experiment may be heterogeneous with several layers of firewalls, network address translators and proxies. To overcome these limitations in connectivity, relay(s) or server(s) may be used to relay the messages between the user interface and the experiment program. The relay server could be placed as part of the RLMS or part of a broader network of makers and users nodes.

In a small network with low latency (such as within a city), a central relay server (possibly the RLMS server itself) can be used as a relay node. Messages are just passed without any modification.

The proposed RAL system is of P2P architecture. Thus multiple users will be accessing the system and some of these can act as relay servers depending upon their networking and processing capabilities. This architecture may be similar to Skype, JXTA protocol or overlay network systems [157] covering a large region with varying latency.

Regardless of the relay mechanism a central module as the RLMS is required. This includes the management of MCUs, like assigning unique global identification numbers to the MCUs and associating them with the user accounts. This should also maintain a scheduling scheme for deciding which client can access the experiment and when and authentication components for verifying that all messages being transferred in a session originates from a valid source and transmitted to a valid destination.

The network environment of P2P RAL is further discussed in Chapter 9.

5.6 An Implementation – Results and Analysis

This section discusses the setup of experiments, their operation, network characteristics with respect to *flow control* and *queuing* assuming that the experiment is using the lowest levels i.e. *atomic* commands only.

5.6.1 Test-bed Configuration

For the test setup the UIM and IEM described in Section 5.2 are used with minor alterations to collect specific data at the UIM. Figure 5.7 shows the basic network architecture of the system with the users' side and experiment side being connected by a cloud based relay server.

The maker side consists of a MCU and the actual rig with sensors and actuators. When plugged in to the network with Ethernet, the MCU immediately registers itself with the RLMS, which is a centralised server machine (for these tests) also acting as the relay server and opens a WebSocket link to it. The user side consists of a PC that can run the UIM which upon initiation also connects to the RLMS in a similar manner. In reality the relay server may be replaced by an overlay network on the Internet. The network also consists of a server machine with Wanem 3.0 [158] running on it. The Wanem is a Linux based network emulator to create a simulated network environment with different round trip time between the devices and also other network properties such as bandwidth. The IEM to execute incoming instructions was written in their native languages for Arduino UNO (C++) and BeagleBone Black (boneScript based on NodeJS).

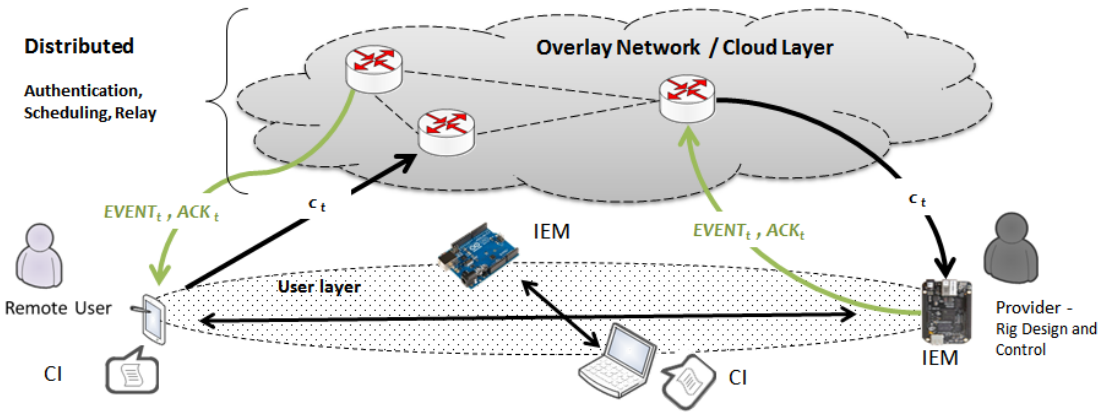


Figure. 5.7. The distributed network architecture consisting of the user sites and the experiment sites

5.6.2 Latency Measurement with WebSockets

To test the latency induced in the communication process, the UIM was used to measure the time between the dispatching a message and its acknowledgment. Figure 5.8 shows the percentage of increase in round trip time (RTT) and the message drop rate for the random sequence of messages (with 95% confidence). A total of 8999 random integer values were generated in the IEM to represent the delay between 9000 instruction messages (g_i). The value for g_i was kept between (g_l, g_u) 90-100ms, 190-200ms, 290-300ms, 390-400ms and 490-500ms with randomly introducing values between less than 10ms. The final command sequence contained 38.9% instructions generated within 10ms of the last one. This represents a typical scenario of a rig with multiple or quick controls. An average RTT is recorded with and without the queuing mechanism. The architecture follows the direct access control with the BBB. The delay is caused by the propagation delay and the processing delay at the MCU. The time taken for actual read write operation on Arduino and BeagleBone is in the order of 1-10 µs which is negligible.

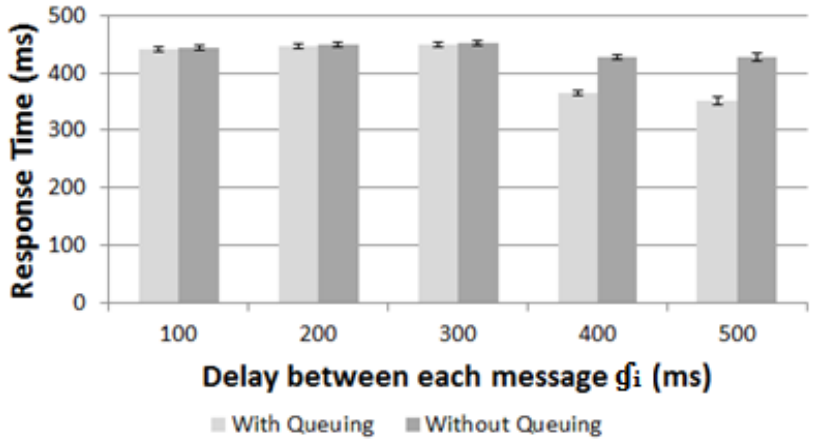


Figure 5.8. Queuing reduces traffic and response time

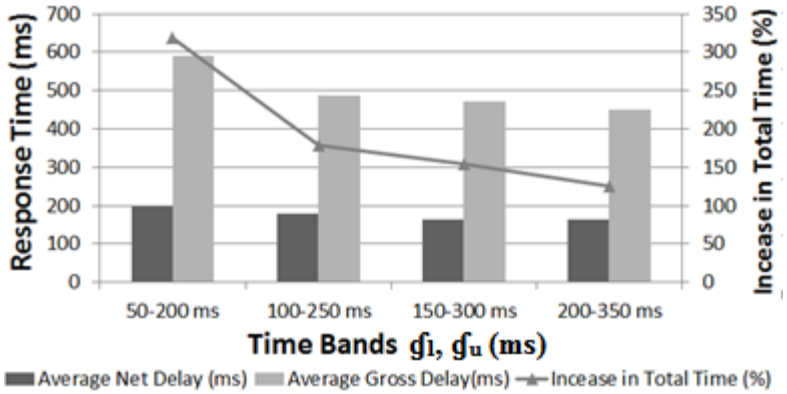


Figure 5.9. Flow control increases the session time.

The BeagleBone is the most suitable platform for the purpose of the P2P RAL architecture based on the high processing, networking and control capabilities and thus primarily used in this study. The BBB was able to reply to all messages in both cases. With queuing the total number of messages was reduced to 67.32% of the initial 9000 messages. The average RTT are similar for g_u < 400 ms beyond which there is a sharp fall of 74.07 ms for g_u = 500ms in the average RTT. This is due to the fact that with queuing, there are fewer messages been sent on the network. Indirectly, this also reduces the probability of losing and resending messages on the Internet. The overall session was not delay beyond 10ms.

To test the effects of flow control, a random sequence of 999 integer values representing the time difference g_i was generated and a total of 1000 messages were sent with each message was sent after waiting for each g_i with (g_l, g_u) = 50-200ms, 100-250ms, 150-300ms and 200-350ms. RTT between the user node and the Arduino is set at 300ms and the window size was set to 5. Figure 5.9 shows the result of this test. Without any flow control, 30% of the messages were lost or missed by the Arduino.

The flow control algorithm with *go back n* was able to send all messages through to the Arduino UNO. However the resultant time taken to send a sequence of 1000 messages was 318% of the total sum of all intervals (T) i.e. the actual time that should have been taken between the first and the last message. With the flow control 77.5% of messages were missed at least once by the Arduino UNO i.e. they had to be resent at least once. The average net delay of a message i.e., the time between it is sent the first time and it's acknowledgment received, is 590ms with a standard deviation of 255ms and the average gross delay i.e. time between it is sent the last time sent and acknowledgment received is 196ms with standard deviation of 80ms. Changing the delay bounds (g_l, g_u) from between 50-200ms to 200-350ms reduces the average net delay to 450ms, the average gross delay to 162ms and the overall increase in total time consumed *(T)* to 125%, without affecting percentage of messages that were missed at least once.

None of these values are absolute for an MCU as these will differ with experiment and network environment. Moreover, the effects of flow control and queuing with high latency and low processing powers also diminish with the use of composite

commands. For a given experiment with consistent characteristics, such as $g_{i,}$, key observations include the following.

- Queuing is able to reduce the volume of messages sent and thus reduce the delay per message in most cases. Thus a queuing mechanism improves the performance of an experiment session.
- Applying flow control is able to send all messages through, but it can considerably increase the total time consumed in a session for a slow MCU or low values of g_i. Thus a flow control mechanism may not always improve the performance and should not be heavily relied upon.
- Increasing the frequency of messages increases the delay per acknowledgment of messages.

This shows that all MCUs are not suitable for all experiments due to difference in capacity. The distributed RAL system must identify the type of the experiment node and follow corresponding flow control and queuing control methods. It is however the maker who must decide the suitability of an MCU for an experiment.

Conclusions

The generic model can be used to create a remote laboratory environment that is highly scalable with participants being able add their own rigs and share them with peers. A potential key factor to real time control by users is the flow control and queuing of instructions and their effects were studied here. The communication protocol discussed here covers only the basic elements. Advanced queuing methods and flow control may be used by creating tools for producing a model for individual experiment rigs.

Although users do not have to manage the underlying messaging structures and data exchange, there are two main issues that users need to be aware of whilst creating their programs: *port mapping* and *delays*.

Ports are implemented and used in the program as variables that can then be used to create the logic that drives the rig. This requires that users understand the linkage between the physical connections and the software variables.

Despite implementing queuing and flow control, users may need to arrange their

program logic so that a forced *delay* is induced between command instructions for the experiment. As such, the user must be able to produce an acceptable program outcome when creating time sensitive program implementations. Time critical programming however would not be typically required for basic data acquisition situations of standard rigs.

This generic model provides the basic IEM architecture. The microcontrollers are the key components of this framework and it has been shown that they are capable of delivering acceptable performance with support for adequate bandwidth and latency relative to the experiment. With respect to the P2P RAL, the generic model is capable of reducing the makers' efforts to create and host experiments to a great extent. This chapters shows that common objects available commercially can be turned into smart devices to be able to communicate over the network and Internet. While the application of the proposed models has been specifically about P2P RAL, these devices implementing the proposed models and methodologies can also be applied to other distributed control scenarios, in particular in the context of the IoT.

The following chapters discuss several intelligent tools that are based on this CI-CU model. These intelligent tools can support the makers to create their experiments and maintain reliability and stability of the equipment.

6

Intelligent Tools: Support and Validation and Evaluation

> *This chapter presents the homogenous set of tools that can be part of the P2P RAL RLMS based on the automaton model.*

In P2P RAL individual users are expected to create experiments and share them with others. This requires the users work with a flexible, yet controlled environment both in terms of hardware and software. MCUs as hardware platform can provide great flexibility as shown in the previous chapter. For the software or programming environment, the P2P RLMS must be able to view each experiment as a generalized model to provide supporting tools to create a universal UI and control program logic for the experiments. The automaton based model and its commands discussed in the previous chapter provides the generalized model of the experiments.

This chapter introduces the methods of using this model to implement features that improve users experience and operational reliability. This includes functions to validate user commands to ensure rig safety and determine whether user support is required. This also includes evaluating the user interactions with the experiment. A universal set of supporting tools is required to address these functions to enable wide-scale sharing and collaboration.

The extensions of the automaton model through Markov Decision Processes (MDP) are an original contribution of this dissertation. The MDP-based model of the experiments can be used to determine the correct course of the experiment run. MDPs are created based on the recorded makers and subsequently user interactions creating the unique states on the CU ports. MDP can be constructed from the state space of the rigs which provides a set of experiment state transitions that are valid and permissible.

This way the MDP can be used by the experiment controller to ensure stability as well as support other users by evaluating the current state of the rig in their experimental session as described in Section 6.4.These contributions have been published in [159]. This process use training data initially collected from makers interacting with the experiment before it is published for public use.

This chapter is divided into two major sub sections. Section 6.1 discusses Markov decision processes in the context of experiment control in more detail and Section 6.2 introduces tools that used MDP to support makers and users.

6.1 Markov Decision Process

One major issue is the evaluation of the students' performance and providing guidance should the student require it during use of the experiment. While in a centralised version, a specific interface for each experiment can be built, in a P2P context a generalized set of tools is required. Evaluation and guidance can be based on the same data structure. Evaluation requires verifying whether users have gone through a set of states in the rig, possibly in a specific order. Guidance or support [159] is the process of providing hints and feedback on what the next state should be given a current state. Both the process of evaluation and guidance require the concept of a finite set of definite states.

As described in Chapter 4, the CU can only have a finite set of ports *R*. The state of each port can define the state of the rig. Note that this state is different from the state of the CU working model described in Section 4.6. These states are the intermediate 'physical' states of the ports on the CU regardless of the current state of its operation. The state of the experimental rig ports can be organized into a MDP [159] to determine the best course of action (or command sequence) that will change the experiments port states in the most desirable way with regard to learning outcomes.

6.1.1 Rig State Space

The state space of an experimental rig is dependent on the status of the ports. Each port signifies a variable in the experimental rig whether it is connected to a sensor or actuator. Changes in the state space are caused by changes in the value of any port as in shown in Equation (4.2).

In terms of simple control, only actuators have any impact on state transition as these are the only components that can directly change the orientation of the experimental rig. But, from a decision and automation point of view, the state space contains the values of all active ports on the MCU both sensors and actuators. The state space of the rig is infinite as each actuator in the rigs can have a value between -∞ and ∞. But in reality while training and running the rigs, it will only attain a finite set of states.

The rigs and its controller can be represented as twin-finite state automata. This architecture consists of two sides, CI and the CU. The CU acts upon the inputs from *CI* and the human users. The language between *CI* and *CU* is the communication protocol for the instrument control in the P2P RAL. This language consists of the very basic (or atomic) components of instrumentation - read (𝕣) for reading the value of a port (sensors and actuators), write (𝕨) for writing a value to a port (for actuators) and wait (𝕒) to pause the CU for maintain synchronization. The aim is to convert the finite state space into a MDP.

6.1.2 Related Work – Markov Decision Processes and Control

MDPs are models to represent stochastic processes and have been applied in many fields to model partly random decision processes. The Stochastic Shortest Path MDP [160] or SSP MDP is a particular version of the MDP that specifies a set of goal states that must be reached from any other state. For a system modelled by MDPs, there is a decision maker or agent which decides what to do in the system. The agent is provided with a plan or policy that gives the best chances with minimal cost or delays to succeed in reaching certain goal states. From an MDP perspective, in a P2P RAL experiment while the rig is being used, the human learner's input is a random factor and they act as the *agents*.

MDPs are used to model systems that maintain the memoryless properties i.e. choosing a new system state solely based on the current state and the corresponding action. However, there are some approaches that store the past information into the current state of the system and carry it forward [161]. The next best state and the corresponding action are chosen based on a prescribed policy (π) that maps each state to one action. Again the policy may not be static for every time step of the system [162] and it may be updated with variable rewards within the system. MDPs are used in artificial intelligence [163] to model and create decision-based support systems. It

provides a framework to model complex problems that have large state-spaces and complex cost functions. It also provides a model to further develop learning algorithms to aid reinforcement learning corresponding to the system. There are also some well-known and efficient methods to solve MDPs such as the Value Iteration Algorithm (VIA) [164].

In [165] the MDP is used to model the aircraft's movements and autonomously avoid collision. The performances of different types of sensors are evaluated in the model. But the aircraft state vector is fixed to the properties of an aircraft, thus limited to a specific application. In the RAL scenario, MCUs provide a generic platform to create variable experiments. Thus a method is required that can generate a MDP for any rig using the MCU based architecture [119]. Similarly another particular application to guide people with dementia is reported in [166]. It exploits the MDP's implicit capability to manage stochastic dynamics and capturing the trade-offs between multiple objectives. All of these applications are capable of handling high dimensional data and large number of states, although the computational capacity required becomes a challenge on low-cost devices like MCUs.

The next section proposes a *new* MDP based on the experimental rigs and uses it to implement intelligent tools for the CU.

6.1.3 States in the MDP

The experiment rigs in P2P RAL have to be intelligent enough to create the control policies and avoid erroneous rig states. Thus creating the MDP for the rigs can help in two ways:

First application is in setting up the admissible boundary of the state space of the experimental rig. It can then always keep the rig in a *valid* state that can be obtained from the MDP. There can be two broad types of states:

- *Valid* state: a possible rig state that is stable and in the MDP, thus permissible. It is when the rig is not executing any command and the rig's parameters are not changing.
- *Error* state: a possible rig state that will break the rig and make it inactive, thus not permissible.

The valid states can be determined in the MDP. But *error* states are not possible to be identified as they are never recorded in the MDP. There can be two other types of states:

- *Interpolated* state: a possible rig state that is not in the MDP, but may be valid. Whether the state is *valid* or not may be determined by interpolating nearby *valid* state.
- *Undesired* states: A possible state that is not in the MDP, but cannot be validated in any way i.e. it may not break the rig, but not permissible either. An *undesired* state is essentially assumed to be an *error* state by the rig.

The second application MDPs are in determining the next best step towards the immediate goals keeping in view any other goals depending solely upon the current state of the rig. The learners' actions are matched against these best moves to *evaluate* their interaction and determine if any support is required for them.

6.1.4 The Experimental Rigs as MDPs

The MDP is created from the state space of the experimental rig. An experimental rig may consist of multiple sensors (or actuators with feedback mechanisms) each of which is considered a variable in the state space. For actuators, it contains sensors to determine their current state. The conversion of the experimental rig's state space into MDP maintains a direct relationship between the MDP states and the rig's state space, i.e. rig's state that are positioned in the MDP adjacent to other states that precedes or succeeds them during the course of rig control. This can help in evaluating whether the transition in experimental rig state is desirable.

Construction of the MDP requires a training data set containing sample input commands. The training set *X* has the set of makers' inputs during testing phase of the experiment,

$$X = \{ x_n^0,\ x_n^2,\ \ x_n^3\ \ \dots\ x_n^{\varrho} \} \qquad \dots (6.1)$$

where x^t_n is the state vector at time *t* of the experimental rig, *n* is the dimension of the feature vector i.e. the number of variables (or sensors) in the rig and ϱ is a finite integer. The feature vector contains the values of all ports connected to a component, both sensors and actuators. Each x_n^t is a stable rig state when a command has finished

executing and before the beginning of the next command execution. A command can be *composite* i.e. contain multiple instance of $\mathbb{r}, \mathbb{w}, \mathbb{a}$ to accomplish an action that requires to either maintain strict time intervals between multiple states transition from x_n^t to x_n^{t+1} (e.g. $\mathbb{w}_1 \mathbb{a} \mathbb{w}_2$) or even change multiple ports in parallel in a single $\mathbb{w}$ command. However, each *composite* command can be broken down to its corresponding set of atomic instructions and executed in reverse order (e.g. $\mathbb{w}_2 \mathbb{a} \mathbb{w}_1$) to get back to the previous state x^t_n.

Thus, the MDP for the rig system is defined as

$$Y = \{\mathit{III}, A, T, G, \alpha_1, \alpha_2 \ldots \alpha_{|G|}\} \quad \ldots(6.2)$$

where, $\mathit{III} = E \cup F$. E contains all the valid states the rig can be in. F is a set of failed states corresponding to each transition $t \in T$ for valid states. E represents a small subset of all possible states of the rig as most others would be error or undesired. Each element in E corresponds to one or more elements in X.

$A = \{ w(P, V)\}$ are the write commands that are issued by the user. This is the random factor in the MDP as the agent may choose to issue any command regardless of whether that is optimal or not. P is a set of port(s) the command works on and V are the values to be written.

T is a set of transition rules (or edges) that defines action allowed from a state, that would lead it to the next state(s) if the associated command is executed. $t \in T$ also contains the probability of success of the transition i.e. $0 \leq T(s, s') \leq 1$.

G is the set of goal states and $G \subset \mathit{III}$.

α_i are rewards strategies corresponding to each goal state $i \in G$. Each reward strategy consists of a matrix of rewards for each transition $t \in T$.

The Figure 6.1 shows an example of the MDP where A, B, C, D, E, G are valid states, D is a goal state and F_{AB}, F_{AD}, F_{BC} ... are failed states corresponding to the actions of a valid states. The characteristics of the MDP graph are:

- There can be no self-loop in the MDP graph i.e. there can be no action that will keep or bring the rig to the same state.

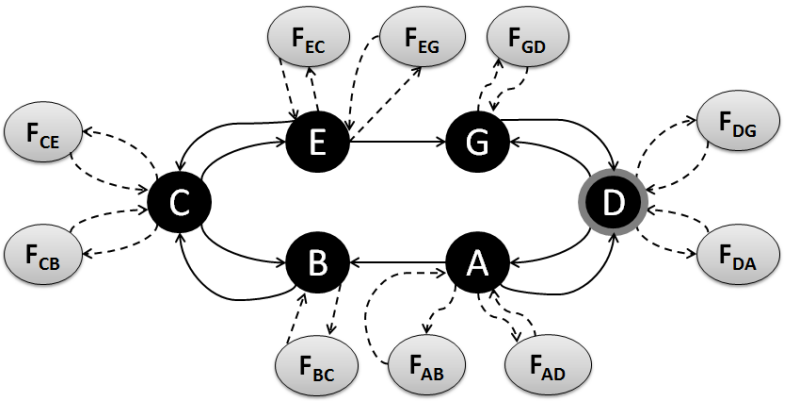


Figure. 6.1. Example of an experiment MDP graph.

- There can be many numbers of actions (or commands) in the MDP corresponding to the components on the CI. But any *action* from a state *s* can lead to only one other valid state *s'*. So, the actions can be represented simply by the corresponding states (s_i, s_j) it is between i.e. the *edges* between *valid* states represents a command. For two states s and s', if there is only one edge (s, s') during training with a command (e.g. $w_1 a w_2$), the edge (s', s) may be created by reversing the command (e.g. $w_2 a w_1$), if that is permitted.

- A path should exist from every node to another node i.e. it cannot have an absorbing state or locking states. This means, that the rig cannot stall at any position with that learner having no control over it to bring it another state. The existences of the routes are vital as the rig may have to automatically restore itself to certain states from any other state by executing the commands or action associated with each edge in reverse. There may be two pairs of states that have only one directed edge, but these should allow traversal from one side of the graph to another. A single edge between two nodes represents a one-way transition. This may be due mechanical constructions such as valves that operate in one-way. But being a semi-autonomous system, the rig must be able to reach to a state preceding the single edge transition i.e. there must be an alternate path.
- For every pair of adjacent state (s, s') in the rig, there is a failed state $f_{ss'}$ for it connected to only s. The *failed* state represents the situation when a command fails and the rig enters a state that is basically an undesired state. It represents only the failure of its connected valid state *s* to reach s'. Thus, the aim of the

rig in the fail state is to move to a valid state automatically which is always the corresponding connected *valid* state. The probability of success from each failed state to the success state is always considered 1. If it is unable to restore itself to its *valid* state then the rig is considered broken. There can be no specific command associated with the edges on $f_{ss'}$.

- There are nodes that represent the goal states. Goal state may be determined by a number of ways - the node with the highest degree or the most visited node during training. But the best way is to collect the goal states from the makers explicitly. These states signify the achievement of a target i.e. learning outcome in the experiment. Goal states or task can be a single state in the MDP or there can be multiple states in which case the MDP includes multiple rewards strategies (α_i). This will generate multiple policies for each reward strategy. Each reward strategy α_i allocates the maximum *utility* to the goal state *i*. Correspondingly, a policy exists for each goal state *i* as = π_1, π_2, ... π_g. each reward strategy α_i or π_i gives most importance to the goal state $i \in G$.

Another important aspect of the MDP is its quality as MDP is trained by its maker. The makers are not expected to cover all possible states and transitions during training. For this purpose, a number of edges may be added by the rig itself depending upon conditions set by the maker. The quality of a MDP after training then can be defined as:

$$Quality = \frac{number\ of\ edges\ from\ Training\ Data}{number\ of\ edges\ added\ by\ the\ rig} \quad \ldots (6.3)$$

Quality can indicate the competence of the makers, for example. It provides a measure of how well a rig is built which may affect the maker evaluation in a learning context.

6.1.5 The MDP Generating Algorithm

The MDP is generated using the following proposed new algorithm from the training data that explicitly contains the goal states. Makers can use the following steps to generate the MDP:

Algorithm 6.1

1) *When the experiment is run by the maker for training, record each command and the corresponding state (x^t_n) that is reached when it finishes i.e. the values of the ports along with the goal states. This recording provides the training data set.*

2) *For each state transition, s_1 to s_2 recorded, add the states and action as the directed edge (s_1, s_2) in the MDP.*

3) *Check whether there is any node that is not reachable from any other node. If there is any such node, check whether all the transitions can be bi-directional.*

    (i) *If yes, for each pair of state that has one directed edge, add another edge in the opposite direction with the reversed command. The number of such addition is noted to calculate the quality of the training set.*

    (ii) *If no, then the training data is insufficient and more data is need. The algorithm stops here.*

4) *For each edge (s, s') set probability T(s, s') = 0.99.*

5) *For each edge (s, s'), add a fail state $f_{ss'}$ with edge T(s, $f_{ss'}$) = 0.01 and T($f_{ss'}$, s) = 1.*

When an MDP is created for the first time, Step 4-5 assigns static values for the probabilities of transition. In subsequent training sessions, the success rate of any existing edge (s, s') may be recorded and the values for probabilities in T(s, s') and T(s, $f_{ss'}$) can be updated.

6) *Calculate the degree deg(s) of each state $s \in$ Ш. For each reward strategy, α_i*

    (i) *Initially assign for each edge (s, s') in the MDP a reward value α_i(s, s') = deg(s')^2.*

    (ii) *Then the reward for goal state i, corresponding to α_i, α_i(s, i) = 2 × max (deg(s')^2).*

    (iii) *For all fail states f_s from state s, α_i(s, f_s) = 0.*

7) *Once the MDP is constructed, the Value Iteration Algorithm (VIA) is used to determine the best policies (π_i) for the MDP.*

The VIA() starts with the value function V_0^r (s) = 0 for all state $s \in$ Ш. Then the following is repeated until for all $s \in$ Ш, $V^r_{(i+1)}$ (s) - V^r_i (s) $\leq$ 0.001 i.e. it converges.

$$V_{i+1}^{r}(s) = \max_{s' \in E} T(s,s')[\alpha(s,s') + \vartheta \cdot V_i^r(s')]\} \quad \ldots (6.4)$$

for i = 1, 2, ..., where, ϑ is the decay rate. In each iteration for each state s, the policy (π_r) records the state s' as the best next state for which $V^r_{i+1}(s)$ is the highest. The VIA itself is repeated for all rewards strategies α_i to generate corresponding policy plan for each goal state. The algorithm generates the MDP and a set of optimal policy plans corresponding to each goal state.

Analysis of the MDP generating tool

Step 1-5 creates the MDP according to the properties discussed earlier and adds the fail states. Step 4 makes sure that the probability of transition is never 1 for transitions between valid states. Even if no failure is recorded, there is always a chance of failure. Step 6 calculates the reward matrix (α_i) for each goal state in G. It ensures that each of the edges leading to the goal state has the highest reward value. Step 7 calculates the best reward possible i.e. value function for each α_i. Note there is no summation as in the regular VIA [162] because every successful command or action leads to only one *valid* state and *fail* states are not counted.

Different MCUs have different computational capacities and power resources. The number of states generated in the MDP may be very large and edges between each state even larger. Processing the VIA to calculate the best policy map takes the largest computational effort. This algorithm has a decay rate (ϑ) that may be altered to increase or decrease the speed of the algorithm.

Figure. 6.2 shows the effect of changing ϑ from 0.99 to 0.5. While the iterations reduce exponentially, the quality of the policy will drop at certain point (in this case ϑ = 0.65). Thus, a suitable value for ϑ may be determined for a particular rig for a given

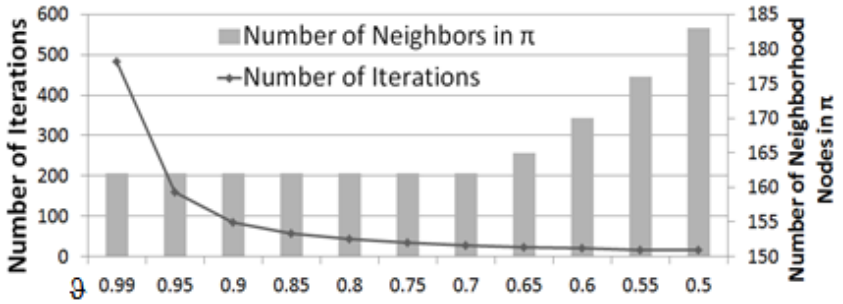


Figure. 6.2. Relationship between decay factor and the accuracy of the policy.

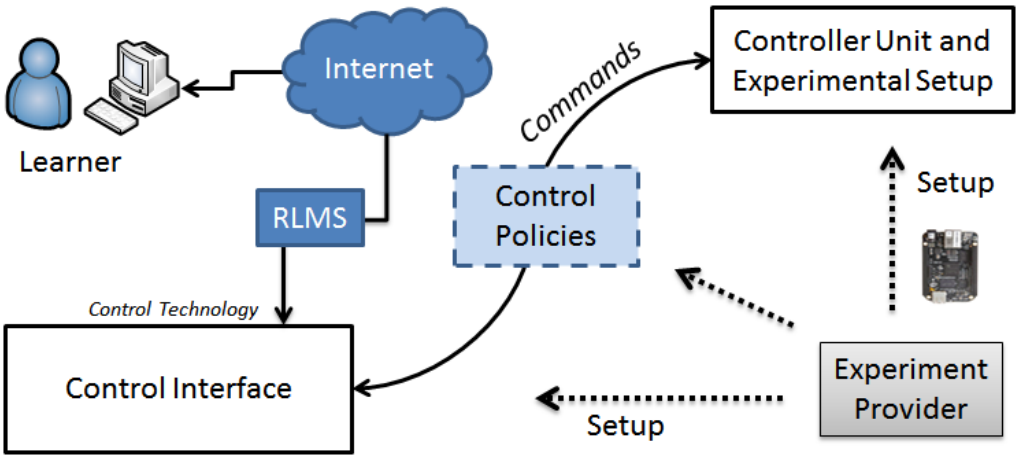


Figure. 6.3. The system architecture of a RAL experiment showing the control policies.

training data set. This will change respective to MCUs and the experiments built around them and useful when the policy needs to be recalculated quickly such as during an experiment session.

The MDP is used to address the issues of enhancing user experience by changing the level of commands to be used and validation and support of users experience in the next sections.

6.2 Supporting Tools for Makers and Users

As mentioned before, the P2P RLMS must provide universal tools to support both *makers* and *users*. There can be two types of support provided - while creating an experiment as a maker and while using the experiments as a learner.

Supporting makers is done by providing adequate tools with which the maker can create the experiment CI after assembling the experiment setup. These include validating the commands that are executed on the rig and attempting recovery.

Supporting users is the process of determining whether the user is performing optimally for an experiment and then providing possible support in taking the future steps in the experiment session.

Both of these are done with the MDP described earlier. For each experiment, there is a CU that follows predefined *control policies* created by the experiment maker to operate the experimental setup as shown in Figure 6.3 to guarantee stability and reliability. The control policies are conceptually a separate functional feature that lies between the CI and CU and processes the command flow between them. The control

policies may be applied in many ways such as along with the UI and CPL or separate software. However, in P2P RAL [119], it is difficult for *makers* to create specific control policies by themselves. Each experiment has definite *goals* to achieve corresponding to its learning outcome and unless proper guidelines are set by the makers, the learners may be unaware on how to operate the rig to get to those *goal* states.

6.2.1 Control Policies for Centralised and P2P RAL

Remote Access Laboratories being remote in nature must have some form of automation of the experimental rigs to help guide the experiment run without the assistance of humans. The automations often involve a mechanical or electro-mechanical device that re-configures the experiment rigs as the current learner wants in a given experiment session. The RLMS implements the control policies that determine the exact manner of operations and the limits of parameters both inputs and outputs. The control policy differs between experimental setups depending upon the components used and their configuration. The main aims of control policies are validating commands and attempting recovery.

Validation of commands is the process of identifying whether the rig will reach an invalid state. The CI in the RLMS must make sure that the rig is always in a stable state by blocking or rejecting any inputs that are not within the allowed range of parameters. Validation of commands before executing them is very important as in a remote laboratory condition, the access to experiment is automated and if a an experiment becomes unavailable due to improper command execution, then it may take lengthy periods of time to reset the experiment to a usable state.

Despite taking measures, rigs can still enter into unstable states. Attempting recovery includes steps to bring the rig into a stable state if that happens. The CU informs the RLMS and makers about any unstable state that is persistent and cannot be rectified without human intervention.

In a centralised or federated RAL, the RLMS is managed from a select set of computer nodes. The entire RLMS is provided as a service by universities or institutions [167]. The RLMS stores the control-interface (CI) created by the *maker*, which collects inputs for operating these rigs. The control policies regarding the

inputs are hard encoded into the control program. For example the definite limits of an actuator are specified in the program. This requires deep understanding of programming to create such policies. Also, it is more difficult to alter these policies in case the hardware is changed.

Supporting Makers

In a P2P RAL, individual *makers* are given the opportunity to create a rig and it's CI. The maker, based upon their knowledge of any particular experimental activity creates and shares an experimental setup. MCUs give enough flexibility to create an experimental rig along with necessary automation. Makers can connect any sensor and actuator to the ports and create the CI using a visual programming language on an online platform [118].

However, providing this flexibility comes at the cost of lower reliability of the components used in the rig. The validation of commands is even more important in case of P2P RAL as there could be very little support from experiment hosts. The rig must be protected from entering a state from which it cannot transit to another valid state. Also, the makers of the experiments have disparate backgrounds and knowledge about control and automation. The experimental rigs and CI created by them are less reliable both in terms of the control policy implemented as well as the actual equipment are not guaranteed to perform accurately for a lengthy period of time especially without human supervision. Thus the makers must train the rigs to create the appropriate control policy for an experiment setup.

The *control program logic* created by the maker, process the learner inputs for the UI to generate the corresponding symbol sets or communication commands composed of a combination of these three atomic instructions.

Figure 6.4(a) illustrates a typical example of a MCU based experimental rig. It is built using LEGO Mindstorms parts and based on a LEGO smart Brick as the MCU. This experiment demonstrates a pendulum with 3 actuators. The aim is to swing the pendulum and take measurements at different heights of the ball. Figure 6.4(b) is the corresponding web browser based CI with a number of buttons relating directly to an actuator on the rig. The users can view the outcome of the experiment via a web-camera stream as shown in Figure 6.4(a). There is also an animated character that can

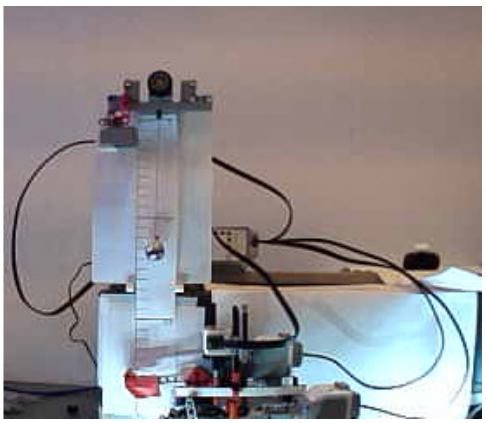


Figure. 6.4. (a) A pendulum experiment setup

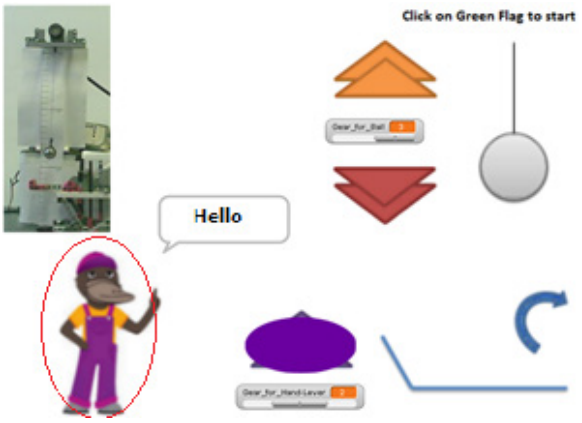


Figure 6.4 (b) The control interface of a RAL experiment in SCRATCH.

provide feedback and guidance (red circle).

Makers have difficulties in hard encoding control policies in the CI as these require expert knowledge. They are however, capable of running the experimental rig with basic commands associated with a control interface. Systems constraints include:

1. The users and experimental rigs can be geographically located anywhere and interact with the rig via the Internet. Video feedback is used for viewing experiment outputs. As the system uses the Internet, control message between learner and experiment node are subject to delays. This means that there can be a chance that the learner may give an asynchronous wrong input depending upon what they perceive as the current state of the rig.

2. MCUs have limited computational capacity to process data per unit time.

3. The learners who interact with the experiment are provided with detail about the experiment and its goals. But, initially they will not be aware of the exact steps that need to be completed to achieve the experiment outcomes.

Also, as the P2P RAL is decentralised, there is no external entity to co-ordinate between the learner and the experiment rigs.

Supporting Users

Every experiment will have a set of tasks that signifies the completion of the experiment successfully. This means the experimental rig must go through the particular states of its state space within the experiment session. These set(s) of state

may be called *goal* states. In the P2P RAL system, it is difficult identify whether the learner have met those goals. However, the makers are able to run their rigs as they desire. This way they can train the experimental rigs to accept valid inputs only.

The MDP can not only provide the admissible boundary of the state space but determine the best sequence of actions leading to the optimal state transitions. Thus once the CU has the MDP of an experiment, it can determine whether a user has traversed the rig through certain states that meets the learning objectives i.e. the goal states.

While the Validation of commands and recovery attempts are important and integral requirements of the CU, supporting users may not always be required. This feature may be optional and disabled at maker's discretion depending upon the experiment.

6.2.3 Indicators in the MDP

The progress quality in the transition of the states is measured with the following values:

1. *Absolute distance* - d_{ij} *(*and *Δd)* - The primary indicator is the raw distance (*d*) between the current state and the goal state. This distance is the length of the shortest path in the corresponding policy π_j for current state *i* to goal state *j*. With each new state of the rig, the change in value of *d* indicates whether the learner has moved away from the goal state or not. $\Delta d < 0$ when the rigs state moves towards goals state and vice versa.

2. *Relative distance ə* (and *Δə)* given by

$$ə_{ij} = Length(P_{ij}) \,/\, [Length(P_{ij}) + n_a(ij)] \quad \ldots (6.5)$$

where P_{ij} is the shortest path in the MDP graph, between current state *i* and the goal state *j* and $n_a(i,j)$ is the number of pairs of adjacent nodes in the path P_{ij} for which are not adjacent in the policy π_j. This value indicates the relative distance to the rigs state from the goal state. A value of 1 indicates that the rig can transit to the goal state and it is right on track. A lower value indicates, that in order to reach the goal state the rigs has to go through some suboptimal paths in the policy which indicates it is off-track. Varying probabilities of success in the MDP means that sometimes the shortest transition may not always be the most preferable option in π. *Δə* is the change in *ə*

between state transitions.

ii) *Weighted Relative Distance* ϖ (and $\Delta\varpi$) given by

$$\varpi_{ij} = Length(P_{ij}) \ / \ Length(P_{ji}) \quad . \qquad \ldots (6.6)$$

This indicates the weight of the immediate action of the MDP. For a rig MDP that may have at least two directed edges that does not have an opposite directed edge but still maintaining a directed path between all states, this indicator ($\Delta\varpi$) shows any sudden change in the rig state that is very faulty in terms of getting to the goal state. For instance, in Figure 6.1 if the rig is in position A, then going to D takes is 1 step, but if there is one bad decision of moving to B, then the feasible path length immediately increases suddenly. This sudden change could indicate severe learner's mistakes particularly for one-way transitions of states. $\Delta\varpi$ is the change in ϖ between state transitions. Note that only when a learner makes a wrong choice and chooses a wrong one-way path, $\Delta\varpi > 0$. For all other type of transitions before and after that, $\Delta\varpi = 0$. Thus it is very easy to detect such a mistake.

The change in indicators may be used in multiple ways, depending upon the system it is being used in, to obtain a value representing all three changes. The rig can then automatically decide whether it should intervene in the agents control commands and how much it should govern itself.

Using the indicators

While the *indicators* may be used in multiple ways to evaluate the system and the agent's status during the operation of the rig, in case of RAL a simple binary evaluation is proposed here. The purpose of the intelligence in the MCU using the MDP is to guide the learner automatically and decide whether helping the learner is required or not. Thus the evaluation result (κ) can have only the values '*yes*' if the state change was profitable in some way or '*no*' otherwise., depending upon the rigs position in the MDP. However to make the decision, the last few transitions must be monitored and recorded accordingly.

Whenever a command is executed, the Algorithm 6.2 *Evaluate(i)* is run for the current state *i*. If the current state is a goal state, then all variables are reset. There are three variable to monitor the *d*, $\Delta\varpi$ and the total number of transitions. The number of time

d increases for all goal states, number of time $\Delta\varpi \neq 0$ for any goal state and the number of transitions are recorded in q, r and p. If any of these goes over a threshold r_c, q_c or p_c respectively without reaching a goal state, then the users is in need of assistance and κ return 'yes'. The threshold p_c is determined by the distance to the closest goal state at the beginning of the experiment session or when a goal state is reached. A tolerance of ε that maybe added to p_c along with the values of r_c and q_c are system settings put by the system administrator or the maker changing the difficulty level of getting any help for the learner.

Algorithm 6.2 Evaluate(i) with global counter variables p, q, r

if $i \in G$ *then*

$p \leftarrow 0, \quad q \leftarrow 0, \quad r \leftarrow 0, \quad p_c \leftarrow \min_{j \in G \,\forall\, i \neq j} d_{ij}$

else

if $\exists\ k \in G, \Delta\varpi_{ik} \neq 0$ *then*

$q \leftarrow q + 1$

if $\nexists\ j \in G, \Delta d_{ij} < 0$ *then*

$r \leftarrow r + 1$

$p \leftarrow p + 1$

if $p > p_c + \varepsilon$ *or* $q > q_c$ *or* $r > r_c$ *then* $\kappa \leftarrow$ *yes*

else $\kappa \leftarrow$ *no*

6.2.4 MDP Inputs

There are few variable components that need to be defined or acquired from the makers to be able to create or use the MDP properly. These are:

1. *Initial state*: As mentioned earlier, in the MDP there exists a path between every pair of states. Thus the maker can choose an initial state. The experimental rig will revert back to the initial state at the start of each experimental session by traversing through all the intermediate states.

2. *Whether all transitions can be treated as bi-directional?*

3. *Whether state interpolation is allowed?* The MDP defines a boundary which is at the best partially known as maker is not expected to cover all possible feasible state of the rig in the training. It is in theory possible for the MCU to transit to an intermediate state that lies on the path between any two known states. However, such interpolation may not be allowed at all if the rig is not completely free to operate and may break down at certain states that are in between known stable states. Allowing interpolation assumes bi-directionality is allowed.

6.2.5 Rig Operation

The rig operates by first receiving the incoming commands and then processing it. With the MDP architecture, the rig is intelligent enough to make decisions on its own whether the learner/agent is following a feasible chain of control commands. The rig operation goes through the following steps:

Step 1. At the beginning of each learner's experiment session, the rig reverts back to the initial position and the values of d, ϖ and initial p_c are calculated.

For every new *write* instructions, the steps 2-7 are repeated. Other instructions (*read* and *wait*) are executed immediately.

Step 2. When a write command $\mathbb{w}(P, V)$ is received in state s_i the corresponding expected state is calculated. It is checked whether executing this command will lead to a state (s_{i+1}) such that, s_{i+1} is valid and (s_i, s_{i+1}) exists in the MDP. If it is not valid and interpolation is allowed s_{i+1} is checked if it can be interpolated.

For a state (s_{i+1}) to be an interpolated state, there must exist at one other state $s_2 \in \{\text{III} - G\}$ such that,

- s_i, s_2 exists in MDP and
- the feature vector i.e. the values of all the n ports must be exactly the same except only for one port (say n_j) in s_i, s_{i+1}, s_2 and the value for n_j in s_{i+1} should lie between n_j in s_i and s_2 i.e.

$$s_i[n_j] < s_{i+1}[n_j] < s_2[n_j]$$

This ensures that the interpolated state is actually traversed by the changing port, but not recorded. This may occur when there are different parameters, like speed of the servos (see Figure. 6.4), used while training from the one used by the learner. If it is not valid and interpolation is not allowed then s_{i+1} is *undesired*.

Step 3. If the new state is not *undesired* i.e. *valid* or *interpolated*, then the command is executed. Otherwise the command is rejected. After multiple rejections of write commands, the rig can decide that the user needs support.

Step 4. The resultant state (s'_{i+1}) of the command execution is matched with the expected state (s_{i+1}). If $s_{i+1} \not\equiv s'_{i+1}$, then the rig is in the *failed* state. In this situation, the rig tries to recover back to the previous state s_i by trying to write the value to the ports as in s_i.. If the rig cannot restore the states of all ports to the earlier values, it is considered broken and requires the makers intervention.

Step 5. Once a state is successfully changed, the values for Δd and $\Delta\varpi$ are calculated. In the RAL scenario, the probability of success of an action from any given state is very high and generally equal for all transitions. So the value of $\Delta\vartheta$ is not useful in context of RAL. However, the other two indicators, Δd and $\Delta\varpi$ are very important. The value for κ is then calculated with Algorithm 6.2 *Evaluate()* for the resultant state. If κ is 'yes', the learner is provided with hints to the next feasible state towards the nearest goal state. The nearest goal state *j* is the one for which the path is the shortest in corresponding policy π_j from the current state for all active goal states.

Step 6. Once a goal state is achieved, it is considered done and from the learner's perspective there is lesser incentive to re-approach that state. Thus, once a task state *j* is reached, its corresponding values for Δd and $\Delta\varpi$ are not considered for calculating κ in *Evaluate()* i.e. removed from G.

Step 7. If the current state s' is interpolated from previous state *s*, then add s' to the MDP. At this point there is no edge between s' and *s* which could also be the case if *s* was interpolated by the last command. In either case

add edges (s, s') and (s', s) and incorporate them with the *fail* states $f_{ss'}$ and $f_{s's}$ into the MDP by following Step 4-6 of MDP generating Algorithm 6.1 (as in Section 6.1) accordingly by applying the steps on the new edges/states. Then re-calculate the policies.

There may be possible temporal relation between learning objectives and correspondingly the goals states. In some experiments it may be required to complete a set of tasks before proceeding to others. This can be handled by activating a new reward strategy at the given time once a certain goal state is reached.

To do this a directed graph of goal state (G_T) nodes may be maintained. Any directed edge $a \rightarrow b \in G_T$ implies that the goal state b can be active only when goal state a has been attained at least once. Thus initially only a small set of goal state are active and available. In some experiments there can be only one initial active goal state if the order of learning objective is very strict. Note that the user may or may not go to a goal states that is currently not active, but it will not be counted as part of the evaluation until all it's previous state have been attained.

Using the tree, for algorithm *Evaluate(),* only those goal states are considered that are currently *active*. Once a goal state a is achieved all goals sates b that are connected to it such that $a \rightarrow b \in G_T$ are considered active and a itself becomes inactive.

6.2.6 Example and Results

The example considered to illustrate the use of MDPs is the pendulum experiment mentioned earlier. This experiment has three actuators i.e. the feature vector in each state contains three values of the actuator (n = 3). The values returned are integer numbers (if the servos rotates twice full circle, the value is 720 degrees; if it rotates backwards then the value is -720 degrees). The rig was trained with a sequence contains (ϱ = 79) transitions that generated 73 states in the MDP. There are 4 goal states defined in the experiment - C7, C16, C25 and C34. The learner starts with the state C0 which is the initial starting position. The learner can send commands to the rig and leave the experiment in any random position at the end of their session. The rig takes its state back to C0 for the next session. The probability of success of each command to the rig is 0.99. This is a high value as there is little probability of it failing and it is and it is equal for all transitions as all the actuators have the same

reliability. The training allowed addition of bidirectional edges as all state transitions (s_1, s_2) recorded can be done in reverse (s_2, s_1).

Figure 6.5 shows the final values of each state after the VIA is run corresponding to the 4 goal states. For each of them the goal state has got the highest value followed by the state that is closest to it e.g. C7 is adjacent to C6, C8, C12 and C13. The *fail* states also get high values, but as the failed states are only connected one valid state (and the reward for the transition from the valid to the fail state is 0) for all fail states, the outgoing edge is chosen in the policy (π_i). The values of all the fail state closely follow that of their corresponding valid state, but are always smaller.

Figure 6.6 shows the distance form any state to the nearest task state. As the probabilities of transition success are all same and the transition are bi-directional, the values for $\Delta\varpi$ and $\Delta\text{ə}$ always remains the same. So this experiment, the value of $\Delta\varpi$ has no meaning and the evaluation (κ) is solely depended on the value of d. For example $\Delta d > 0$ if the ball is moved up many times beyond the reach of the hand lever as in states C6 to C0 and to C3. The distance will keep increasing to all the goal states.

Note the significant increase in the distance for the states from C50 to C73 in Figure 6.6. This is due to bad training as these states were generated as part of the training data set. They basically represent the maker generating transitions that are not very effective towards reaching the goal. This is important for system like RAL for teaching purposes if the maker want to make the usage of the rig as flexible as possible.

6.2.7 Using MDP in P2P RAL

The MDP creates a unique data structure for an experiment interaction. It presents a mathematical model of the experiment usage and thus the MDP can be used in the P2P RAL to aid both makers and users.

Makers' main advantage is that they do not have to enforce the control policies on their own and hard code them into the CI. The maker's inputs to the creation of control policies are minimal. The CUs can identify the most ideal sequence of activities and act accordingly. The safety and integrity of the experiment setup can be ensured with the validation and by keeping the rig within a desired limited state space.

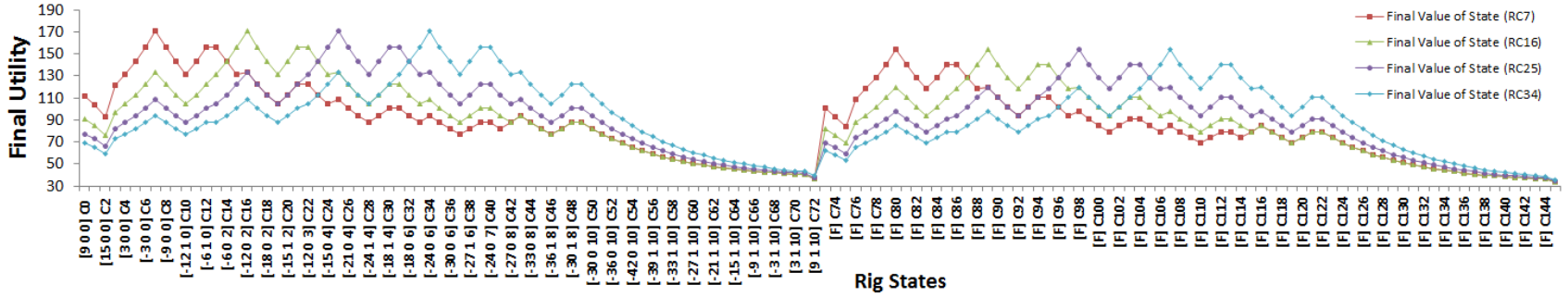


Figure. 6.5. The final utilities or values of the states in each α_i corresponding to the goals states C7, C16, C25, C34. For failed states only the highest value of shown for all failed states for a valid state.

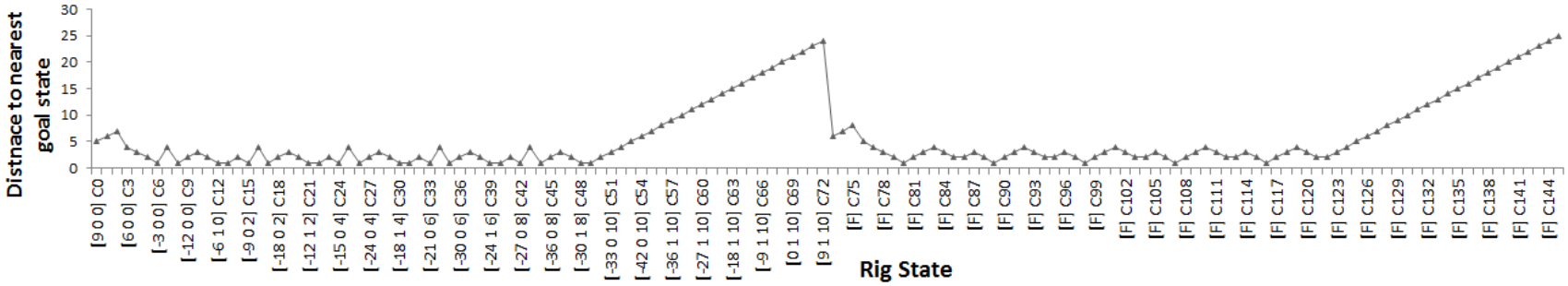


Figure. 6.6. The distance to the nearest goal state for each state. For failed states only the smallest value is show for each corresponding valid state.

If any recovery procedure is successful, the makers can be saved the trouble of resetting experiments as well in case something goes wrong. The RLMS can automatically keep track of the user interactions and thus determine whether the user has performed as desired by the maker.

For the users, they can be provided with support and be monitored. The CU can provide the support by detecting whether the user made a wrong decision with regards to reaching the goal of the experiment activity. This could also help in making time-critical decisions when required.

With the MDP, the rig is intelligent enough to judge the quality of the users' use of the rig. Thus the system can allow for evaluating the learner's performance. A higher number of instances where the users make wrong decisions ($\Delta d > 0$ and $\Delta \varpi \neq 0$) can be recorded and a feedback may be provided on the interaction.

All of this can be achieved without setting any specific limits in the IEM for the CU. The rigs i.e. CUs can all run the same algorithms to create and parse the MDP regardless of the experiments.

Presenting Guidance to Users

Once the CU determines that guidance or support is required, it has to be presented to the user. The exact methods of providing guidance to the learner is out of scope, but once the decision to guide is made and the path to the goal state established, the rig can guide the users using any visual cue. For the MCU controlled experimental rigs, the actions are each individual commands allowed from the CI components. All actions may not be defined for all states restricting the learner and implementing the control policy. A catalogue of learner-friendly *terms* may be defined for each command or MDP action $\mathbb{w}(P, V)$ by the maker and stored against the corresponding edge (*s, s'*). While presenting the guidance, the next steps can be presented on the interface by using the corresponding *terms* of the action required to change from current state to the next state in π_j in the shortest path towards the nearest goal state *j*.

Limitations of using MDP

There are some limitations of the MDP approach. The major limitation is the need for training data. The experimental rig must be used multiple times by the makers and

testers to generate a sufficiently large training data set that can encompass all aspects of the experiment. This means that the rig must be used to its operational limits to ensure that the training data set as well as the corresponding MDP can cover all possible states. This is difficult to do perfectly as makers may not foresee all possible uses of the rig, thus rendering certain inputs from the users invalid with the respect to the MDP even though they may not be unstable. In case of remote laboratories, it is used for learning purposes and the proper way to achieve the goals is as important as the learning goals themselves. Thus, following the makers' steps is acceptable when applying the MDPs to the RAL scenario.

Also, the goal states may be difficult to judge. In a poorly designed rig, the actual events of the experiment result may not be captured properly from a particular state. In those cases, the effectiveness of the indicators reduces. But, this can be resolved by adding a dedicated sensor which will confirm the task events taking place. The particular variable then can uniquely identify the task state.

Finally, recovering a rig is very difficult if multiple ports' values were changed in parallel in the previous command. But it is simple if only one port is changed in one atomic command i.e. there is no parallel change in the ports.

The flexibility to match intermediate states can mitigate the impact of improperly trained rigs to certain extent, but the interpolating techniques needs to be improved and the rigs, very well trained. However, with a large number of users using the system as part of the training, the MDP can be accurate.

The MDP can be used to determine whether the user has reached the desired goal states and corresponding learning objectives. This could suffice to evaluate the users' performance with the experiment in most cases where there are clear goal states. However, some experiments may have goals state that needs to be reached multiple times. In these cases, the just reaching goal states do not indicate a good learning outcome. The exact manner in which the experiment is used must be determined to evaluate the users. An example of this could be an experiment of moving a robotic car in an open space. The state space of the car is finite and there can be certain goal states corresponding to positions reached in the space, but reaching those states does not guarantee that the user has used the experiment correctly.

The solution to this involves profiling the user interaction and comparing it with the maker interactions. The profiling requires data mining techniques and a special clustering algorithm is proposed in the next chapter.

6.3 Summary

This chapter described MDPs a universal tool for evaluation, validation and guidance during a users' experiment session. An MDP based model of the experiment is presented which is an extension of the CI-CU model by converting its state space into an MDP. This MDP is then used to evaluate the user's performance, provide support and ensure the safety of the rigs.

In terms of IoT or WoT, the MDP model of rigs and the clustering algorithm can be used for any application with multiple operational master-slave nodes. The main contributions are the three indicators. In a given applications such as an IoT system, these indicators may be used in various ways to determine the course of actions and also find out whether the system is taking the best decisions or not. It is useful in determining the impact of decisions in any environment with autonomous agents taking decisions.

7

Intelligent Tools: Advanced Evaluation

This chapter presents a method for advanced evaluation and validation of users' interactions with experiments.

As mentioned in the previous chapter that the MDP may become ineffective in certain circumstances. Thus a different approach is required to match and evaluate user interactions with respect to the maker interactions for a given rig. This approach is based on the relative difference in size and frequency of composite and atomic commands with respect to time. The chapter focuses on using the temporal relationship between commands.

A new constrained clustering algorithm is proposed in Section 7.1 for advanced evaluation of the user interactions and also adaptive user interfaces. The clustering algorithm creates/identifies and analyses the *clusters* or groups of executed commands within a time period to determine the manner in which the rigs were used. It can be used to create clusters of commands with desired properties. For creating clusters, all commands passed during maker experiment interactions must be recorded according to be used as training data. The content of this chapter is based on [168].

Clustering is used to obtain composite commands from the atomic commands. Clustering commands allows the experiment controller to obtain usage patterns as described in Section 7.2 for a particular experiment. It also identifies composite commands when necessary as discussed in the next chapter.

7.1 Clustering Commands

As discussed in Section 4.7, several levels of commands can exist for a given experiment. A higher level of language or composite command can be composed of

smaller lower level or atomic commands. The clustering algorithm aims to aggregate atomic or lower level commands into a definite set that can be referred to as *closely related commands.*

Such a set of commands can be *loosely coupled* i.e. not strictly part of any higher level composite command. But it will indicate which components i.e. CU ports and their corresponding devices in the experiment are closely related or frequently accessed for an experiment. This chapter use the *loosely coupled* approach to create a profile of the experiment interaction based on the makers' interactions. The set of commands clusters can be *closely coupled* as well when it forms a definite higher level composite command. This property is exploited in the next chapter.

The aggregation of commands signifies multiple repeated instances of the same set of atomic or lower level commands being executed. This aggregation can be done with a data clustering algorithm as described here. The input is a set of commands according to a timeline collected from the makers' interactions. It is represented as a one-dimensional data set *D*.

7.1.1 Literature Review - Clustering of data

Clustering is a large aspect of data mining related system implementations. It aims to create groups of data from given datasets such that each group contains similar data points which are different from other groups. There are several strategies of clustering based on the user's requirements [169]. Clustering has also been widely used in networking and geographic information systems [170]. Some of the major clustering algorithms are Hierarchical Agglomerative Algorithm (HAC), DBSCAN and k-means. Some of the common strategies for creating a cluster are Distance-Based Clustering, *Partition-Based* Clustering e.g. *k-means* and *Density-Based* Clustering.

Distance-Based Clustering assumes the relationship between each points or entities in terms of the Euclidian distance between each of them, i.e. the closer the points are, the more likely they are to be in the same cluster. The well-known Hierarchical Agglomerative Algorithm method is the widely used implementation of this strategy [169]. However, general hierarchical clustering does not specify any upper bound on the cluster size in terms of the distance. HAC follows a greedy method to iteratively join the nearest data points together to form a new cluster until a desired condition is

achieved.

The k-Means Clustering Strategy [169] works by first assuming a predetermined number of clusters present in the data space. Then a set of positions are generated either randomly or using a metrics such as Kaufman Allocations. [171]. These are each assumed to be the mean or median of a cluster representing it at the start. Subsequently, data points are compared and merged with the existing clusters to generate new means or median until all points are merged.

Density-Based Clustering creates clusters based on the proximity of points [170]. The DBSCAN is the most popular implementation of this method. This method uses two inputs - a lower limit on the number of data points in a cluster and an upper limit of the maximum distance between two points in a cluster. This method is faster than the previous two in all cases and can identify irregular shaped concave clusters. However, there is no way to specify an upper bound on the diameter of the clusters. Hence, the clusters are formed by gathering all closely situated points or sites into one cluster regardless of the diameter.

For creating the clustered command sets, a constrained hierarchical agglomerative clustering (CHAC) method is used. The data mining approach of clustering usually aims at maximizing the size of a cluster without any constraints and defines cluster size as the number of data instances in a cluster. A clustering approach with maximum size has been discussed previously [172]. This assumes that the number of clusters is known beforehand. Some of the prevalent clustering strategies are examined for their suitability to use in this problem.

The following sections propose a new clustering algorithm approach based in HAC to create a cluster of commands. This proposed approach is then used to identify closely related components to create profiles of experiment interactions.

7.1.2 Proposed Clustering in P2P RAL Control

The HAC [168] is performed by a greedy method where initially every element is considered to be in its own cluster. The HAC algorithm then iterates through all pair of elements and in each iteration the two closest clusters in distance are combined to form a larger one. This creates a hierarchical structure of clusters. The iteration stops when a desired condition in terms of clustering is obtained. In the current context, two

constrains are implemented for the *CHAC(D, ε, y)*:

- Two clusters may be joined together only when the distance between them is less than ε.
- No cluster can have more than γ elements in it. Hence, if the shortest distance is determined to be two clusters that could have a combined size i.e. number of elements of more than γ then they can never be joined. The algorithm then moves on to check the pair of clusters with the next shortest distance.

The clustering process begins with a one-dimensional set of data points (D) that represents the command and the distance between them is a single integer value representing the time difference between them. The CHAC process iterates through each pair of commands and joins the two commands that are closest to each other respecting the two conditions mentioned above. The algorithm stops when no clusters can be joined any further.

Clustering on a timeline with commands with respect to time will generate a set of clusters with at most γ commands in them and the time difference between each successive command is ≤ ε.

Algorithm 7.1 CHAC(D, ε, γ)

Initially each command $c_1 \ldots c_n \in D$ is in its own cluster $P_1 \ldots P_n$

allsitesclustered ← false

While allsitesclustered = false

Find the pair of clusters Pi, Pj with minimum distance i.e. min{d(Pi, Pj)}

If $d(P_i, P_j) < \varepsilon$ and $|P_j| + |P_i| \leq \gamma$ //according to Constrains

Join P_j to P_i and remove P_j

If $\nexists (P_i, P_j)$ such that $d(P_i, P_j) < \varepsilon$ and $|P_j| + |P_i| \leq \gamma$ then

allsitesclustered ← true

End while

7.2 Proposed Method of Evaluating User Interactions

In this section, a proposed Closely Related Component (CRC) list is described as a

method to determine whether the rig has been used according to makers intended design.

7.2.1 Command Operations – Mathematical Notation

The commands are issued to a particular port that sets a value for it. The state of the ports on the experimental rig is reflects the state of each port of the CI. For any component x_i, $\Lambda(x_i)$ represents all commands that are issued to x_i. For a set of components J, $\Lambda(J) = \Lambda(x_0) \times \Lambda(x_1) \times \ldots \times \Lambda(x_{|J|})$. As discussed, there are two basic types of commands:

- *READ* - This command returns the value of the specified port that is connected to a sensor to gather data from the rig's environment. For example $l \in \Lambda(x_i) \Rightarrow l = READ(x_i)$.
- *WRITE* - This command sets the port specified connected to a particular component (for example, an actuator) in the rig to a particular value. The *WRITE* commands returns with true is the command was executed successfully or false otherwise. For example $l \in \Lambda(x_i) \Rightarrow l = WRITE(x_i, \ value)$

Conversely, if $x_i \in \Lambda^{-1}(u)$, then x_i is referred by the command u. Likewise a set of components may be called as a component set (J) of a group of commands B if there is a one to one relation between successive commands in B and components in J. Also, $l_1 = WRITE(x_i, v_1)$ is ideally different from $l_2 = WRITE(x_i, v_2)$.

For finding the closely related components, the values passed in the *WRITE* commands or whether it is a *read* or *write* command are not relevant and this section concentrates on the invoked components only. In the current context, this provides with a less restricted profile of the users' interaction. The values and command types may be taken into account of necessary for very strict profiling.

7.2.2 Command Flow

During the experiment duration, the state of the system changes according to the user command i.e. $Y_{t'}(i) = f_t(c_{i-1})$ where $c_i \in \Sigma$. The controller must ensure that commands c_i are executed in order and with the time interval such that

$$Y_{\tau'}(i+1) - Y_{\tau}(i) = f_{t'}(c_i) - f_t(c_{i-1}) \quad for\ i = 1,2,\ldots$$

where τ' and τ represent the time instants at the rig and t' and t represent the time instants at the CI when the commands (c_i and c_{i+1}) are issued. The latency (ψ) in the network means the execution of the command will be i.e. $\tau > t$.

Figure 7.1 shows a typical communication flow between the CI (S) and the CU (Y). $l, u, v, w \in \Lambda(X)$ are instructions or commands from S to Y. Due to network delay or latency, the command l arrives after ψ_l time to Y.

7.2.3 Closely Related Components

From the CI-CU model, Y has a finite set of physical states based on its ports R as described in the MDP. Let these states be

$$Q_R = Q_A \cup Q_B$$

where Q_A is a set of stable states and Q_B is a set of unstable or intermediate states. Due to the nature of the CI-CU model, discrete functions are associated with the CI for a given experiment. This means that each experiment has a unique profile of commands that are executed on it depending upon the commands (atomic or composite) called from the UI. Thus a rig will enter a set of commands that will keep it in Q_B i.e. an unstable state in terms of the experiment for a period of time before it enters a stable state in Q_A which signifies the end of a composite command. Hence there can be a repeating sequence of commands in the communication i.e.

$$L_Y^j \ni uv^i w \qquad for\ i \geq 0, \qquad j > 0$$

where u is a starting instruction of a composite command which take the rig to Q_B and

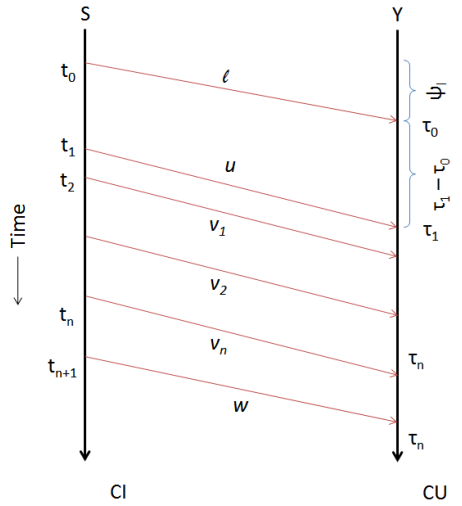


Figure. 7.1. An example experiment session communication flows.

w is the final one which brings it back to Q_A. Any set of commands *v* may be repeated in between *u* and *w*. Whenever the rig is not being used, it is in Q_A.

A CRC set (*J*) is a set of components that are referred to in the command sequence repeatedly i.e. there exist $uv^iw \in \Lambda(J)$ for $i \geq 0$ and within a short period of time limit (ε). $J = \{x_0 \ldots x_j\}$ is a complete CRC if there does not exist any $x \in J$ and $x \in J' \neq J$ and J' is also a CRC. A component may be adjudged closely related to itself, if commands are repeatedly executed over it within ε. The time difference between each successive commands $\{u, v\} \in A$, is ideally negligible, or the commands are simultaneous i.e. $t_v - t_u \approx 0$. However for clustering analysis, the time difference between the executions of two CRC commands may not be larger than the limit.

$$\tau_v - \tau_u \leq \varepsilon$$

The components may not be complete CRCs and for any J, the strength of the relation between components may be defined as *closeness co-efficient* ($\Phi(J)$) and *usage probability* as $\theta(J)$ with both values between 0 and 1. It may be noted here that the value of ε is considerably greater that the time gaps (< 10ms) between commands considered in Section 5.6 for queueing purposes.

The goal is then to identify and record the probability of occurrence of such command chains or actions (uv^iw) that appear in high frequency in an experiment session. To do this, first a clustering step is performed to identify the dense command zone and command chains (A) over a timeline. Once these command chains have been discovered they are used to establish the characteristics θ(J) and Φ(J) of these.

7.2.4 Preparing the CRC List

The clustering process partitions the whole set of commands in an experiment session into a set of command chains $B_\gamma \subset L$, such that, for all elements $y \in B_\gamma$, $|y| \leq \gamma$. The CRC list is a 2-dimensional matrix with each row depicting a CRC Set and its properties (θ and Φ). For a given historical data set (*D*) of an experiment session containing all commands according to time they are executed on the instrument, a *CRC list* (W_C) is obtained as follows:

Step 1. First a suitable value for ε is chosen depending on the desired application of the CRC List.

Step 2. Then the $B_\gamma = CHAC(D, \varepsilon, \gamma)$ is executed repeatedly for $\gamma = 2, 3 \ldots$, until a desired precision of differences in successive $\Phi_\gamma(J)$ and $\theta_\gamma(J)$ are achieved i.e. they converge. At the most the γ can be increased up to $|D|$, but both $\Phi(J)$ and $\theta(J)$ converge after only a few iterations as the clustering is restricted by the distance constraint (ε). For each value of γ, the following steps are performed:

Step 2.1 Once the clusters are formed, each cluster of commands $b \in B_\gamma$ is replaced by its component set. For example if $b = luv$ where $l \in \Lambda(x_0)$, $u \in \Lambda(x_1)$, $v \in \Lambda(x_2)$, then the component set of $\Lambda^{-1}(b) = x_0x_1x_2$. Thus all clusters in B_γ can be re-structured as its component sets to obtain T_γ where $a \in T_\gamma \Rightarrow a = \Lambda^{-1}(b \in B)$ i.e. T_γ contains all the components sets of each clusters of B. This gives the cluster of components as they are referred by the commands. Any cluster can contain multiple instances of the same component.

Step2.2 A global list (W) of all unique clusters obtained for various values of γ is updated with records as $\langle J, E(J)\rangle$ where $J \in T\gamma$ and $E(J)$ is the number of times J as appeared in $T\gamma$ for all γ until and including this iteration. J may appear in T_γ for multiple values of γ. A new CRC list (W_C) is prepared from W in each iteration of γ and matched with the previous iteration. W_C has each entry as

$$\langle J, \theta(J), \Phi(J)\rangle$$

For each $J \in W$ (and also in W_C) there is a probability of being executed in the experiment session given by,

$$\theta(J) = E(J)/T$$

where E(J) is taken from W and T represents the sum of all $E(J)$ in W i.e. the total number of separate clusters of components sets recorded until and including this iteration.

$$T = \sum_{J \in W} E(J)$$

Also, for each J i.e. CRC set, there can be a degree of closeness among its components,

$$\Phi(J) = (E(J) + G(J)) \,/\, (E(J) + N(J))$$

where *E(J)* is taken from *W* and

$$N(J) = \sum_{x \in W \ \wedge \ x \neq J \ \wedge \ J \cap x \neq \emptyset} E(x)$$

N(J) > 0 means that the components called in J are not exclusive to *J* and some component of it has been invoked from another components set *x*. This basically gives the probability, at a given time, of *J* being executed in the experiments session, provided any of its components is being called at that time. *G(J)* represents the sum of all $E(x)$ for any $x \in W$ such that $x = J^i$ for $i > 0$ i.e. *x* is composed solely by repeating *J*. Note that any command set with $\Phi(J) = 0$ or $\theta(J) = 0$ will not appear on the CRC list.

The *CRC list* obtained in the final iteration of *Step 2* is the *CRC list* of the experimental rig obtained with *D*.

For two training data sets D_1 and D_2, if the value of ε is same, then they may be concatenated and processed. Otherwise two separate CRC lists are created and merged such that the resultant CRC list:

- has all rows for each $J \in D_1$ or $J \in D_2$
- the values for θ(J) and Φ(J) are averaged if J is common for D_1 and D_2

This list may be sorted according to *γ, θ(J)* or *Φ(J)* depending upon the required information from the list. For example sorting the list by decreasing order of *θ* gives the indication of probability of any command set J to be executed in an experiment session from highest to lowest while sorting on the Φ gives an indication of most strongly bonded CRC sets and the corresponding command chains. It may be sorted on γ to obtain the number of commands set for each value of γ.

The next sections presents a test case, results and applications of an experimental rig and its CRC list.

7.2.5 Example and Testing

For testing the proposed CRC List generating process, a LEGO based robotic vehicle with a mounted sensor (see Figure 7.2) was built. The two wheels actuators A and B work in a differential manner for making the robot turn and move in parallel for moving front and back. Two sensors (D and E) are mounted atop actuator (C). The

sensor (D and E) do not stream any value but the user has to request the value through an UI when they desire. A, B and C are also controlled through an UI. So there are a total of 3 actuators and 2 sensors in the rig i.e. $X = \{A, B, C, D, E\}$.

The maker of this experimental setup is unable to create any mathematical model of the rig and thus the only commands can be from the list as shown earlier in L(Σ). The users inputs passed to the UI are executed on CU i.e. the LEGO Mindstorms. The experiment is designed to move the robot around and collect data with the sensors at certain positions. A session of 145 seconds is recorded and used as a training data set. The network latency is negligible.

Choosing the values of ε is very critical to the correct use of the CRC List. The value of ε is dependent on the context the CRC List is used. For this example ε = 50ms is considered. This value is recorded as the minimum difference between inputs at the UI for this experimental setup. The corresponding CRC list can be used to analyse user experiences with respect to the user interaction.

Essentially, in this example all states, except the states that break the sensor wires by turning the sensor at higher angles, are goal states. This means that only actuator C is bounded by a lower and upper limit, but other two actuators can move infinitely within the open space.

Results

Figure 7.3 (a) shows the different commands sets or cluster that appear most frequently. The commands set AB appears most frequently with a θ(AB) = 0.69. Also it is obvious that AB are the most tightly bonded (Φ(AB) = 0.65) components as the experiments relies on the robot being moved frequently and then collecting data.

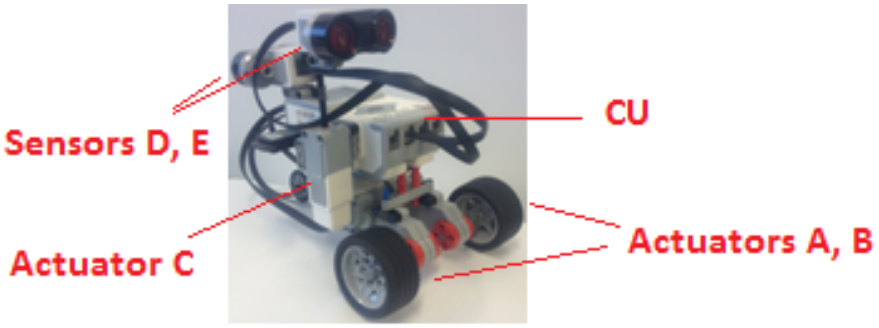


Figure. 7.2. A sample setup with LEGO Mindstorms EV3.

However, C has been used less indicating that the user relied on moving the robot to different positions rather that rotating the actuator C to different positions.

For Figures 7.3 (a & c), the step two iterations were performed from $\gamma = 2$ to 14 and 15. The values of the $\Phi(J)$ and $\theta(J)$ are almost similar and will continue to converge with increasing γ. Note that increasing γ also results in newer and larger command sets. The convergence is shown only for J common in both $\gamma = 14$ and 15. There are much larger clusters in T_γ with $\gamma = 15$. If the iterations are continued then there will be larger component sets and even more instances of AB will be generated. Thus any further values of γ may be used if searching for a particular large component set.

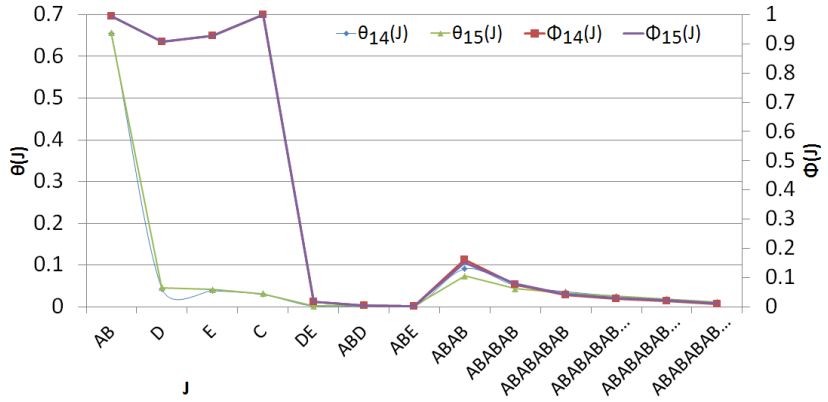


Figure 7.3(a) The component set and CRC list

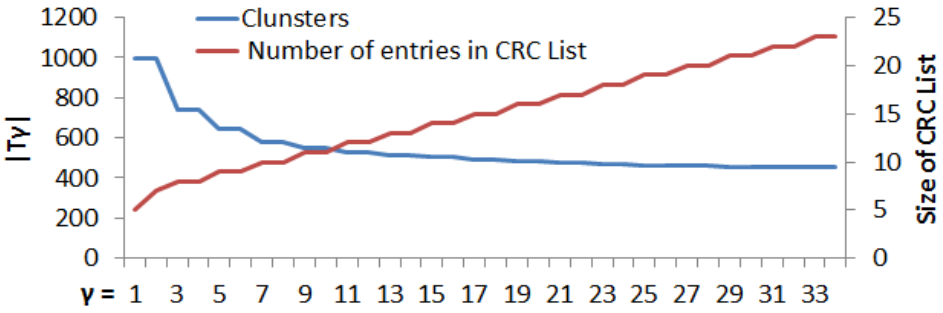


Figure 7.3(b) Change in the number of clusters and CRC list

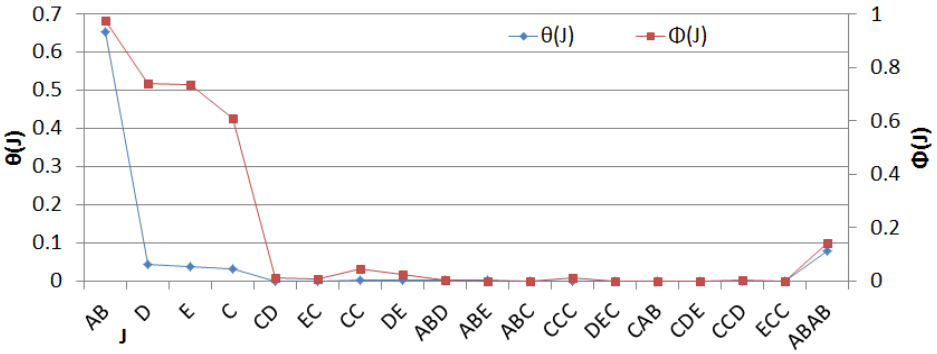


Figure. 7.3(c) The component set and CRC list (D_2)

Figure 7.3 (b) shows the number of clusters formed and the size of the CRC list itself i.e. the number of unique command sets identified with changing γ. The $|T_\gamma|$ value keeps decreasing and reached a stable state for γ = 34 while the CRC list increases almost linearly with increase in γ. Figure 7.3 (c) shows a different session of the same experiment (145 s), where the rig has been used differently. AB is still the most tightly bonded component set. But C has been used more frequently in this session.

The main application of this method of finding CRCs is for analysing and identifying individual actions. The method of obtaining CRCs list can be used to compare any two experiment sessions by analysing the most occurring commands sets and most accessed components of the rig. The makers' interaction with the experiment while building it generates the training data set (D_m) which can be used to create the CRC list CRC(D_m). The CRC list then cane be matched with other users' interaction (D_u) which can be used to generate a CRC list CRC(D_u) as well. While the two lists are not expected to match exactly equally, if the list contains similar elements with similar values for θ and Φ then the user can be deemed to have used the experiment correctly. Otherwise if there is new elements in CRC(D_m), then the user has done the experiment in a different way than the makers' expected.

After repeated use of the rigs, definite actions may be identified involving the CRCs pertaining to a particular experiment. The actions are classified as most likely to least likely as well as undesired. This will allow for determining whether of the rig is being operated in a desired manner or not. The rig can use the training data set to possibly identify certain inputs that will put it in an unusable state.

While this method to use the CRC list can indicate whether the rig is used according the makers desires or not, it cannot guarantee whether the learning objective have all been attained in the exact manner if there is strict temporal relation between the learning objectives and the corresponding goal states. But if a user reached all the goal states and has the CRC list matches the makers CRC list, then it is mostly likely that the user had good learning outcomes.

While learning outcomes are largely related to the way the rig is uses, thus making a fixed sequence of state changes in the rig most desireable, it may be possible in certain circumstances to entirely use different commands to still attain the same objectives. This is illustrated in the Figure 7.3 (c) which uses the same experiment in

a different way, but still could have collected the required data. However, the clustering cannot determine whther the rigs was used to achieve the required learninng objectives. Future works can look into resolving this issue.

7.3 Summary

This chapter has presented a cluserting algorithm that can be used to create profiles of the nodes behavior and validate any communication based on that. This form of evaluation can be effective where there is no clear goal state. User interactions are mapped to the maker interactions to identify differences. The discussions in this chapter have focused on relaxed clustering considering only closely related components an ignoring type and parameters. But if it necessary, the function Λ can be modified to generate specific symbols for type and parameters so that Λ has only a single element for a given combination of port, command type and parameter as a set which will apply a very strict form of evaluation.

In broader terms, with respect to the IoT, this tool can be used to identify and measure interactions of different master slave combinations in a master slave environment. It can also highlight any temporal differences for the same combination. This can add to the intelligence of each master and slave on what is an ideal communication exchange if the devices to communicate with multiple master or slaves with time.

In the next chapter, the clustering algorithm is used to further enhance user experience by identifying commands automatically.

8

Intelligent Tools: Adaptive User Experience

> *This chapter presents a method for adapting user experience with regards to network conditions.*

On the Internet, constant latency between the CI and CU is not guaranteed. Lost messages result in additional latency for re-transmission and queuing results in jitter. Thus, depending upon the experiment, if the events of the CI have any strict time properties, a higher level language using composite commands is required in order to maintain the required discrete events and corresponding commands. A composite command initiates a finite chain of executable instructions on the CU. This fixed set of instructions can retain any time-interval property required by the experiment. Also, for network-based control systems, fewer messages lead to higher system efficiency [150] and messaging quantity may also be reduced by queuing and using a proper protocol as discussed in Chapter 4-5.

This chapter concentrates on the problem of interactivity of an experiment in the context of RALs. First an experiment is describes as in a continuum in terms of its interactivity in Section 8.1. Then the clustering algorithm is used for ensuring a good learning experience as well as rig safety by altering the interactivity depending upon network conditions as described in Sections 8.2 to 8.4. Section 8.5 provides an example of the experiment continuum with a particular experiment.

This chapter entirely consists of new contributions [173] and discussions based on the previous chapters. It focuses on the transition between the two control strategies, DAC and UAC where commands can be aggregated into functions, which are stored on the CU and invoked by the learner at the CI. The automaton model helps in identifying the constituent commands of repeatable functions to be stored in CU.

The user experience can be enhanced by changing the interactivity of the experiment according to the network conditions and by providing them the best interaction with the experiment for given network conditions.

8.1 Experiment Interaction Continuum

The interactivity between the user and the experimental rig can be represented as a *continuum* from a fully real-time interactive experiment, to a fully batched experiment (see Figure 8.1). The same experimental setup can be expressed as a fully batched or fully interactive depending upon how much input per unit time is given by the users. The interactivity level depends upon the types of commands used (atomic or composite) and used according to the desired learning outcomes of the experiment. Although all experiments can in theory be run as both batched and interactive, not all experiments will have a suitable application in terms of educational outcomes.

The interaction between the user and the experiment can be defined based on learning objectives and rate of commands. In the current context, a *learning outcome* is the completion of a set of tasks or the rig being in a given state that explains some knowledge concept. An experiment session may be composed of a set of *goal states* (*G*) that occur after the commands from the users have been executed. The feedback of such a *goal state* is usually through video or some other data format. In P2P RAL it is through visual feedback and CU acknowledgements. There can be several intermediate states between goal states called the *milestone states* which are valid states that lead up to a *goal state* which is also a milestone state.

Any pair of commands must be issued to the experiment at a definite rate such that time between each pair of command corresponding to their goal states is static for the experiment.

An experiment can be described as "absolutely" interactive if it only uses atomic commands i.e. gives complete freedom to the users. The components of the experimental rig may still be subjected to a range of possible conditions, but the users

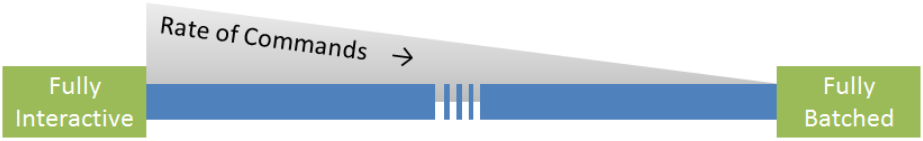


Figure. 8.1. The interactivity continuum for an experiment

are allowed to execute commands that make the minimum possible change in the rig's condition.

On the other hand an experiment may be described as a completely batched experiment, but not technically absolutely batched as the size of composite commands that can be created from the set of lower level commands is virtually infinite by staking multiple levels of instructions executions. An experiment can be said to be *complete batched* if the UI issues composite commands that causes maximum change in the experiment corresponding to the objective of the experiment. For a complete batched experiment, there can be only one command passed possibly with multiple parameters that gets the result all at once.

8.2 The Experiment Session

A *goal state* can be accomplished most effectively if the corresponding commands leading to the *milestone states* are executed with fixed time intervals in between. In a given experiment session (e), the *goal states* can be expressed as an unordered set $G = \{g_1, g_2, g_3 \ldots g_n\}$ where $n > 0$. Each element in G is accomplished after at least one relevant command has been executed for it resulting in the goal state. Thus

$$C(g_i) = \{ c_1^{t1}, c_2^{t2}, \ldots c_p^{tp} \}$$

where $p > 0$. If $C(g_i)$ is ordered then the commands can be aggregated into a composite command. But $C(g_i)$ may not always be ordered in which case smaller composite or atomic commands may be used. The most important factor is the time interval between the commands in $C(g_i)$. The value for the time gap for any pair of commands i.e. $\{t_{i+1} - t_i\}$ must be less that a constant value for ensuring rig safety. However, giving users control of the experiments requires that the users issue more of the commands in $C(g_i)$, rather than the CU automatically issuing them as a part of a higher level composite command.

It may be noted here that these time gaps $t_{i+1} - t_i$ is considerably larger than the time gaps ($< 10ms$) considered in Section 5.6 for queueing. These time gaps represent events that occur considerably apart in time to affect visual or structure changes on the rig and thus the learning outcomes.

For a given experiment and its *goal state* set, the experiment can be assigned a most

suitable or default spot on the interactivity *continuum*. For example an experiment involving moving a car will be highly interactive and use atomic commands and thus by default is near the *absolutely interactive* type. However, each experiment can be altered in terms of the rate of commands to become a more batched version of the same experiment by using more composite commands. If the rate of commands is compromised heavily it may affect the *milestone state* set. It could also affect the goals states if the goal states have strict time gap properties between them. For experiments that are by nature interactive, the learning outcomes are best if the rate of command is higher compared to its batched version. For example if a robotic car was to go from a point A to point B in a n experiment with *interactive* control, the user can run several atomic commands to complete the task. This could involve many milestone states and goal states in between the two points. On the other hand if a single function i.e. composite command in *batched* mode was used to accomplish the task, then the robot will still reach B from A possibly completing all the goal states but the user will have no control over the robot and can only watch the events.

Static time gaps can be achieved easily if the remote node and the experiment rig were close to each other such as in a LAN. In the P2P context, the makers can achieve this perfection every time they perform the experiment, as the latency is negligible on the maker site between the makers CI and the CU. The learning experience when operating remotely is dependent on the ability of the system to maintain the time interval as much as possible. The risk of not being able to maintain the time gap increases with lower orderliness of the components of $C(g_i)$ in which case, commands with smaller *control length* have to be used.

However, being on the Internet, the user experience is largely depended on the status of the Internet services specifically bandwidth and response time. While sending and receiving commands and corresponding data do not involve large bandwidth requirements, latency between the user and the experiment is very important. This becomes more obvious in a P2P scenario where the experiments themselves are widely distributed as compared to the "centralised versions". Thus depending upon the current condition of the Internet and the capacities of the device being used by the users, an experiment may be scaled down to a lower interactivity session involving composite commands.

To support multiple levels of interactivity of the same experiment alternative user interfaces are required. Each UI will have corresponding components that invoke respective command sets. The CI has to select which UI or UI components can be active during the experiment session. This change in interactivity levels can happen as a single initial change or multiple dynamic changes.

For a *single initial change*, the network condition is determined at the beginning of the session and the relevant parameters are set for the entire duration of the experiment. The default interactivity level of the experiment may be downgraded if required. This however may not be suitable if the experiment sessions are long. This will require the CI to select only one UI at the beginning of the session.

For *multiple dynamic changes* the network condition is periodically checked and the interactivity levels are periodically updated as well. This however creates an irregular flow in the experiment session possibly interrupting the learning experience. This will also require the CI to select between multiple UIs or UI components that are valid at different points of time in the experiment session.

In changing the interactivity levels, some of the milestone state or goal states may not be attainable, but at least some of them will be achieved. Obviously, with better Internet connection, more of the goal states may be achieved. However, to implement the changing interactivity levels, the corresponding composite commands must be stored on the CUs for it to be able to parse them. This can be done in two ways.

One way is that the maker may create the different levels of commands explicitly so that the UI can fall back to the interactivity level required. This enables fine-tuned operations to be executed on the CU, but requires expertise from makers.

Alternatively, the CU may be intelligent enough to determine the chain of commands that are executed repeatedly and store them as *functions* in consultation with the makers. This restricts the function's capabilities to the intelligent capabilities of the CU, but allows automatic identification which is easy for makers.

The clustering mechanism described earlier can provide for automatic detection of some of the repeating sequence and store them as functions. With it, multiple levels of commands for different interactivity may be stored in the CU and used accordingly. This may be done for if network conditions become unsuitable for highly interactive

experiments. For such experiments the time gap between the commands executed are of high importance with respect to learning experience. But this may not be maintained with high latency. It is necessary to maintain acceptable quality of learning experience while ensuring the safety and integrity of the rig, if the network conditions are bad.

Also, the maker of the experiment may deem an experiment to be run in different ways for different learning outcomes for particular users' condition. In that case they can specifically create the composite commands at desired levels and associate them with different CI or UI.

8.3 Identifying Functions Automatically

The clustering algorithm in *CHAC( )* may be used to determine the set of commands that are executed repeatedly on an experimental rig. The clustering algorithm works by joining closely executed commands into groups. The clustering algorithm takes two inputs ε and γ. ε represents the maximum time gap between the commands execution and γ represents the maximum size of the cluster.

The clustering algorithm can be used to identify closely related commands if the commands are considered individually along with any parameters. This algorithm works on the maker interaction (or possibly other previous experiment sessions) with the rig as training data (*D*). This interaction which is a set (*D*) of commands according to a timeline is used as training data as shown in Figure 8.2. Note that the latency at the makers' side is zero and thus the CI-CU interaction and performance is optimal.

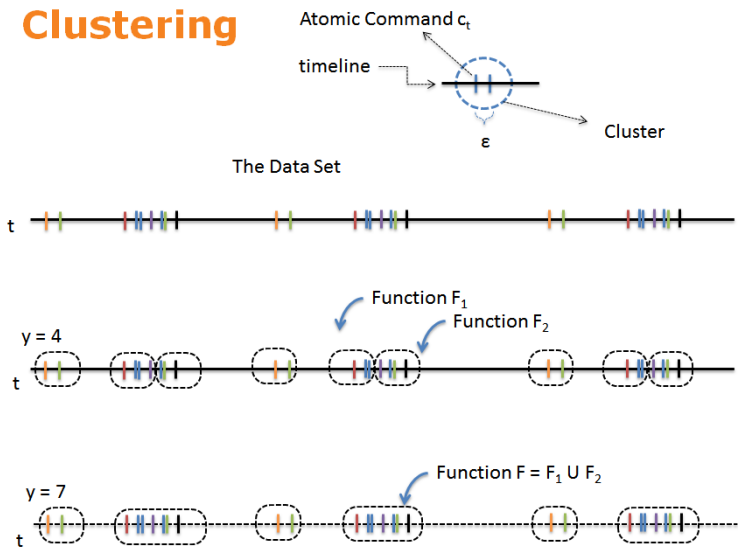


Figure. 8.2 Clustering the repeating set of commands.

Thus for different values of ε and γ different sized clusters may be obtained. Figure 8.2 shows three levels of changing γ with a fixed value for ε.

By incrementally increasing the values for γ for a given ε, several repeating sequences of commands may be identified. γ is kept increasing until no new cluster can be formed and there is a convergence. The CU then calculates which clusters are most repeated during an experiment session. Once these clusters have been identified, the commands can be aggregated into a newly named function.

The value of ε needs to be set appropriately for an experiment. A smaller value for ε will create smaller clusters thus representing a more interactive communication compared to a larger value for ε which will encompass a larger number of commands into a cluster which represents more composite commands.

Thus, the clustering mechanism may be used to obtain composite commands at various levels. For example (in Figure 8.2) with γ = 4, two individual functions F1 and F2 are identified. These two functions maybe stored in the CU and called accordingly. Invoking F1 and F2 guarantees that the constituent commands f each functions is executed in order with time gaps. However, at γ = 4, it is not guaranteed that the time gap between F1 and F2 can be maintained with an adverse network condition. Thus with γ = 7, a larger composite command is created by combining F1 and F2, which will guarantee that the commands are all executed in time. This however reduces interactivity as F1 and F2 cannot be executed independently, thus prohibiting any goal state that may be associated within these functions. This parameter γ may be called the *interactivity* level.

The clustering algorithm here requires a *closely coupled* approach of aggregating lower level commands as described in Section 7.1. This means that the commands repeated and accumulated into a cluster must be strictly part of a higher level command. The mathematical representation in Section 7.1 can now include the type and parameter of the commands passed, into account while clustering. This means that

$$j_v \in \Lambda'(x) \Rightarrow j = WRITE(x, v) \quad or \quad j \in \Lambda'(x) \Rightarrow j = READ(x)$$

for x ∈ R, (set of ports on the CU), where j is a command and $j \in \{\mathbb{r}, \mathbb{w}\}$ for a given parameter v.

This means that a command sequence that is clustered can be

$$J = a_0 a_1 a_2 \ \ldots$$

where $a \in$ R, (set of ports on the CU) and a_i represents the commands with parameter i. Then the clustering can be performed in a similar manner using Λ' instead of Λ as mentioned in Section 7.2.4. This will generate the group of clusters of commands and then the value of θ and Φ for each of them. Once again the list of potential identified commands must have $\theta(J) > 0$ and $\Phi(J) > 0$.

Limitations

This approach of identifying commands however does *not* identify the composite commands with conditional checks. Only composite commands that are repetitive sub-commands with the same parameters can be identified and grouped together. They will also not be able to identify commands that have large time gaps. There can be other methods to identify commands, but they will definitely have a interactivity level associated with them relative to the positon in the *interactivity* continuum.

This method can only identify potential functions, and the maker must approve any function to be saved on the CU or the RLMS services. Whilst storing a function e.g. F1, the maker may be able to add a static input parameter set to the function. The input parameters to a function may alter the subsequently generated atomic instructions from it. But as the function is being executed on the CU, it will always take constant time $T(F1, p_1)$ for a given parameter set p_1.

8.4 Automatically Altering Interactivity

Each experiment can be altered in terms of the rate of commands to become a more batched version of the same experiment by using more composite commands. For this, multiple UIs may be created for each level of interactivity. The UIM can find out the most suitable UI for a given latency and maximize the number of milestone states and goal states covered. The problem may be formulated for a given experiment *e* as

$$G_r = \max|G(e)| \ such\ that$$

$$t(c_{i+1}) - t(c_i) < \Delta \quad \forall \quad c_{i+1}, c_i \in C(g_j) \quad \forall \quad g_j \in G(e)$$

where Δ is a constant and $|G(e)|$ is size of *goal state set G(e)* of *e*.

With the clustering the CU has the ability to identify the functions automatically. The functions are then stored in the CU with the makers' knowledge. Using the various levels of commands will require the users to create different components on the UI or create entirely different UIs for each level of interactivity with different command sets. Next it is important for the CI to determine when to change the level of interactivity.

For a good learning experience, the rate of command execution must stay above a minimum limit corresponding to the current level of interactivity (γ). Thus for a given experiment, the rate of commands

$$k^e_{min} \leq r_e(F1, F2) \leq k^e_{max}$$

where $F1, F2 \in D$ which is the set of all commands in the experiment session timeline. For an experiment using absolutely interactive UI which issues only atomic commands, the rate is expected to be high. The rate however could vary with time as some goal state may require slow rate of inputs compared to others. Thus for a given set of goal states in an experiment, $G = \{g_1, g_2, g_3 ...\}$ that is independent of time, the rate of commands is

$$k^g_{min} \leq r_l(F1, F2) \leq k^g_{max}$$

where $g \in G$ and $F1, F2 \in D_l$ which is the set of all commands in the experiment session timeline corresponding to g. It is difficult to determine what the learners intention is if the components of G are unordered i.e. the learners can choose any change in the rig for a new learning event without following a specific order. If there is an order i.e. G is an ordered set or the current objective of the learner is determinable, then the experiment may act differently for each g_i to maintain the different rates.

In a real time experiment session, it may become difficult to maintain the rate due to network conditions. The *goal state* of an experiment may occur only when a command $C_i = \{a_1, a_2, a_3 ... a_n\}$ with $n \geq 1$ is executed.

Thus, for each individual *goal state* g_i, the minimum rate required can be determined by analysing the makers' interaction with the instruments. For example if F constitutes of two different functions F1 and F2, then the time between F1 and F2 has

to be constant or the functions must be executed at a constant rate. In other words for experiment e,

$$r_e(F1, F2) = T(F1, p_1) + \tau(F1, F2) + T(F2, p_2)$$

where

- $\tau_{min}(F1, F2) < \tau(F1, F2) < \tau_{max}(F1, F2)$ represent the range of time gap between the current function F1 and the all possible next functions F2 as obtained from the training data.
- $T(F1, p_1)$ and $T(F2, p_2)$ are the time taken to execute the function F1 and F2 with any input sets p_1 and p_2 respectively.

$\tau(F1, F2)$ is the only variable that may change with erratic latency. $T(F1, p_1)$ and $T(F2, p_2)$ will be constant for a CU whether they take an inputs p_1 and p_2 that alters the number of instruction or not.

If the value of $\tau(F1, F2)$ cannot be maintained between the ranges as read from the training data i.e. if

$$\tau(F1, F2) > network\ latency$$

for any $F1, F2 \in D$, then the CU and CI automatically change the current rate of interactivity i.e. increase γ and choose to run F.

Assuming an experiment starts with a default value for γ depending upon the default interactivity, the change in the interactivity levels can be done in two ways, as a *single initial change* or *multiple dynamic changes*.

For a *single initial change*, the CI and CU checks the time gap $\tau(F1, F2)$ for all pairs of stored functions and atomic commands $(F1, F2) \in D_y$ corresponding to the current γ from the training data from the makers interaction. For a pair (F1, F2), $\tau(F1, F2)$ may lie within a $\tau_{min}(F1, F2) < \tau(F1, F2) < \tau_{max}(F1, F2)$. If it is found that the value of $\tau(F1, F2)$ may not be held within the range for any $(F1, F2) \in D_y$, the interactive level is dropped i.e. γ is increased by 1 and the time gaps are checked again. This process goes on until a suitable value for γ is obtained where all $\tau(F1, F2)$ for a γ can be within the range as found in the training data. The SIC method only works for a short experiment session, as the latency could change largely over a lengthy period of time.

For *multiple dynamic changes*, the method has to be online. There is a current function F1 being executed at any given time or when a new function F1 is started, the CI and CU checks the time gap τ (F1, F2) for all pairs of stored functions and atomic commands $F1, F2 \in D_y$. If it is found that the value of τ(F1, F2) may not be held within the range for any (F1, F2), the interactivity level is changed i.e. γ is increased by 1 and the time gaps are checked again. This process goes on until a suitable value for γ is obtained where all τ(F1, F2) can be within the range as found in the training data. Alternatively if all the values of τ(F1, F2) are found to be in the range, then γ is decreased and the process is continued until τ (F1, F2) remains in range. Multiple dynamic changes are difficult to implement, as it will require a very dynamic and responsive UI that could handle the change.

Since the training data set is static, the commands in the CU can be associated with a value of γ offline prior to the start of any experiment. The CU does not have to calculate the τ_{min} and τ_{max} online.

8.5 Adaptive Control Interface Example

While the clustering and monitoring of the network latency can automatically switch between interactivity levels, the makers themselves can create the individual composite commands or functions, which can be more sophisticated including conditional checks, for example. This section illustrates the implementation of an experiment as both batched and interactive experiment.

The P2P RAL Programming Platform

The P2P RAL system RALfie, uses SNAP 103 as a programming platform to create the program logic and the UI or CI for the experiments. The program created by the makers may or may not contain functions that are stored on the CU depending upon whether the makers wants the experiment to be absolutely interactive or complete batched. But if the programing platform is able to find repeating sequences of commands or the makers explicitly creates the composite command i.e. functions, they are then stored by the CU. The steps required to create a stored function in the RALfie programming environment are as follows:

1. The makers create an UI and the program logic of the experiment in the SNAP environment on the browser.

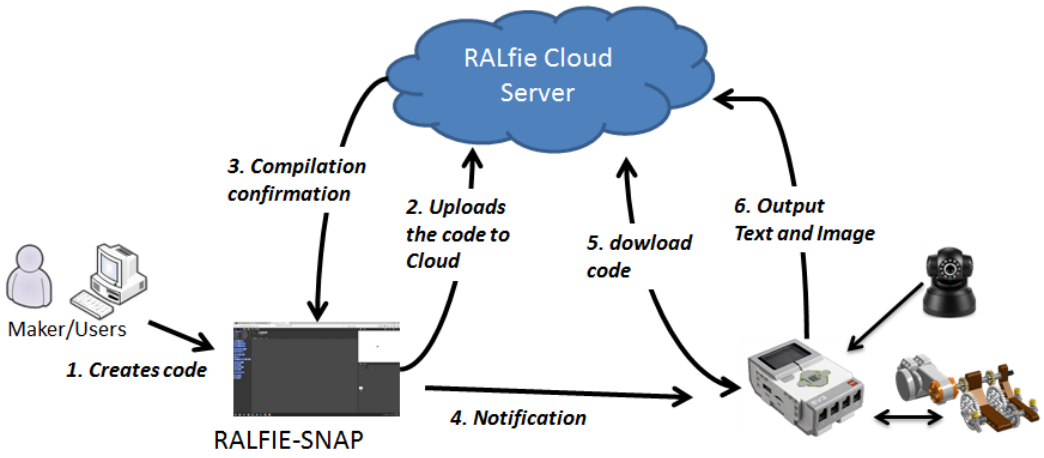


Figure. 8.3. The system architecture to create, compile and upload the code into a CU.

The makers can create exclusive functions for an experiment, or the system itself may identify any function as described earlier.

2. Either way, the corresponding function code is uploaded to the RALfie cloud, which then compiles it corresponding to the target MCU.

3. A compilation confirmation is sent to the SNAP in the maker's browser with the development environment.

4. The maker's browser then notifies the CU to download the compiled function from the cloud and store it locally.

5. The MCU downloads and saves the functions. The makers UI at this point can issue the command though a SNAP block. The MCU runs the command during which the UI may disconnect. Alternatively the maker can improve the code, and re-upload it multiple time to get the desired outcome.

6. Once the function is saved and the experiment is published, the UI used by makers/users can call the functions with a SNAP block which the users cannot alter. When the function is completed, the output file is stored for downloading later.

The SNAP functions are referred by a name as the maker desires and the function is then called from the UI the maker creates. Once the functions starts executing the users/makers can close the connections, but the MCU can still be operating the *function* depending upon the number of the instructions.

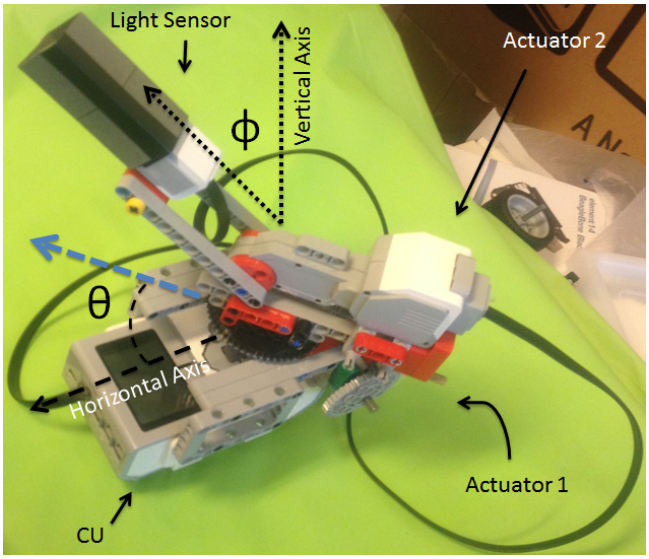


Figure. 8.4. The solar tracker experiment rig

The Solar Tracker Experiment

As an example a solar tracker experiment is discussed. The aim of the experiment is to track the sun's path in the sky during a year. The procedure of the experiment is to find the position of the sun in the sky in terms of θ (horizontal) and φ (vertical). The rig is shown in Figure 8.4. It consists of two actuators to rotate a sensor in the three-dimensional hemi-sphere space. The actuator and sensors can be controlled from the LEGO Brick which is the CU in this experiment. The actuators can also be read to find out the actual degree of rotation.

The variable parameters in this experiment are the values from the sensors reading, i.e.

1. The ambient light intensity return by the sensor (r)

2. The horizontal rotation by actuator 1 (θ)

3. The vertical rotation by actuator 2 (φ)

Thus the learning outcomes for this experiment is the change in the values of the $[r, \theta, \varphi]$. If there is a change in r after changing the θ or φ, then it must be processed and forms a *goal state* (all *milestone states* in this experiment are *goal states)*. Even though the values of r may not change for a given (θ, φ) it is still a goal state. The *goal state* occurs with changes in the actuators.

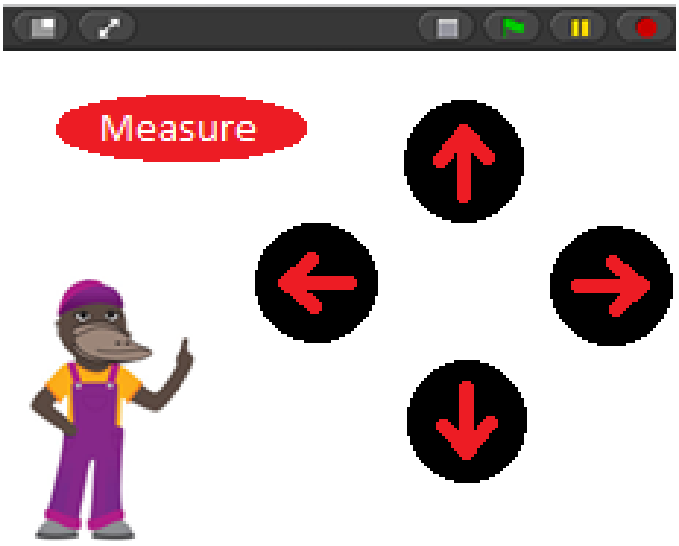


Figure 8.5 An Interactive mode Interface

This experiment may be run with various levels of interactivity as both an *interactive* and *batched* process. These two experiment cases are discussed here:

1. *A absolutely interactive experiment*:

   To run it as an absolutely interactive experiment, the makers can create four buttons representing the unit change in each direction as shown in Figure 8.5. The top and bottom buttons are for moving actuator 1 in opposite direction and the other two provide similar functionality for the actuator 2. Each of the button issues atomic instructions to rotate the corresponding actuator by unit distance, (5 degrees in case of the LEGO motor). The unit distance will depend upon how accurately the hardware can be used. The Actuator 1 is limited to angles between 0 and 355 degrees while the Actuator 2 is limited between 0 and 90. The user presses the button as they desire and reads the value of the sensor. After taking sufficient readings, the user can determine the location of the sun. Each time the user presses a button, there is a new *goal state* regarding this experiment's learning outcomes on the rig that is noted by the user.

2. *As a complete batched experiment*:

   To run it a complete batched experiment, the maker creates specialized functions in SNAP. These functions are converted to the corresponding

programs suitable for the MCU. In this example, the SNAP function is converted to Java and compiled on the RALfie cloud and downloaded into the LEGO Brick. In the case of solar tracker, the following algorithm is implemented explicitly in the function READ_SUN().

Function READ_SUN(a)

$r_b \leftarrow \emptyset$ and temp $\leftarrow \emptyset$

For time from 1 to 11 step 1

Rotate Actuator 1 from 1 to 360 degrees with step 5 degrees

Rotate Actuator 2 from 1 to 90 degree with step 5 degrees

r ← read sensor value

if($r > r_b$)

r_b <- r

temp ← θ, φ

End if

Next Rotation

Next Rotation

Store the highest value r_b with *temp* and time *a*

Wait until an hour has pass since starting to measure

$a \leftarrow a + 1$

End For

The code is then attached to a single component i.e. button in the user interface as shown in Figure 8.6.

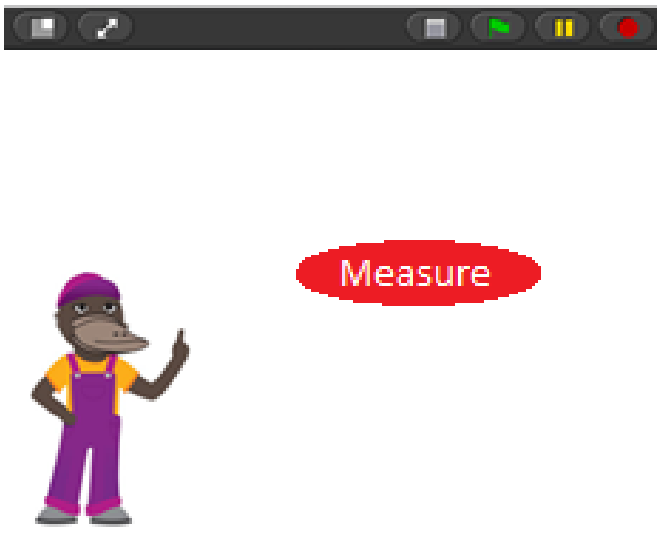


Figure 8.6 An Batched Mode Interface

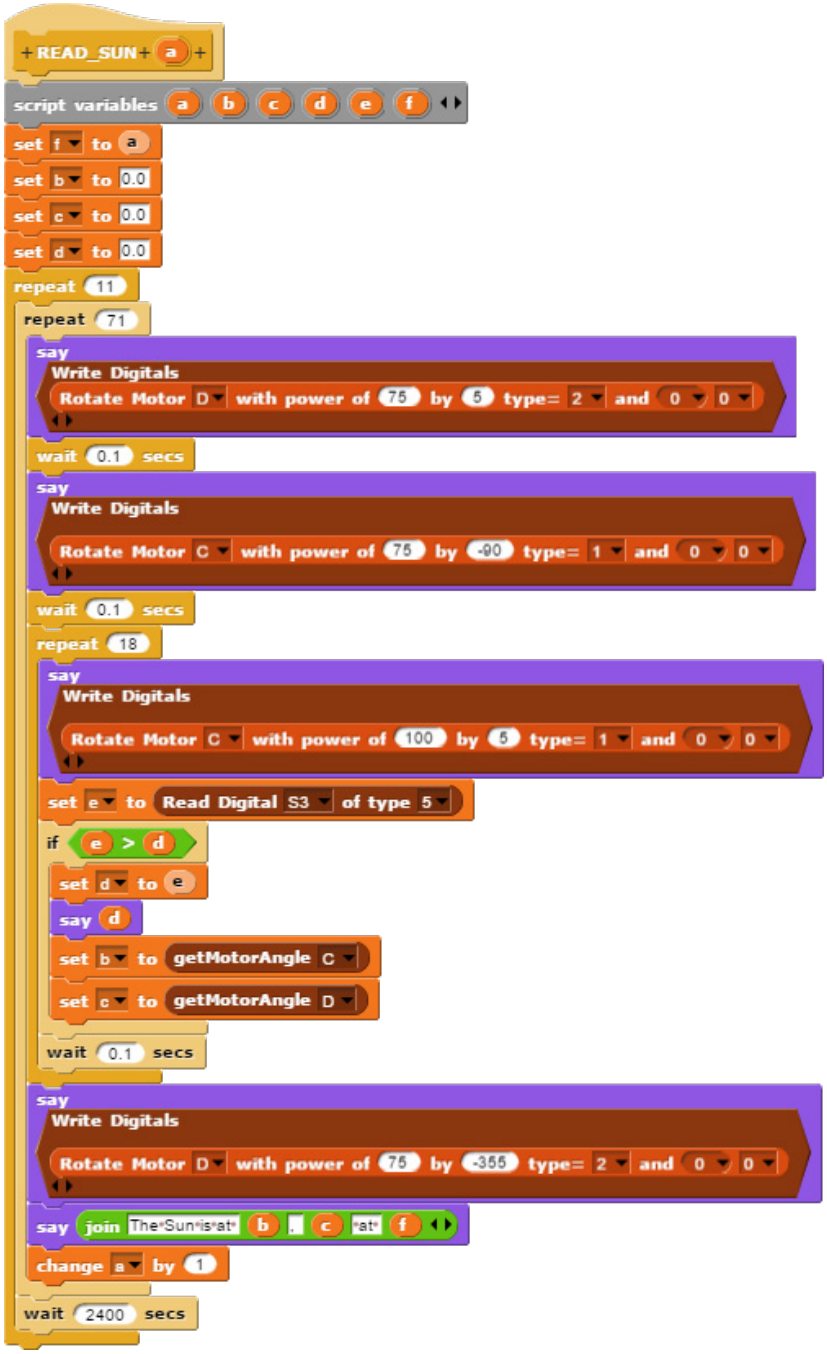


Figure. 8.7. An example of a manually created composite command or function of the solar trackers that is compiled and uploaded to the LEGO

This function runs for 11 hours in a day. After the function is finished the data is extracted from the log files and stored in a readable format. The user can log in at a later time to check the progress of the experiment or get the data after it has completed.

In Figure 8.7, the 'Rotate' blocks are *write* commands and considered atomic. It rotates the motors by the degree specified. Assuming it takes approx. 5 mins to find

the sun (this will depend upon what devices are used) each time, it stores the highest values of the motor angles for each hour represented by variable *f*. The starting hour is passed to the function. The 'say' blocks store the output of its contents into a file on the CU for later references.

Results and Discussions

This example illustrates the *continuum* in the experiment interactivity. With an interactive mode of the experiment, the users have more control over the experiment. The engagement is also higher than a batched experiment. This provides real-time learning activities rather than reading a chart of data after the experiment is complete. Alternatively the batched version allows collecting more data over larger period of time, but engages the experiment for longer periods for one user (or user group) compared to the interactive version.

This example also illustrates the difference between the "absolutely" batched and "complete" batched experiment. The term *complete* and *absolutely* may be used interchangeably, if the level of clustering of commands into function(s) is maximum possible in context of the experiment i.e. how much time or how many commands can be allocated to a user per experiment request. In this case, the students can initiate the READ_SUN function with a parameter they desire which allocates up to 11 hour period to the particular user or user group which is the maximum considered for this experimental setup. Thus the experiment is completely batched as well as absolutely batched for this experiment using the function READ_SUN.

Much more complex functions may be developed that generates a larger number of atomic commands. For example, the entire Function READ_SUN may be put inside another loop that runs the algorithm for 7 days, thus giving the user control of the rig for a week and create a more complete batched experiment. Alternatively, the same user/user group may be allowed to run the experiment 7 times on 7 days, thus generating the same set of commands executed, with much more interaction.

With regards to application in P2P RAL, the maker can build the experiment and run it locally multiple times. If the contents of the function READ_SUN are in the main program of the maker (without it being a specific function), the same set of commands are sent from the CI. As the commands sequentially, the P2P RLMS is

able to identify the cluster of commands that will ultimately resolve to the READ_SUN function. Thus the maker need not explicitly identify the function on the system.

8.6 Summary

The clustering method can also be used to create adaptive user interfaces. This can improve user experience with respect to network conditions involves changing the level of composite commands used dynamically. The concept of an experiment continuum was introduced which shows that experiments can be altered from interactive to batched versions. A Clustering Algorithm was used for determining the level of complexity of commands for selecting the correct UIs for corresponding network latency. The adaptive CI is however applicable for experiments that conforms to the CI-CU model described in earlier Chapters. Experiments that cannot be described with the CI-CU model, may have a different kind of goal state set which may prevent the adaption process.

In the context of IoT or WoT, alternating interfaces could be useful in determining the rate of flow of commands. In a IoT system with time-critical constraints, it is necessary to adapt to network conditions. This would ensure that all commands are executed securely and with respect to time.

The next chapter introduces the RALfie network system and the ways to optimize performance.

9

Enhancing Network Performance

> *This chapter presents the RALfie network system and an overlay network scheme to minimize the network latency between two nodes in the P2P RAL system using the concept of Nano Data Centres.*

For real-time remote laboratories network latency is an important factor in regards to usability and user experience. P2P RAL evolves at least two nodes, the experiment and user site. If no direct connections between the nodes are possible, a relay node is also required. The network architecture for peer-to-peer RAL can be implemented in a number of ways. The simplest way is to setup a TCP socket between the MCU and user. But, whilst TCP sockets are the simplest option, due to different NAT structures in the Internet it requires additional network mechanisms like UDP hole punching or STUN methods, both of which are not guaranteed to work in all environment. The entire system is a P2P network and thus the nodes can be organized with an efficient *overlay network* to minimize the latency between them.

This chapter discusses the P2P network system literature review with respect to RALs in section 9.1. Then it discusses the network setup of a P2P RAL system involving VPN and WebSocket technologies in Sections 9.1 and 9.2. In addition, a method is proposed in Sections 9.3 to 9.6 to minimise latency between network end-nodes. The proposed approach has been tested by simulation to determine relay node locations. The concept of Nano Data Centres (NDC) is used to establish an overlay network scheme.

The content of this chapter is based on **[**108, 174, 175].

9.1 P2P Overlay Networks

A P2P Overlay Network is a computer network built on the top of another computer network. It does not control routing of packets in the underlying network but co-ordinates the communication from outside between its nodes. It allows routing of messages to destinations without mentioning destination IP addresses at the source. A P2P overlay network consists of a set of super peer nodes associated with other peer nodes, which aids in transferring data between nodes that are otherwise incapable of communicating directly. A P2P overlay network dedicated to NAT traversing is discussed in [176]. This approach also selects a random number of relay nodes (super-peers) for every peer. However, no consideration is given to the method of selection and optimizing the distribution of the peers among the super-peers.

Both structured and unstructured overlay P2P systems (e.g. Chord [31], CAN [31], etc.) are designed for quick search and efficient file storage mechanisms for a huge number of files. Also, the data may be divided into separate parts and efficiently stored and distributed to guarantee access to them. However, the RAL system does not need such a storage mechanism, instead peers or super peers are required to share their bandwidth for relaying information. The data exchange is point to point. P2P architectures for distributed laboratories have been proposed in [177]. This allows implementation of distributed experiments that operate through the network. It however does not address the problem of selecting proper routes dynamically such that QoS parameters are optimized.

In RALs, the data is not only exchanged in real-time, but the data itself is generated in real-time i.e. live. There is no scope (or very limited possibility) to determine future actions and states of the client or rig nodes. As such the chunk and cache [178] based approaches which are successfully used in overlay networks for streaming media cannot be used here. This makes it more important that the possibility of peers getting orphaned or left without super-peers be minimized. To do this affectively, the peers must be distributed more widely among the super-peers.

QoS Optimization in Overlay Networks

Clustering based approaches have been used to enhance the QoS of P2P overlay networks [179]. The overlay network scheme tackles triangular inequality violations

of RTT in the internet with an Internet Coordinate System (ICS). Each node requires knowledge about all other nodes through direct measurements between each node or partially estimating network distances thus sacrificing accuracy. Thus the proposed method although reducing RTT is not effective with high churn rates (rate of change in users) [179] due to slow algorithms (for estimation or measurements of RTT). This also requires an ICS algorithm to be run on the nodes which is computationally expensive. None of this is applicable or possible for RALs using low-power MCUs or smart devices in IoT systems.

Real-time multimedia P2P systems (e.g. Skype [12]) that are based on overlay network are used for audio/video streaming between mobile devices. These, although being real-time, can allow for lossy transmissions with techniques like adaptive bitrate [180] and lossy compressions [181] maintaining an acceptable quality. Such systems focus on finding powerful nodes to process and relay information with higher bandwidth. However, with respect to RALs, instrument instructions can neither be compressed (as they are too small and compact, carrying the minimal critical information regardless) nor be lost. Also, the instructions must be executed on a certain timeline on the rig for proper execution.

Thus the relevant P2P network systems that have been proposed are not inherently suitable for the P2P RAL. All of these can be used as the P2P RAL's network system, but they do not address or optimize the specific requirements of the P2P RAL altogether i.e. real-time, but lossless communication in a an *unstructured* network. Thus the real implementation of the RAL system has been kept very simple using a VPN server as described in the next section. This system achieves the transparent direct communication between the CI and CUs.

The mechanism discussed later in this chapter (Section 9.4-9.8) aims to create clusters with the nearest possible NDC in terms of QoS parameters like RTT.

9.2 The P2P RAL - RALfie Network Setup

Due to the types of network connectivity available at the participating sites, not all nodes will have unrestricted access to the Internet. Some nodes will reside on private networks and require Network Address Translation (NAT) to access the Internet. For example, most home networks fall into this category as a home networking router

manages connection between the residents and outer world. There are a number of options to overcome the limitations that are imposed by the widespread use of NATs. Port mapping can be setup either manually or automatically using Universal Plug and Play (UPnP). In corporate environments, nodes are often located behind proxy servers and are not able to run servers. The peer-to-peer community has developed mechanisms to overcome these limitations (for example Skype and JXTA [157]) where the restricted nodes use unrestricted nodes to exchange data between them.

The nature of P2P RAL means that nodes are geographically distributed. Figure 9.1 depicts two maker sites (green clouds on the right) that connect to a central Virtual Private Network (VPN) via User VPN Gateways. The local network that makes up the site connects the computer that is controlling the experiment as well as cameras and other networked equipment. An Access Gateway with an associated user and site database is also connected to the VPN and this gateway is also connected to the public Internet. Two users are shown on the right hand side.

The core of this P2P RAL is a peer-to-peer virtual private network. Previous work has demonstrated that latency is critical [175] and P2P communication is essential for real-time interactive applications [182]. In the context of an Australia-wide system, relay nodes can introduce considerable network delays which results in unacceptable lag for users. If users and makers communicate directly, the effect of network latency

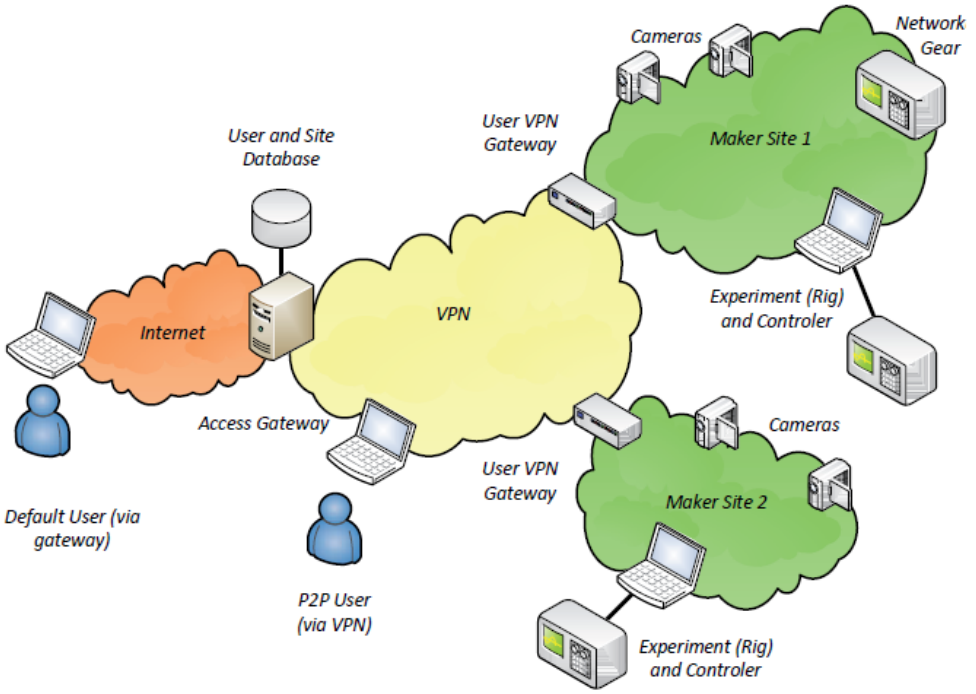


Figure. 9.1. The RAL experiment components.

is minimised. As a secondary effect, access traffic is distributed to many nodes and not concentrated at one particular gateway.

The diagram omits the Internet that is required to setup the VPN between the sites. From the perspective of the devices, i.e. the Access Gateway and the User VPN Gateway, these are connected to the same IP subnet and can be reached directly. At the local maker site, the experiment and other equipment has to be connected to the P2P RAL network. There are potentially two options to achieve this: use existing local networking infrastructure or set up a separate local network. The former requires less additional infrastructure, but potentially means that a third party has access to the local network and the computers that are connected to this network. As security settings for local networks are often permissive, this can be a security concern. This option also requires considerably more configuration and requires the correct setup of the local network. Furthermore, settings will have to be adapted to existing local configurations. Another issue is that both experiments and video feeds need to be authenticated and this has to occur transparently and independent of the networking environment.

To overcome these issues and to make the deployment as seamless as possible, the proposed system uses a separate local network. This also implies a separate IP subnet for maker sites. To ensure the separation between the local private network that provides Internet access and the local experiment network, the experiment are placed in a Demilitarized Zone (DMZ). All participants, including users and makers, need to be authenticated to connect to the P2P RAL system and in essence gain access to nodes connected to the VPN. The Access Gateway maintains both user/maker and experiment databases. It is also the network node that authenticates users.

Whereas authenticating users via a central web gateway is straight forward, authenticating peer-to-peer connections is more complicated. Both users and makers generate a public and private key pair at the time of registration and the public keys are known to the central broker node. Once a peer-to-peer session for a particular user has been authenticated, the broker node has to distribute the respective public keys to the node that are party to the transaction. For the maker site, the key of the user has also a Time to Live (TTL). The session is active, as long as the key at the experiment site is valid. If the session is terminated prematurely, the keys are revoked.

9.3 RALfie Implementation and Further Work

This section discusses specific details of a P2P RAL system that has being developed for the RALfie project.

9.3.1 User VPN Gateway (RALfieBox)

From a participant's perspective, a RALfieBox is the core of the system. Technically this is a User VPN Gateway managing the local network that hosts the experiment, connecting to the VPN overlay and authenticating user access. These systems are based on common low cost home gateway devices and run custom Linux firmware. These devices are preconfigured in a way that automatically joins them to the overlay network once the node is connected to the Internet on the WAN port. The LAN ports are used for RAL appliances such as cameras and controllers. The systems are based on OpenWrt firmware (http://openwrt.org/) and specific hardware is not required.

Makers have no administrative access to the RALfieBox. The WAN port is connected to the Internet and the VPN client establishes one external connection to the VPN note on the Access Gateway. All other incoming traffic on the WAN interface is dropped except incoming VPN connections.

Makers can use the local network to configure the local activity and setup the camera etc. Each device is paired with the RALfieBox. This involves an initial step of identifying the IP Camera and MCU network interface and pairing them to the RALfieBox. But this step needs to be done only once and can be done by an expert before the RALfieBox can be used to create the experiment without further support. All RALfieBoxes use the same local IP subnet. By using the same address space across all local P2P RAL networks configurations, instructions and support are simplified. Makers do not have to deal with IP addresses at any stage.

9.3.2 RALfie Portal and Gateway

The RALfie portal and gateway on the internet (https://ralfie.net) is the main website and the experiment details repository in the system. This portal also contains the gateway that provides the connection between the VPN and the outside world i.e. the Internet. When an users logs into the RALfie portal and selects an experiment, the user id not part of the VPN. Instead the users' connection is established with the portal

which then relays the communication to the corresponding RALfieBox depending upon the experiment selected.

The portal also stores the experiment data that are created for an experiment for example the corresponding CI including UI and CPL and experiment activity details including the aim and descriptions of the experiment. While ideally these should be stored in a distributed manner, the current RALfie version stores this on the portal based on cloud services. The next sections focus on the possible network setup and how the RALfie portal and gateway services could be setup in a completely distributed manner using Nano Data Centres (NDC).

9.3.3 Increasing Network Performance

Simply setting up the RAL network as described above successfully provides the end-to-end connection. But it does not guarantee the best network conditions with regards to network latency. The following sections propose an overlay network based on NDCs that tries to optimise i.e. reduce the network latency as much as possible. At this point the proposed method has not been implemented and has been tested through simulation only.

Nano Data Centres are a new concept of using a large number of low spec computing devices such as *home gateways* to provides services that are normally provided by full-scale data centres, e.g. computing and storage services [183, 184]. Such systems have been studied in the context of content distribution systems like video on demand, for example. On the other hand, highly interactive systems for peer-to-peer applications place stringent requirements on end-to-end QoS metrics such as delay and jitter to be accepted by users. Latency is thus an important aspect of such systems [185]. For interactive real time systems both the source and the destination contribute to delays and its effects. In P2P systems, all nodes are capable of originating and terminating connections. Depending on the connection type, nodes act either as *clients* or *servers*.

This works well for application where the P2P system is concerned only with the peers exchanging data among themselves. For certain systems, such as distributed remote laboratory systems, a number of centralized data storage and content-distribution services are required. Limited by their network access, some peers will

not be able to expose server ports to the Internet easily. Users are often located on private networks accessing the Internet via proxies, firewalls and Network Address Translators (NAT) [186]. In such architectures users can initiate connections to the outside world, but nodes on the outside cannot initiate connections with nodes on local networks. Technical solutions exist to overcome these issues, for example port forwarding; however, these are often prohibited by the network administrators out of security concerns or require technical knowledge to be setup. In this case it is necessary to relay data via additional nodes.

This sections addresses the following problem: Given a set of geographically distributed sites with peer nodes with (a) their probability of using the system, (b) their inability to listen to incoming connections from other peers and (c) the requirement of a supporting *content distribution system*, the aim is to ensure optimal locations of NDCs with respect to geographic routing principles [187, 183]. The Round Trip Time (RTT) between any two peers must be minimized subject to node capability constraints. As end-to-end delay is a critical factor, the geographic location of NDCs is critical in ensuring minimal latencies [175]. This is particularly important as nodes are potentially distributed over a large area.

A clustering approach similar to Section 7.2 is used to group sites into clusters from which a set of NDC sites is selected. The clusters are created according to the system characteristics based on its sensitivity to a QoS parameter - the response time. The re-clustering with respect to time is shown to be adaptive and improves the average system RTT. This is a part of response time by determining the optimal path based on the principles of geographic routing.

9.4 Background and Related Work - NDC and Overlay Networks

This section discussed the literature review of the NDCs and Overlay network. NDCs are normally used to create content delivery networks. For content delivery, a number of replication servers can be setup around the world to minimize latency with respect to geographical location [184]. NDCs allow saving considerable energy and still maintaining scalability. Such systems are more spread apart geographically than conventional data centres and are often larger in numbers to make up for their lower performance. In the context of establishing an end-to-end connection between users, NDCs may play a role in relaying data as well as in addressing communication issues

such as firewalls. The *Traversal Using Relays around NAT* (TURN) [188] protocol uses this principle. A significant impact of protocols like *Interactive Connectivity Establishment* (ICE) [189], STUN, and TURN protocols on delays in operation of P2P *Session Initiation Protocol* is presented in [190]. These factors further necessitate correct positioning of the relay NDC nodes.

A similar overlay architecture, Service Overlay Network (SON), has been described in [191]. It is designed to address point-to-point QoS to facilitate the creation and deployment of Internet based P2P systems. Internet infrastructure supports primarily *best-efforts* connectivity service. The data in a network system from one node to another node typically traverses multiple domains. The focus is on the bandwidth allocation as a major problem of setting up such SONs in [191]. In [192] a balancing strategy has been proposed to overcome the unbalanced data flow distribution in a SON by aiming to achieve system optimization by adapting to the condition of the network. This work aims to create a topology similar to SON based on multiple NDC sites.

Overlay networks based on Distributed Hash Tables is another form of P2P network architecture. DHT based P2P networks are mainly designed for storage and search mechanisms. They are optimised to deal with changing network topologies, as the majority of nodes in the system are unreliable [31]. Other P2P approaches can also identify a set of super-peers among a set of peers who host certain quantities of content that are then consumed by other peers [31]. In the current context, ensuring the low latency is of utmost importance. Any search and storage mechanism may enhance the system performance as NDC sites (equivalent to super-peers) are identified. Unstructured P2P system allows the peers to join the network without any prior knowledge of the network topology [8]. This type of network uses flooding mechanisms to communicate and locate necessary information. Peer respond to a query with a list of all matching content to its higher level nodes. Despite these systems being computationally in-efficient due to flooding, the unstructured P2P systems such as torrents are the most widely used P2P network system on the internet [8]. This is mainly because of their higher reliability i.e. there are some nodes that are always present and efficiency in practical situations. The NDC mechanism proposed in this thesis follows this pattern of P2P network.

There are some methods used for location-based routing in ad-hoc networks [193, 194]. All nodes have knowledge of their neighbours and in some cases discover distant nodes called anchors [193]. Source nodes pass data to the next available node in the direction of the desired destination. The next receiving node again passes the data to the best suited note in the direction of the destination. Based on these principles, a greedy strategy - the GRA (Geographic Routing Algorithm) is proposed [194]. There have been a number of studies about proper positioning of super-peers in a network [195] that aims at the determination of placement of relay stations of WiMAX systems in different geographical scenarios such as mountains, lakes etc. In the context of this work, the policy of sending data based on distance and geographic position is the main target, although the network is not ad-hoc. The system can however adapt its topology over time.

Geographical Load Balancing [196] is a system with clusters at various locations across the globe that dynamically routes data to such nodes based on proximity to the user, system load at that time and local electricity cost [197]. The proposed methodology in this work can be used to determine NDC site positions and change between them dynamically based on user participation in the system by considering only relative response times. However, other parameters may be incorporated in the clustering algorithm.

The main aim is to minimize the RTT in a real-time system given the problems of node location uncertainty and changing users' probability by determining the positions of the NDCs using actual geographic positions. The desired network characteristics may be centred on two parameters - average system response time and percentage of population covered within a limit [175]. These two parameters may be measured for all the candidate sites to determine the NDC site. For this work, the average system RTT is used as the indicator.

9.5 Basic Overview of the Overlay Network System

The P2P RAL system poses the following problem with the respect to any P2P overlay network system: There are dynamic users who can enter or leave the system at will. The duration of the users being in the system is variable but finite, yet not spontaneous. The probability of the number of users from a certain location is variable and depends on its population and various other factors. The NDCs can host a large

amount of information repositories such the experiment list and corresponding multimedia data including text videos and images related to the experiments. NDCs are also responsible for relaying data between users due to firewall or security issues.

The objective of the overlay network of NDCs is to minimize a certain cost associated with the exchange of information such as average system RTT. This includes configurations where only one NDC node is used as a relay node/site [175] by the entire system, and a set of super-peer NDC nodes used among all peer nodes. The probability of users' participation changes with time and is assumed to be predictable. The input parameters in the procedure to determine the NDC sites are

- a series of sites (Ч) where multiple users' nodes are located
- the changing user nodes participation pattern in them (σ) and
- the distance (considered proportional to the latency) between them (ψ).

The communication is assumed to be one-to-one. The corresponding network architecture consists of several nodes that are situated across a wide geographical area.

The process of finding the NDC sites start with clustering the sites in Ч according to their distance with each other. Each site is placed into clusters such that each of them is within a certain limited distances from all other nodes in their respective clusters. The clustering can change over time as users behaviour changes. These clusters may be used in two scenarios:

> If the participation pattern of the users is cyclic i.e. the change in clustering displays a static pattern, the NDCs may be permanently assigned in determined centres sites of clusters. This way a SON for the system may be implemented.
>
> Otherwise, the clustering has to be done repeatedly and the desired NDCs may be activated when they are suitable and de-activated to conserve energy [183] when there is new set of more suitable NDCs sites i.e. the cluster and the cluster centres change. This requires that each site in Ч at least possess a NDC even if it is de-activated intermittently.

In a practical scenario an NDC site would be data centres or in case of P2P RAL participating schools with dedicated servers.

9.5.1 Estimating System Response Time for QoS

Response time is an important performance factor in real time systems. In a distributed system with multiple peer nodes interacting with each other through one NDC, the system average delay can be estimated by calculating the distance proportional, population weighted average delay [1175]. The average RTT of the system with relay station a in a system of known n nodes (Ψ) with available population data is given by:

$$\Psi_{system}(a, S) = \sum_{i=1}^{n} \sum_{j=1}^{n} P(i,j) \times \psi_a(i,j) \qquad \ldots (9.1)$$

where a is one of the candidate NDC nodes, $P(i, j)$ is the probability of participation between nodes $i, j \in \Psi$ and the $\psi_a(i, j)$ is the RTT between nodes i and j with a as relay centre [175]. The selected optimal NDC site R is then given by:

$$R(S) = \min\{\Psi_{system}(a) \quad : \quad a \in S \qquad \ldots (9.2)$$

9.5.2 Creating Autonomous Peer-to-Peer Overlay Networks

Using just one central relay R_i node for a time period $t_i - t_{i-1}$ can put excessive load on it. It is not very efficient in terms of response time and a peer-to-peer system is more efficient. It is also prone to failure if the central node fails. In order to reduce risk and load on a particular NDC, multiple NDC may be used as stations at any point of time.

Each node is to be associated with a nearest feasible node that can act as a relay and represent it to other nodes in the network. For e.g., if A and B are two sites in clusters C1 and C2 as shown in Figure 9.2, A can communicate with C1 which then relays to C2 and subsequently B. Clustering the available nodes can produce an appropriate set of cluster centres for relay. The cluster centres can independently act as individual relay stations representing all sites within the cluster. There is no single 'dominant' NDC but an overlay network of NDCs that operate in a P2P manner. The system RTT may be calculated as [175]:

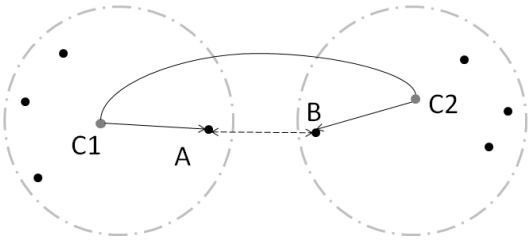


Figure 9.2. An example of cluster regions C1 and C2 at particualr time when users at sites A anbd B are communicating through their respective cluster heads.

$$\Psi_{system} = \sum_{i=1}^{k} \sum_{j=1}^{k} P(i,j) \times \psi(i,j) \quad \ldots (9.3)$$

where *P(i, j)* is the probability of communication between the clusters *i* and *j*.

However, the users' probability of entering and leaving the system is dynamic and changes with time. If user nodes are constantly associated with particular centres such as A with C1 or B with C2, then the system becomes inefficient. In this example, if other nodes in cluster C2 become inactive, A and B, although close to each other will end up with a longer route resulting in higher RTT. Thus the clusters must be re-created with respect to time for obtaining optimized path.

The clustering also changes with time as the probability of the users accessing the system changes or the number of users accessing from a particular site changes over time. This can be done by either calculating the sites in real-time after periodic intervals or may be pre-calculated for a longer period of time if the user pattern (σ) is stable and known before-hand. The output of the algorithm is a vector of NDC site arrays (Γ) from a pool of available sites (Ч) according to time.

$$\Gamma_{\sigma}(S) = \{ Q_1^{t_1}, \ Q_2^{t_2}, \ Q_3^{t_3} \ \ldots \ Q_n^{t_n} \}$$

where $Ч$ is the set of sites, σ is a function representing the change in users' accessibility with time, $0 < t_0 \ldots t_n$ are the points in time when the NDC site array is changed to $Q_i \subset Ч$. Each site $q \in Q_i$ will act as the NDC site for their clusters from that point of time (t_i) to generate the optimal point-to-point response time i.e. Equation (9.3). Additionally, a set of central control NDC nodes $R_i^{t_i}(Q_i^{t_i})$ using Equation 9.2, may be decided among NDC sites in a similar manner according to time t_i.

$$\Pi_{\sigma}(\Gamma) = \{ R_1^{t_1}, \ R_2^{t_2}, \ R_3^{t_3} \ \ldots \ R_n^{t_n} \}$$

9.5.3 Users' Participation Probability

The users' participation probability $\sigma(i)_t$ for a site is the proportionate number of *active* user population from $i \in Ч$ compared to all sites at time *t*. The definition of *active* is the number of people who could log into the system at *t*. This value of $\sigma(i)$ may not be *cyclic*, as users may respond to real world scenarios. However, this proportion is considered to be predictable. One forms of data that can be indicators to σ(i) is *Scheduling*. If a large number of users have booked time with the system for a certain period, this data can give a good estimate of how many people will access the system. However, the users may not use it at that time and thus the $\sigma(i)_t$ only remains an indicator. The time gap (Δt) between the users booking and accessing the system could be small and thus requires the clustering and calculations of the NDC nodes in real time. This is applicable for P2P RAL systems where users are required to follow scheduling of some sort [198].

Overall the $\sigma(i)$ is determined depending upon various factors that could affect the users' behaviour from a particular location. For the calculations in this chapter, post codes and corresponding population data are used as a source of geographic and population data. RTT estimations are based on the distance between two sites. It is a crucial measurement that determines the propagation delay and response time between two sites. There are around 2500 different sites in Australia which are identifiable by their postal code stored in the database and the geographic latitude and longitude of each site is gathered from Google Maps [199]. The population for each postal code is available from the Australian Bureau of Statistics [200] covering 92% of the total population of Australia.

9.6 The Constrained HAC Algorithm

In this section a new Constrained HAC (CHAC) algorithm is presented that limits the size of a cluster by Ω kms.

9.6.1 The cluster diameter limit - Ω

For most network-based information and multimedia systems such as RAL, the resultant RTT is negligible for QoS or performance up to a certain distance (Ω). Hence, to generalize the distance, the sites need to be clustered into groups such that the RTT within a cluster is negligible in regards to the service requirements. To

achieve this, the geographic size of clusters has to be linked to a system performance parameter. The geographic spread of the group must be limited by a value Ω (in kms) determined by [175]:

$$\Omega = \tau\ (c \times v) \qquad \ldots (9.4)$$

where τ is the RTT, c is the speed of light and v (= 0.27) is the geographic propagation delay constant [201]. The effective network capabilities of communication links is dependent on factors like type of fibre, the number of repeater nodes (routers, switches, hubs, etc.), the route of the linking cables, etc. geographic signal propagation rate v was proposed to account for these factors while calculating the propagation delay.

The distance between two sites is calculated using haversine formula as follows:

$$d = R_E \times 2 \times \tan^{-1}\left(\sqrt{\frac{a}{(1-a)}}\right) \qquad \ldots (9.5)$$

where,

$$a = \sin^2(\Delta\phi/2) + \sin^2(\Delta\lambda/2) \times \cos\phi_1 \times \cos\phi_2$$

and ϕ is the latitude, λ is the longitude, $\Delta\phi$ and $\Delta\lambda$ represent the change in the latitude and longitude and R_E is the radius of the earth (= 6371 kms). The RTT between any two sites i and j is when directly linked [175]:

$$\psi(i,j) = r_f \times d_{ij} \ / \ (c \times v)$$

where $r_f \geq 1$ is a small random factor to simulate the real environment. Ω is the limit for the diameter of any cluster. The value for Ω will depend on the system it is applied to.

9.6.2 CHAC2

This algorithm is similar to the CHAC in Section 7.2, but has only one parameter corresponding $\boldsymbol{\Omega}$ to ε. In general, HAC creates a tree structure called dendo-gram. At each level of the dendo-gram a new cluster is created by merging two lower level clusters. If the dendo-gram tree is cut at one particular level, the nodes in that level represent the resultant clusters. However, if the tree is cut in such way then the intra

cluster distance is not taken into account. To obtain clusters of limited size, the tree has to be pruned at different levels starting at different nodes. In the current context the condition is that Clusters C_1 and C_2 are merged only if all pairs of node-node distances are less than the limit i.e.

$$i \in C_k \quad \nexists \quad i,j \in C_1 \cup C_2 \quad : \quad d_{ij} \geq \Omega \qquad \ldots (9.6)$$

where C_k is the merged cluster and $i \in C_1 \cup C_2$. This step increases the computational complexity of the algorithm, but it still remains in the order of $O(n^2 \log n)$ using priority queues. The centroid, center or head ($q \in Q_i$) of each cluster (C_i) is determined by averaging the X (longitudes) and Y (latitude) axis position of the sites located within the cluster (average linking).

To further increase the efficiency of the algorithm, a near-node list is kept for each cluster. The near-node list stores all nodes that are within the distance of Ω from the centre of the cluster. This near node list is kept because any cluster cannot be joined with another cluster at a distance greater than Ω. When a cluster is created from two clusters then its centre is created by averaging the nodes of the constituents' clusters on both axes.

The near-node list is updated by inserting all the nodes from two clusters into that of the new one. Nodes must not be repeated in the list so before inserting it has to be checked whether the node already exists in the other's list. Also a new node may be inserted into the list only if its distance from the new centre is less than Ω. The CHAC2 algorithm is as follows:

Algorithm 9.1 CHAC2(Ω)

Initially each site $S_1 \ldots S_n \in Ч$ is in its own cluster $C_1 \ldots C_n$
allsitesclustered ← false
prepare near-node list L_i for all clusters C_i
While allsitesclustered = false
    For each C_i search through the L_i to find the pair of clusters C_i, C_j with minimum distance i.e. min{d(C_i, C_j)}
        If d(C_i, C_j) < Ω //according to Equation .9.6
            Join C_j to C_i and remove C_j
            Join Li and Lj and remove L_j

If all pairs of site (C_i, C_j) > Ω then

allsitesclustered ← true

End while

This algorithm is not distributed and can only be done on a single computer. It is designed to determine the set of NDC sites i.e. the cluster centres (Q_i) from a much larger set of (postal code) sites. These NDC sites are used for routing. It may not be optimal and marginally improves on the HAC algorithm, but is able to produce the required type of clusters based on Ω.

9.6.3 Clustering Analysis

Figure 9.3 shows the clustering of sites with Ω = 700km. There are 29 clusters (or NDCs) with an average diameter of 479 kms. The population as shown in Figure 9.4 is very unevenly distributed preserving the original population distribution but this also means there is disparity in probability of communication load from a cluster. The numbers of sites within clusters change even more drastically as there are remote places in Australia which by them becomes a cluster. The cluster diameter (Ω) was changed from 50 km to 2000 km. The number of clusters reduces drastically from above 300 to less than 10 (see Figure. 9.4). This means that to maintain the same quality of service more number of NDC sites in higher density must be setup for lower values of Ω i.e. the system is more sensitive to the QoS Parameter.

The central NDC sites $R_{\Omega}(Ч)$ were determined by using the Equation 9.2. The average system RTT was calculated with a random factor r_f = 1.1 to simulate the real environment. But the average system RTT remains almost the same for the all the selections (see Figure. 9.5) with a standard deviation of 3.28 ms and average RTT of 63.75 ms. This confirms that the clustering can effectively partition the set of NDC sites as required by the value of Ω to determine the positions of the NDCs, but the does not affect the outcome of the RTT estimation. There are changes in the position (ΔR_{Ω}) of the NDC station (see Figure 9.6), as the relay position changes are less than 500 kms for each value of Ω below 1000 kms although, the centre changes abruptly after that (see Figure. 9.7a). Note that inside a cluster the RTT is considered as negligible and the RTTs calculated only account for the NDC sites at the cluster centres.

9.7 Application and Test Case

The last section described the method to create clusters of sites based upon their proximity with each other in terms of response time between them. Each site has a different population size and the probability of users' joining from them is considered directly proportional to the population. Geographic routing principles [194] aim to deliver packets or data governed at least partially by the geographic data of source, destination and intermediate nodes. In this clustering approach, the cluster regions may be used in two ways:

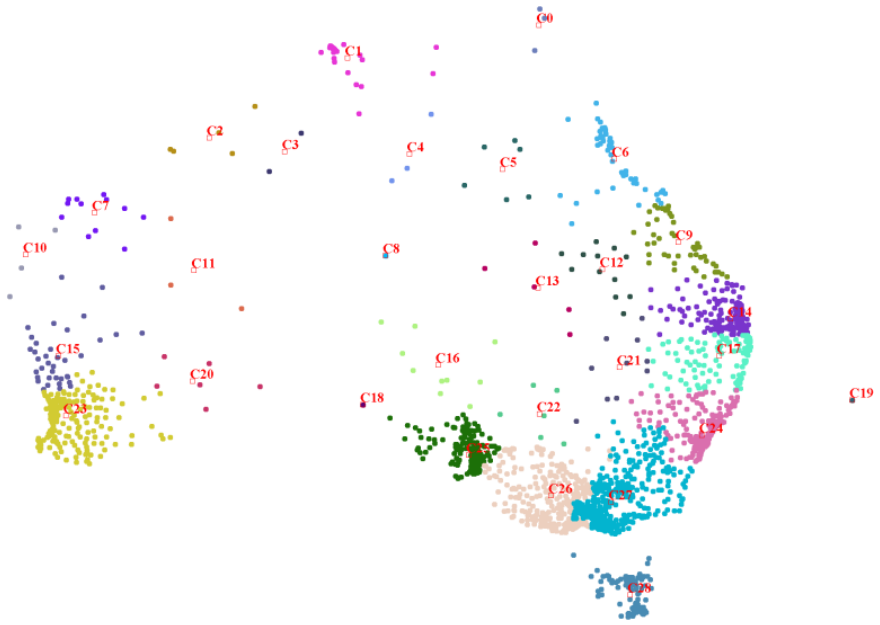


Figure 9.3. CHAC2 Clustering with Ω = 700 gives a total of 29 clusters.

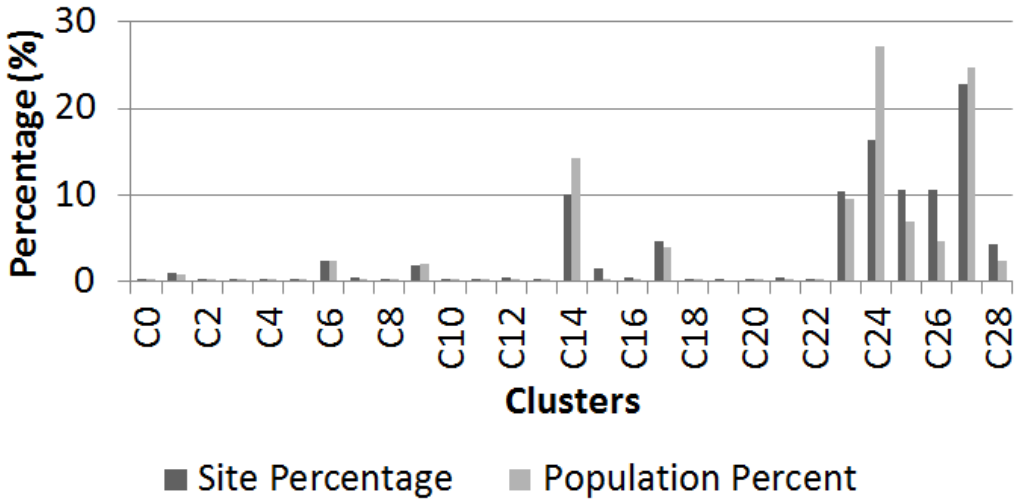


Figure 9.4. Site and Population distribution in the 29 clusters. The population and cluster size (in terms of number of sites) percentage for each cluster.

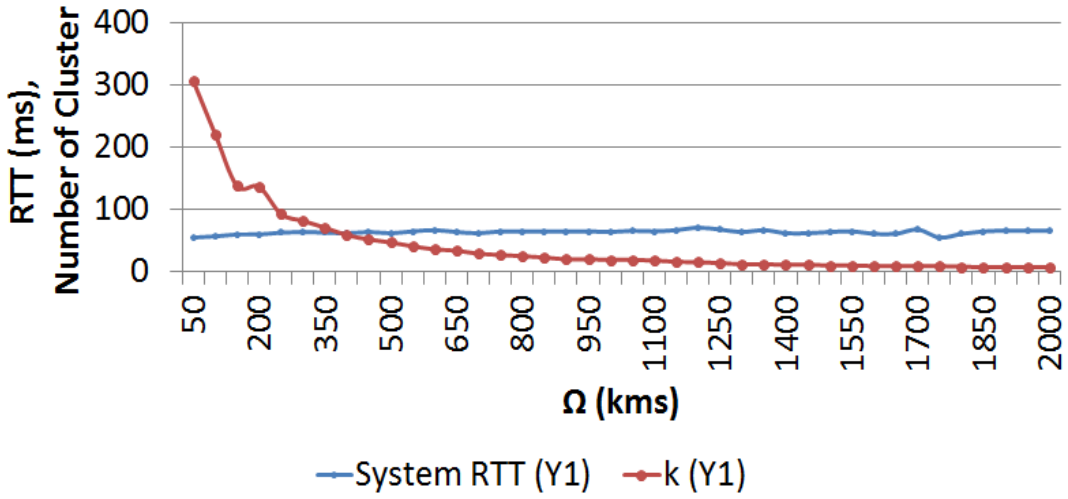


Fig 9.5. The Number of clusters and the average system RTT when the cluster diameter is changed from 50 kms to 2000 kms (step = 50 kms).

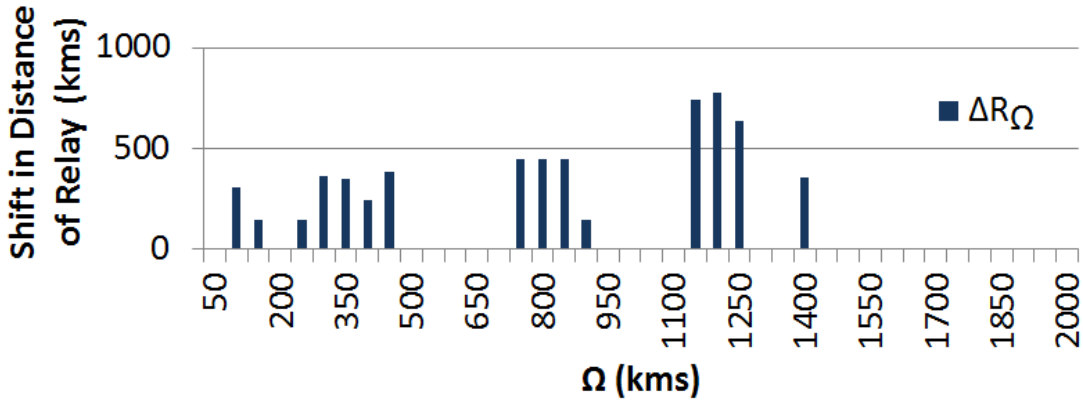


Fig 9.6. Change in position of the central NDC site when the cluster diameter is changed from 50 kms to 2000 kms (step = 50 kms).

If the system needs one single operating central NDC, such as for authentication, introductory look-up tracker node etc., the clusters heads may be considered as representative sites for all sites with the cluster and the NDC may be determined.

If the system operates with a series of NDCs operating in a P2P manner, the cluster heads $q \in Q_i$ can become the *local* NDC node for each cluster and each of them relays the data on behalf of their respective clusters' nodes.

A system may constitute both kinds of architectures simultaneously. An introductory look-up node or tracker node (e.g. such as in a P2P torrent) is required where any peer will first make their query. This introductory look-up introduces the querying peer node to the system. It keeps track of the users' location and participations, uses the CHAC2 algorithm to create clusters and the current list of cluster heads (Q_i) and the sites in cluster $C_i \in Q_i$ they could cover in Ω limit. If any user from the site $s \in C_i$

joins the system, it is assigned to the cluster head node hosting the local NDC.

9.7.1 Test Case Population Participation Function

The population and site data may vary with time. In this case study, the following scenario is considered to illustrate the use of σ in generating Γ and Π.

Considering there is a two hours' time difference between the east and west coast of Australia. This means it may be expected that at some point in time there will be more users at the east coast than the west coast and at another point of time it will be vice versa. This factor changes the probability of users from a particular site with time. The system must adjust relay locations accordingly in regular interval of time to minimize the RTT for the users at that particular time period. The longitude is used here to present a generic case where no sufficient additional data is available on a parameter (other than time) upon which the population distribution may be dependent. It also illustrates a scenario where a cluster may be spread over multiple zones. Time data is however available for every location and can be directly used for this purpose instead of longitude difference.

In its simplest implementation it may be assumed that the users will only start to use the system during the day time of 7 AM to 9 PM. Hence, if a user's or the site's time is outside this range, then the probability of the site in the system is assumed to be zero. This way the number of users will initially increase and as the day progresses the number of users and their geographic spread will reach a saturated level and continue for the day. At the end of the day, once again the users' numbers will start decreasing.

The user's probability in an active site of joining the system may change throughout the day, such as, it is more probable for the users to join the system as the day progresses during the later hours in the afternoon and evening and finally decreasing sharply at night. The users' participation function is given by:

$$\sigma(i) = \begin{cases} T_S \leq T(i) \leq T_E & P(i) \times sin(\theta) \\ Otherwise & 0 \end{cases} \quad \ldots (9.7)$$

where

$$\theta = (T(i) - T_S) \times \frac{\pi}{(T_E - T_s)} \quad \ldots (9.8)$$

where, T_S, T_E are constant starting and ending timelines for the users and *T(i)* is the current time for the site *i*. $\sin\theta$ here returns a probability factor that is multiplied with the population (*P(i)*) to obtain an actual population ratio [175] representing the users' participation probability of the site *i* at time *T(i)*. This is obtained for individual sites within a cluster and all such values are added to get the total population ratio value of the cluster as a cluster may be spread over multiple time zones. The participation probability $P(C_j, C_k)_t$ between two clusters C_j and C_k is then given by [175]:

$$P\left(C_j, C_k\right)_t = \frac{P(C_j) \cdot P(C_k)}{{P_t}^2}$$

where P_t is the total active population across all cluster at time *t*. $P(C_j, C_k)_t$ is used in Equation 9.1-9.3 to obtain *R* and average Ψ.

9.7.2 Determining the NDC Sites

For a system where the users' participation function (σ) is not cyclic the central NDC tracker node is required to keep track of the changing conditions in the network topology. The tracker node performs the following tasks at time $t_i > 0$ with a list of sites $Ч$ and a matrix of distance/response time between them:

1. *Calculate any change in the users' participation probability and* $\sigma_{t_i}(j)$ *for all sites* $j \in Ч$.
2. *After periodic intervals Δt execute the CHAC2 algorithm on the current user sites where the user probability* $\sigma_{t_i}(j) > 0$.
3. *Store the entire newly formed clusters* ($Q^{t_{i+1}}$), *activate the newly determined NDCs* ($q \in Q^{t_{i+1}}$) *and deactivate the old ones* ($q \in Q^{t_i}$) *and assign each site s with its' new local NDC* $q \in Q^{t_{i+1}}$.
4. *Using the NDCs in Γ calculate the central NDC site R' according to Equation 9.2 i.e. R(Γ).*
5. *If a new central NDC* R_{t+1} *is determined, the current central NDC is replaced with the new one.*
6. *If a new user node wants to join the system i.e. makes a join request, it is pointed towards its current local NDC.*

If σ is cyclic as in Equation 9.7-8 which can be repeated for every day and the population of the sites do not change, the sites may be calculated only once and NDC sites can be fixed permanently.

9.7.3 Simulation and Results

The site related data used in this chapter has been discussed earlier in Section 9.5. For the participation function (σ) Equation 9.7-8 is used with *T(i)* derived in Equation 9.9. There are time zones available for each site, but to make it more generic, the Longitude or Y axis of the sites *λ(i)* are taken as the metric to differentiate between times *T(i)* for site *i*. The time period of a site is calculated by taking the current time of western most parts (w_t) as the eastern parts are ahead in time. The starting and ending time lines are assumed to be T_S = 7:00 AM and T_E = 21:00 PM. the iterations are done for w_t = 5 to 21 as the largest difference between time at 7:00AM in east coast is 5:00AM at w_t. The clusters are recalculated every hour (Δt = 1hour) and the time function is as follows:

$$T(i) = \Lambda(i) + w_t \qquad \ldots (9.9)$$

where,

$$\Lambda(i) = \begin{cases} 113 \leq \lambda(i) \leq 133 & 0 \\ 133 < \lambda(i) \leq 140 & 1 \\ 140 < \lambda(i) \leq 160 & 2 \end{cases}$$

where the upper and lower bound are again flexible according to the nature of the system. In this case, it is roughly assumed that the 113^0 E, 133^0 E and 140^0 E are the points of transition. The Λ(i) function can also collect data about the time from a database directly or σ can be obtained from a different source altogether.

Figure 9.7a shows the change in System RTT, number of clusters and position of relay in Euclidean distance. The system RTT is related to the number of clusters which in turn is dependent on the participation function based on time. The value of Ω is considered 700 kms. The central NDC during these times shifts (ΔR) it position across the map (see Figure. 9.7b) with changing probability of participation from each site. The change in central NDC site is not frequent and dependent on the number of clusters and population ratio. The average System RTT adapts with the changing probability of the sites by starting at around 40 ms when most of eastern sites starts to

enter the system. Then with the day progressing, maintains the time of 65 ms as shown in the previous section and finally again drops to 40 ms when the western sites are only participating. The central NDC sites generated are

$$\Pi_\sigma = \{ S_0, S_1, S_2, S_3, S_4 \}$$

The cluster heads in Γ_σ each have an active NDC until the clusters are changed. These NDCs act as a P2P overlay network by intermediating on behalf of their clusters node. Each node gets associated with the best possible centre node such that the communication is the quickest. The maximum total number of clusters and corresponding *active* local NDCs during the day was 29 and the minimum was 10 at 9 PM. The average system RTT recorded with the data being passed through the local cluster NDCs nodes in P2P manner is 39 ms for most of the day. This is considerably

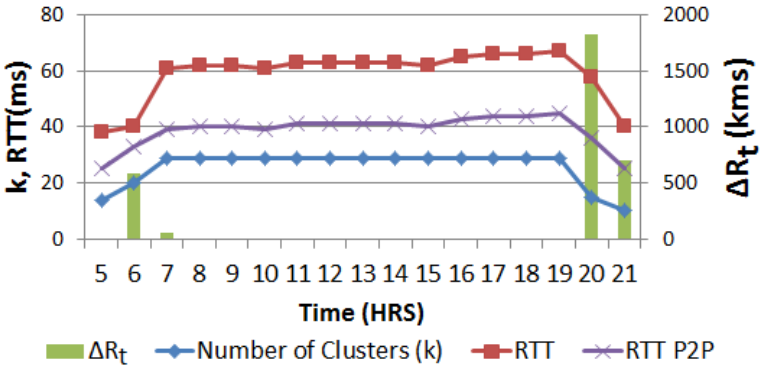


Fig 9.7a. Time shift simulation.

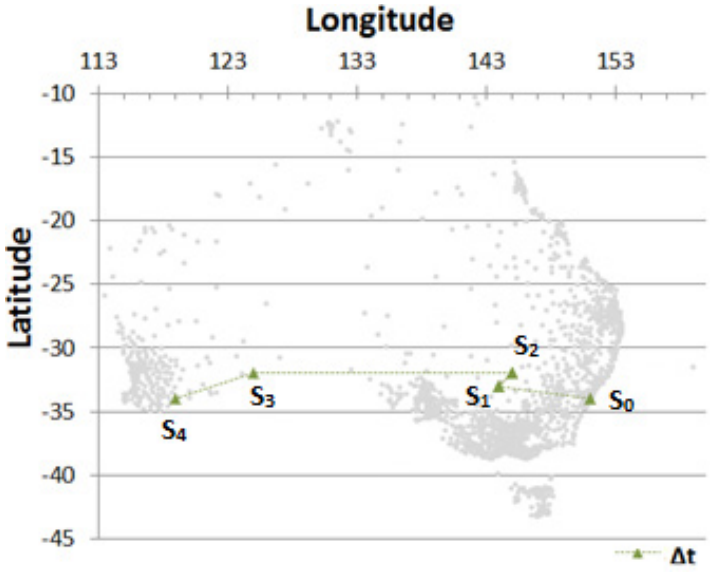


Fig. 9.7b. Geographic transition in the position of the relay. Δt shows the geographic transition according to time shifts

lower that using a single central NDC site for all data exchanges and storage. Note that while calculating RTTs the RTT within a cluster is negligible and only RTT between the cluster centres local NDCs are considered. With a different or more dynamic σ(i), there could be larger number of NDC sites in Π_σ and Γ_σ.

This approach of finding NDC sites for a P2P network does not guarantee the QoS unless σ is very accurate for all times. However, the proper placements lead to overall improvement of the system performance. One limitation of this approach of clustering sites on geographic locations and finding suitable location of NDCs is that it is dependent on the geographic distance between the sites.

9.8 Summary

This chapter discussed the actual implementations of the network setup of the P2P RAL. It uses VPN connections to establish the end-to-end connections between the users and the experiments. For the makers, a RALfieBox is required to setup the VPN connections. The RALfieBox runs VPN software that establishes the end-to-end connection for the users/makers. Each RALfieBox VPN software runs with as unique identification number. The users do not need any kind of special devices ad can access the experiment through the RALfie access gateway on the Internet. The CU and camera(s) are connected to the RALfieBox by the makers. The RALfie RLMS automatically integrates the CU and camera into the system according to the RALfieBox identification number. Thus every experiment is connected to a corresponding RALfieBox number and this is displayed on the RALfie Portal.

Further to the actual RALfie setup, a method to optimize the latency between nodes has been proposed. This method uses the constrained clustering algorithm to create different geographic zones where relay nodes may be located to easiest and quickest access for the makers and users. The proposed scheme of selecting relays first clusters nearby geographic regions into clusters and then select the best regions to host the Nano Data Centres. These selected relay node location(s) can store the experiment data and other RLMS features in the respective Nano Data Centres.

The network connections in the proposed P2P RAL system are helpful for any IoT system with time factors. The proposed method of selecting relay nodes can be used to setup NDCs across a large geographic region for various services to ensure

quickest delivery of data between end nodes. This methodology although focused on a P2P centric architecture can be used for catering centralized services as well.

The next chapter discusses the reliability of the P2P RAL as a whole including the components used for rig creation, network and user capabilities.

10

Reliability

This chapter discusses the design and operational reliability issues of a peer-to-peer remote laboratory and the Web of Things systems.

Involving amateurs in building and operating experiments has its unique challenges. The system is community driven and built; thus people from different backgrounds with varying experience build the experiment. While this can help grow the variety and number of experiments on the system, the experiments themselves may be of lower cost and quality compared to a traditional, centralised RAL system. There are also other technical differences including network [175], types of controller and peripheral devices. As the experiments are physically under the control of the *makers* in isolated locations, there is no centralised monitoring or recovery policy to deal with faulty experiments. Besides, social and other technical challenges, these factors also have direct impact on availability and reliability of experiments in P2P RAL systems.

Reliability of a system is a measure of its availability for use at a particular time [202] and the study of reliability focuses on identifying weak points in a system with a view to improve its availability. Reliability theory is concerned with the statistical nature of failures of devices over their typically useful life. It focuses on random failures, where the failure rates are predictable. 'Random' failures can be attributed to many components within a P2P RAL system. Reliability analysis of the whole system will lead to a better understanding of the P2P RALs operation under load, and present ways to identify the 'weak' links.

A distributed P2P RAL aims to establish collaboration between users and exploit the

individual users' creativity regarding the experimental rigs which impacts reliability. In a P2P configuration makers/users are responsible for their experimental rigs, where the rigs must be created and maintained by them. Stochastically with time, any two pair of users' node, 1 learner and 1 experimental rig will setup a connection and the experiment will be run. Rigs may *not* be kept online 24x7 i.e. there could be certain periods of time when a particular experiment is not available.

The P2P RAL allows multiple makers to host the same experiment with subtly different configurations. This allows virtually an unlimited number of experiments in the system. The system is designed to accommodate rigs as individual modules. Each rig is typically in a different site location and is hence mutually independent of each other, where they can enter or leave the system as the maker desires.

In this chapter, a methodology to determine the reliability of a P2P RAL and similar WoT is presented. It is dependent on three different factors - components and design of experimental rigs, network and users/developers characteristics. The model allows for a comparison of the reliability of distributed and centralised architectures.

The rest of the chapter is organized as follows: a brief description of the reliability analysis modelling is given in Section 10.1.1 to 10.1.3 followed by discussions about the components of the P2P RAL experiments in Section 10.2. The reliability graph and a model to measure the reliability of P2P RAL system is presented in Section 10.3. Section 10.4 shows a comparative analysis between the centralised approach and the P2P system based in this model and Section 10.5 give a generalized description for WoT systems without human factors.

10.1 Related Works Reliability Analysis of Systems

Reliability analysis of large systems is modelled in many ways. Three major types of reliability models are **[**203]:

- *Parts-count models* where only critical components are identified and the failure of any component leads to a complete system failure (e.g. the experimental rigs).
- *Combinatorial models* include creating and analysing fault trees, success trees, and reliability graphs of the system. The overall P2P system is modelled as a combinatorial model in a reliability graph.

- *State-space models* where are possible states of the system generated by any event are taken into account for calculating the reliability.

The reliability modelling process works in a recursive manner by first identifying the major subcomponents of the system. These sub-components may be dependent or connected a serial configuration or parallel configuration [202]. If the subcomponents are simple and uniquely quantifiable then their reliability is taken into account. Otherwise the subcomponent is then further divided into finer subcomponents until a simple component is found.

Reliability for single items is measured in terms of *constant-failure* rates (λ) and *constant-repair* rate (μ). For a given *repairable* item, the measures become *mean time between failure* (MTBF) = 1/ λ and *mean time between repair* (MTBR) = 1/ μ. The reliability is then calculated as [204]

$$\Xi = \frac{\mu}{\lambda + \mu}$$

For non-repairable items, the mean time to failure (MTTF = 1 / λ) is used. Ideally, where a composite chains of devices exists in series, the failure rate of the system can be defined as

$$\lambda_s = \sum \lambda_i \qquad \ldots (10.1)$$

Network Reliability

Network Reliability is another important factor. It is defined in two broad terms [204] – first, the *two terminal reliability* which measures the probability that two nodes in a network system will be able to connect to each other, given the network conditions. Secondly, the *all-terminal reliability* extends this by applying the two terminal reliability for all pairs of nodes in the system thus formulating a result for an entire system of nodes with a given network configuration at a particular time. If broadcasting is allowed, then the reliability must account for 1-to-many connectivity as well. In the current context, the two-terminal reliability is considered.

10.2 RAL Architecture

This section discusses the components of a remote laboratory and the assumptions

made with regards to P2P RAL.

10.2.1 Remote laboratory Sub-components

In order to develop a better understanding of the reliability of the overall peer-to-peer system, individual sub-systems are considered separately. These domains are depicted in Figure 10.1 and include:

- *Electronic or controller sub component* (*A*): These are the peripheral devices or actual pieces of the hardware (sensors and actuators) that either gather data from the environment or produce some action that changes the rig configuration or environmental status.

- *Controller* (*C*): The Controller connects the peripheral devices to the Internet. It also hosts the program logic of the experiments. Controllers range from powerful server PCs [9-12], embedded PCs [205] in ELVIS through to MCUs. These vary in their capabilities and reliability. The smaller, portable and cost effective MCUs are used for the distributed P2P RAL.

- *Network* (*N*): To provide connectivity between nodes a Virtual Private Network (VPN) is setup. This allows the peers to directly communicate with each other over the Internet. With overlay networks [108] there is no central control component and the peer discovery and authentications are done in a distributed manner. Unlike the controller or peripheral devices, the network is not under the control of an individual user.

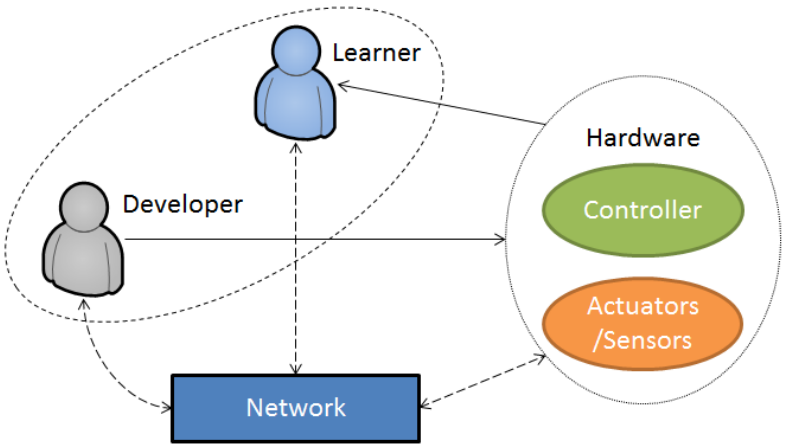


Figure. 10.1. The inter-relationship between the entities of the P2P RAL

- *Users* (*U*): Users are also an important component of the system. For any experiment, there is a developer and the learner who uses it. The developer creates the program logic and the user interface for the experiments. The learner uses the UI for controlling the experiment.

Figure 10.1 shows the relationship between the different components in the P2P RAL system. A Rig ($X = \{C, A\}$) is a composition of a master controller with possibly multiple slave controllers and multiple peripheral devices. If any device $a \in A$ or $c \in C$ fails i.e. stops working, the whole rig X is considered as failed and non-operational.

10.2.2 Operational Assumptions

For modelling purposes, two operational assumptions are made. Operational issues are encountered as a result of running the whole system and the individual rigs. These include the durability of components and the ability to establish a network.

Durability of the component

Each components used in the rig (C, A) is susceptible to wear and tear with time. The longer the rig is used or powered-on, the reliability will correspondingly worsen with time. The components used in the P2P RAL experiments would typically be beyond their initial phase of the `bath-tub curve' [206] of their product life, and hence it is assumed that the reliability will be constant with time.

Network

There are two types of network nodes in the system: peers and super-peers. Not all nodes on the Internet will be able to freely connect to other nodes without the aid of another type of super-node [176] or through the use of technologies such as STUN [157]. This concept forms the basis of an Overlay Networks. Thus reliability in the network is subject to the availability of super-peers. The super-peers nodes are also responsible for search mechanism to find the experiments.

While the operational issues are simple components and can be directly measured in terms of MTTF or availability, the design related issues are dependent on human behaviour and cannot be quantified easily.

10.3 Determining Reliability

This section presents the *reliability graph* and equation for P2P RAL and discusses the components of this graph.

10.3.1 Reliability Graph for P2P RAL

The reliability graph of the P2P RAL system is shown in Figure 10.2. The P2P RAL is composed of three distinct components: makers/developers (*D*), rigs (*C, A*) and the Network (*N*) connected in *serial configuration* [202]. The failure events can be triggered by failure of a maker. There are multiple developers for any given experiments and many developers/makers for many experiments. All of the makers for an experiment have to fail in either creating the rig or making it available in time. This portion itself follows a *parallel configuration* [202].

The next phase is comprised of the rigs themselves. The controllers of the rigs must operate, in addition to each actuator in the experiment. This portion is in a serial configuration, where each of the components in $P(C_i)$ to $P(A_{ni})$ for a rig i must succeed to be able to generate any data.

The final component is the Network. This component is generic in nature. As the P2P RAL system uses the Internet, there are no finite requirements except that the connection must be established between the two ends. There are essentially multiple paths between the experiment nodes where this is also composed of an undefined, but finite, number of routes where at least one is required to succeed.

The reliability of the whole system is depended on the controllers, components,

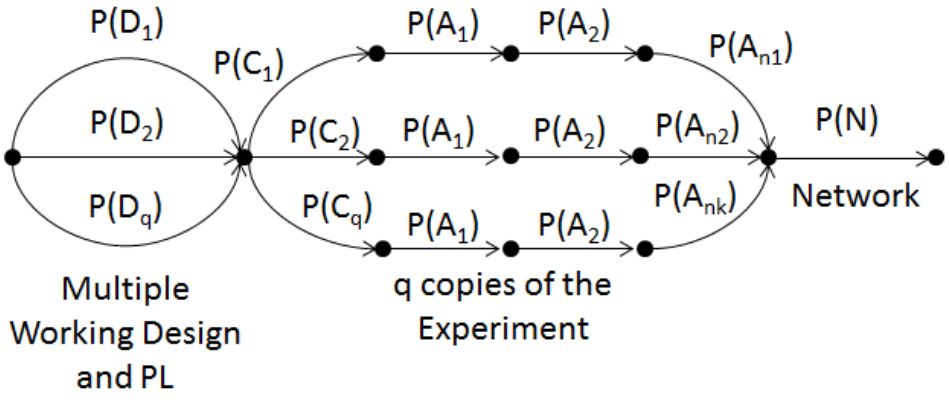


Figure.10.2. The reliability graph of P2P RAL Experiment.

network and the users' reliability. The failure between these are *multiplicative* i.e. for any experiment session to be successful all must succeed. Hence the reliability of a peer-to-peer RAL system with the above components is the probability that a user u at time *t* will be able to successfully perform experiments E.

$$\Xi(u,E)_t = (1 - P(E)_t) \cdot (1 - P(N)_t) \cdot (1 - P(D)_t) \quad \ldots (10.2)$$

where, at time *t*

P(E) is the failure probability of an Experiment E

P(D) is the failure probability of the Makers D

P(N) is the failure probability of the Network N

10.3.2 Experiment Control Reliability

As the rig *X = {C, A}* will fail if just one of the components in X fails, Equation 10.1 does not hold true for P2P RAL. The rig X is repairable but the components C or A is not repairable and their MTTF determines the reliability of X. The reliability of the experiment rigs is dependent on the failure rates of the composing elements. A means to measure and quantify these parameters and variables is derived as follows:

A peripheral device will have a definite *mean time to fail* (MTTF) [207] or expected lifetime (L_A) depending upon the type of the device. Hence, at any given time *t* the probability that the device will fail is

$$P(A)_t = \begin{cases} \dfrac{1}{L_A - T_A^t} & T_A^t < L_A \\ 1 & Otherwise \end{cases} \quad \ldots (10.3)$$

where T_A^t is the time for which A has operated already at time t. Now a rig is composed of a number of components i.e. $1 \leq |A| \leq n$. Hence the probability of a rig (*X*) to fail at any point of time is,

$$P(X)_t = \max_{1 \leq i \leq n} P(A_i)_t \quad \ldots (10.4)$$

Assuming that there are *q* copies of any experiment, the probability that an experiment E is absolutely unavailable i.e. all copies E = $\{X_1, X_2 \ldots X_q\}$ are inaccessible is

$$P(E)_t = \prod_{i=1}^{q} P(X_i)_t \quad \ldots (10.5)$$

This is because each site of the experiment is independent of another. Therefore all of them must be unavailable at any given time for the user to be unable to access that particular experiment.

In case of devices to be used for P2P RAL, the controllers, such as Arduinos, have a high reliability and are used in experimental rigs in centralised versions as well [209]. The peripheral devices used are of lesser reliability and must be used intermittently to maintain their availability for a longer time. With a longer performance time, the operational lifetime of an actuator is reduced due to higher temperature and stress on the components [210]. The typical actuators and sensors used for P2P RAL are consumer grade, constructed of plastic, and mass produced for the educational market, as opposed to industrial grade actuators, typically reserved for commercial automation installation. For example, the same experiment in [83] may also be created for P2P RAL with LEGO based controllers but have to be used at a lower duty ratio to ensure a useful life-span. The reliability of these components can be measured to create a database for monitoring and can also be collected from sources like [211].

10.3.3 Network Reliability

The P2P RAL system not only aims to establish a peer-to-peer collaboration methodology among the users, but actually uses a P2P communication method to establish the connection. P2P is in general more autonomous than centralised systems. For instance, no *system administrator* is required in the P2P system, as each user is responsible for their own node devices and can control their shared resources. However, unlike other P2P systems [31, 157], P2P RAL resources are tangible physical items and unable to be instantly duplicated over the network to ensure availability, in case some nodes fail. Hence in order to maintain availability, multiple copies of the same experiment must be made available. Additionally, if any one particular user is unable to connect to the system, it does not hamper the preposition of the other users.

A P2P system [176], as shown in figure 10.3, has multiple paths available between each node in the network. This ensures that if a particular route (e,g, *a*) is blocked

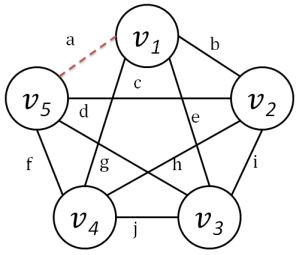


Figure. 10.3. A typical P2P network system.

temporarily from the system, there will be other routes available between any pair of nodes. In a connected graph there exists a path between each pair of nodes. A connected graph of the network is considered as the Internet is the connecting medium. If a node is 'active' i.e. is turned on and able to connect to the Internet, then it can connect to any other node. The service of establishing a connection between the peers (the learner and the maker) is guaranteed, irrespective of the quality of the service.

However, as there are multiple peer nodes in the system with at least some of those behind firewalls or NAT [24, 176], it is possible that despite being able to connect to the Internet, two nodes may not be able to directly connect.

Let *y* be the probability that a user is behind a non-traversable firewall or NAT. Thus the probability that, despite being online, the user cannot connect to the experiment is equal to the probability that both are behind NATs that cannot be traversed, and that no other super-peer is present to mediate between them. Assuming there are *p* numbers of nodes in the system, the probability for each individual one being behind a firewall is *y*. Thus the probability that no nodes are available to be a super peer is

$$P(N)_t = (y)^p \qquad \ldots (10.6)$$

In [176] a survey was conducted of more than 1600 devices, where 25% were unable to traverse the NAT or firewall i.e. y = 0.25. Thus, for a substantially larger number of peers (p > 10), the value for $P(N) \cong 0$. This is based on data from [176] and can be adapted to other applicable results.

10.3.4 User Reliability

Users' reliability *P(D)* is the hardest to determine and quantify. Developer/makers reliability *P(D)* in the system is then dependent on availability or the ability to create

the correct program.

Ensuring accessibility in RALs is a major challenge. RAL is dependent on physical hardware to operate properly. Scheduling algorithms [211] determine when and how a user gets access to an experiment. The P2P RAL can employ an extended time reservation scheduling scheme as discussed in Chapter 13. This allows the makers to setup their experiments at definite periods of time and the users access them during those periods of time slots. This allows the equipment used in the experiment to 'cool-off' maintaining their reliability over longer duration. Other method may include sending reminders to the makers through emails, SMS etc. or implementing additional technologies such as *Wake-on LAN* [212] or other wake up mechanisms that turn on the devices through the network. This can be done through the network ensuring that the nodes are 'alive' when the learner is present in the system.

The second problem is the fundamental issue with a remote laboratory based on user-oriented features. Users may not be able to create a fully-fledged UI or implement correct control program logic. However, the intelligent tools proposed in Chapter 6-8 are capable of ensuring that the rig will be reliably operational even if insufficient for best educational outcomes. Further to that, the moderator in the RAL system can improve the quality of the experiment activities in terms of educational outcomes. A screening process can guarantee a properly functioning UI. The program logic and the rig assembly could still be at risk. But thorough use of the experiment before being published on the Internet could ensure the entry of only 'good' quality and properly functioning rigs in the P2P RAL system.

While these measures can increase the reliability of the P2P RAL system, users' capabilities still remains the weakest link in the reliability chain and is much less reliable compared to the centralised versions.

The developer's reliability is representative of the entire developer population in the system. $P(D)_t$ at any time t, can be thus determined by :

1. The number of sessions until time t where developers have failed to keep their systems online when required (D_f) and the number of experiment sessions (D_s).

2. The ratio of the number of failed experimental rigs (D_r) to the total number of rigs in the system (D_l) until time t. A failed rig due to users can be a result of

incorrectly assembled experimental rigs, or incorrect program logic.

$$P(D)_t = 1 - \left[\left(1 - \frac{D_f}{D_s}\right) \cdot \left(1 - \frac{D_r}{D_l}\right)\right] \quad \ldots (10.7)$$

10.4 Analysis

The Equation 10.2-10.4 and Eq.10. 7 are applicable for centralised RALs as well. Only the network architecture is different where the value for *P(N)* will be calculated differently. A comparison of the centralised and distributed RAL system is described in this section.

10.4.1 Centralised vs P2P Reliability an Example

An example of the way this reliability measurement can be used is illustrated in this section. Table 10.1 shows the assumed values of different parameters. It is considered

TABLE 10.1. Assumed Parameters

Parameters	Centralised	P2P
D_f	1	50
D_r	1	5
D_s	500	500
D_l	50	50
$P(N)_t$	0.00001	N/A
y_t	N/A	0.25
p_t	N/A	10
$P(X)$	0.01	0.05
q	1	1-5

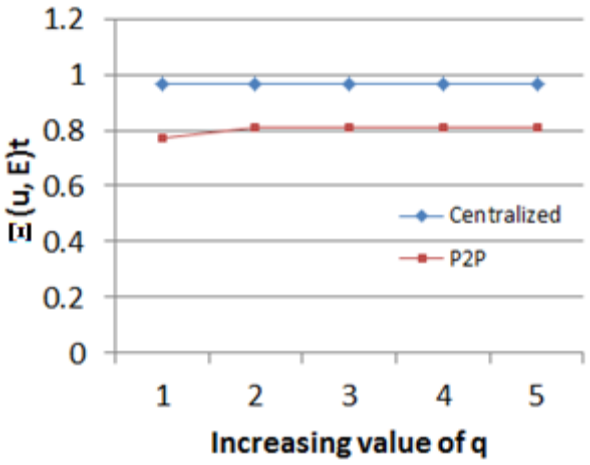


Figure. 10.4. Reliability of the Centralised vs P2P system – an Example

that D_f(Centralised) << D_f(P2P) and D_r(Centralised) << D_r(P2P) as developers are well established and implement industrial techniques, etc. The centralised system is dependent on a single server and as such the failure probability of the network is considered as constant 0.00001. The reliability of the P2P system however depends on the numbers and types of users in the network. The *P(X)* in case of centralised systems is again considered lower than the distributed system due to usage of lower quality devices. The number of nodes in a system at any point of time is considered as 10. The number of sessions (D_s) is considered 10 times of the total number of rigs (D_l) that were available until time *t* are considered same for both.

The value of *q* i.e. the number of copies that an experiment can have in a P2P system was increased from 1 to 5. Note that in the case of centralised system, the value of *q* always remains constant. The reliability for a centralised system is 0.968 with assumed parameters in Table 1. The reliability of the P2P system increases from 0.769 to 0.809 and then stabilizes as shown in Figure 10.4. For any reasonable set of assumptions in Table 1, the reliability of the P2P system will be slightly lower than that of the centralised RAL system. This also indicates the strength of the P2P lies within having multiple copies and owners of the experiments.

10.4.2 Application of the Reliability Analysis

The purposes of the reliability analysis are:

- to detect faults in the P2P RAL system; and
- to prevent them from occurring as much as possible.

From devices to the users, every entity has certain symptoms that can be monitored as part of reliability analysis. When these symptoms are not correct, then recovery measures may be taken. This may involve replacing an actuator, resetting an experiment after it has malfunctioned, finding a new route to the experiments node from the peer node or notifying users about upcoming schedules.

The P2P system is less reliable than the centralised RALs on at least the user related areas. However, the P2P mechanisms are attractive for the following reasons:

1. The over-all cost of building and maintaining this type of RAL is comparatively less.

2. It ensures higher engagement with more 'hands-on experience' for students and collaborations between them.

3. The scalability of such as system is higher with more users joining the system.

10.5 The Case of the WoT

In context of the web of things the human factor can be ignored. In a general web of things system, the impact of human controllers is very limited and the end nodes are not regularly altered. However all other considerations regarding the controllers, components and network are also applicable for any other web of things applications. Such systems contain large numbers of smart end nodes where there is a probability that some of those may fail after some time. Those nodes are responsible for producing collective outputs by cooperating with each other. Failure in any node will decrease the efficiency of the system overall, if not leading to a catastrophic failure.

Thus removing the human factor i.e. P_D, for WoT, the reliability of a master node m operating a slave node s is

$$\Xi(m,s)_t = (1 - P(s)_t) \cdot (1 - P(N)_t)$$

This equation may be used in the same way as illustrated example to determine the reliability and monitor the performance of the WoT system.

10.6 Summary

Like the centralised RALs P2P RAL also offer access from anywhere, and on any type of device. But, one unique characteristic of P2P RAL is that it is intended to be self-sustainable and community driven. Typically centralised systems are monitored by technical staff that ensures that experiments are accessible all the time. Thus ensuring reliability is very important as discussed in this chapter. It is based on three aspects of the system namely the hardware (including the actuators and CUs), the Network and the Users. The P2P RAL can provide similar reliability as the centralised RAL systems in case of Hardware. It can provide better reliability in terms of network being a P2P service. However, being developed by users, it is less reliable in the users' domains with UI creation and rig maintenance. These issues can be dealt with through adequate training and screening of experiments. Reliability analysis can lead to identification of faults, their origin and rectification at the earliest onset.

With the system being constructive and collaborative, there can be higher number, as well as variety, of experiments in the system. While it may be possible to maintain the reliability of these rigs, it next step is to measure and analyse the quality of these experiments to for quality assurance.

The next chapter discusses how all the technologies for P2P RAL described in the earlier chapters can be used in the context of STEM education

11

P2P RAL application in STEM Education

This chapter discusses the integration of STEM education with P2P RAL.

Tools and architecture of the P2P RAL have been discussed in detail in Chapters 4 - 10. This chapter focuses on how P2P RAL is used to enhance the STEM education experience. Science, Technology, Engineering and Mathematics are key subjects in school education that develop skills required to progress into the science and technology related tertiary study and careers. Enquiry-Based Learning (EBL), as well as problem-based and project-based learning, are effective ways to teach STEM in school education [213]. These teaching strategies encourage students to think on their own, work in teams, design solutions and study their effects to gain knowledge and experience of STEM concepts. Generally, these strategies are limited to the local environment at schools. Collaboration between schools to share activities and use them remotely could provide a number if benefits.

RALs can be used to aid in this goal of teaching STEM with EBL providing access for more students to a more diverse range of experiments and creating the opportunity for collaborative networks of students who are using these experiments to share, compare and aggregate data. Previous research has shown that current RAL systems are deficient in features to support STEM education [28]. Most of the RALs are initiated to complement the regular laboratory teaching at universities as a means to increase accessibility to increasing number of enrolled students. Hence activities are designed around services that are provided by universities. Usually these provide a fixed set of experiments that are directly related to the university curriculum. The

experiments are often pre-configured and students have to collect data by changing experimental conditions. Most of these RALs allow little collaboration between students.

While EBL pedagogies has well-established methods, in this chapter, remote laboratory technology described in the previous chapters is merged with EBL methodologies to create an integrated architecture that can support STEM education more efficiently than either of these individually. The limitations of current RAL systems for using it in enquiry-based learning in STEM education are analysed in Section 11.1. A system model for RALs is presented and used to determine the similarity between the RALs and on-site laboratories and determine the areas to expand in Section 11.2.

P2P RAL allows the expansion of the traditional centralised model to a distributed RALs in pedagogic terms. The application of P2P RAL in STEM education is largely facilitated through EBL and the merging of integrating P2P RAL architecture into STEM Education pedagogies is one of the key contributions in this chapter discussed in Section 11.2.

Other key contributions are the specifications for a platform for the UIM that can incorporate language and communication techniques described in Chapter 4 to 9. This leads to the development and implementation of programming tools for creating and hosting rigs (Section 11.3 to 11.5). The P2P RAL employs a quest based learning approach consisting of several game *activities* to engage students built within this programming environment. Several example experiments are presented in Section 11.6 and user and maker feedback is discussed in Section 11.7 respectively. The contents of this chapter are largely based on **[**113, 118, 121].

11.1 Related Work – Pedagogies for RALs in STEM Education

To understand the STEM requirements of RAL systems, an educational model must be used. A comprehensive comparison between the structure and expected learning outcomes of hands-on and remote laboratory has been done in **[**184]. A 4-dimensional model of evaluating a laboratory was suggested. It concluded that compared to on-site laboratories, RALs are similar in two of the four dimensions - developing professional skills and conceptual understanding, a little short in the third - social skills and very poor in case of the fourth - design skills. Another work used university- based remote

experiments to teach physics education in primary schools [215]. This work concluded that there is a relation between students' learning and active participation in experiments. It used a 3-dimensional model by removing the design skills.

An educational system model of RALs in general is presented in this section. RAL systems may be described by analysing the two most basic dimensions mentioned above [215-214]: operational autonomy and pedagogy. Operational autonomy is the scale of technical flexibility offered to the student in an experimental *activity*. It is low when the students have access to only a fixed experiment rig experiment that needs minimal (or no) interaction to get the data; and high when the students can create and alter the experiment conditions to get different results. Pedagogy is the conceptual learning values associated to an experiment i.e. how the experiment is presented and done by the student like enquiry-based learning and project-based learning. A static pedagogy indicates that the RAL experiment replicates the most essential components of learning from the corresponding hands-on experiment and more flexibility in pedagogy implies that RAL experiments are presented in innovative ways taking advantage of ICT for delivery, motivation, and flexibility and student engagement.

Figure 11.1 depicts four quadrants that indicate different levels of operational autonomy and pedagogy. Both of these must go hand in hand and with the increased complexity in pedagogical needs, the complexity and requirements standards of operational autonomy also increase. Current RAL systems offer little flexibility in operational autonomy and associated pedagogy [28]. These are suitable for development of general concepts in higher education where equipment used is

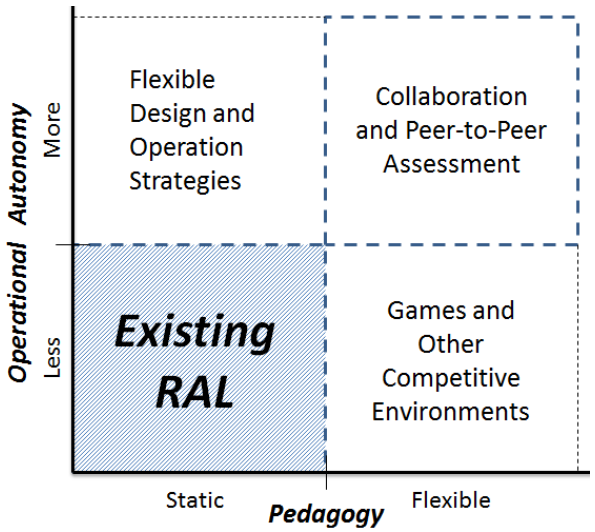


Figure 11.1. The RAL Extension

expensive. Experiments are often measurements generating huge amount of data from some phenomenon within certain conditions. Experiments do not need to be customized according to specific problem sets for users to use them. Users are prohibited from designing rigs. However, in STEM, to understand a concept, one must build, run and see what happens with the experiment. Students may want to share their results with others to get feedback and get new ideas from different perspectives of the same problem. Pedagogical needs of RAL systems are now limited to what is available in a hands-on laboratory as shown in [214]. The concepts are understood as one would read, perform and understand them in an onsite laboratory.

The nature of STEM experiments differs from higher education experiments. STEM experiments may be easily constructed but often creating the rig or setting up the experiment is an important part of the learning experience. This allows the students to better understand concepts and problems related to the activity, which is the main challenge. In higher education laboratories, on the other hand, the equipment is often expensive, proprietary and hard to reproduce. This also means that many experimental setups are static. Users of the experiments are not required to design or build the rigs that support the practical activities.

Enquiry-based learning [216] in STEM aims to make students think and find solutions to a problem by themselves. If this approach is applied in the context of RALs, experiments cannot be reduced to set of instruction. There is a need to present the activity as a problem. Solutions and approaches through which the outcomes are achieved may vary; however, the system has to be able to support students in implementing an experimental rig and related procedures.

11.2 P2P RAL and EBL

This section presents a new methodology to integrate P2P RAL and EBL. The enquiry based learning methodologies encourage students to think of different solutions to a given problem on their own. Usually a given activity produces a question that needs to be answered in order to understand the activity. The cycle of enquiry based learning then follows as: investigation on the topic to find out more details, create a solution typically something physical, observe and record the outcomes and discuss the results among peers. The cycle goes on until the results are perfected to the hypothesis in the *ask* or *investigation* phase. In enquiry-based learning for STEM

activities [217] the following are the most common steps performed by students:

- Given a problem, the first step is to formulate the problem statement that raises questions to the users.
- Prepare a hypothesis of the given problem i.e. what is ideal and most likely?
- Decide on subsequent required experimentations to test the hypothesis.
- Creating an interface that gives proper reflection of the experiment to be performed in a real environment.
- Take measurements and collect data from experiments and analyse them.
- Take cue from other users' results when required, for guidance.
- Teachers are able to facilitate and confirm the correctness of the result obtained.

The EBL stages can be combined with the distributed RALs as follows as shown in Figure 11.2:

1. In the investigation phase after students have gone through the concepts and are ready to make their own design, they can look up in the RAL systems about what others have done.
2. During the create phase they create their own setup to test the hypothesis and use them.
3. Then they use others' system and compare them to find the differences and understand the concepts and improve their own design.
4. Once the setup is finalized, the setup can be put on the RAL system for others

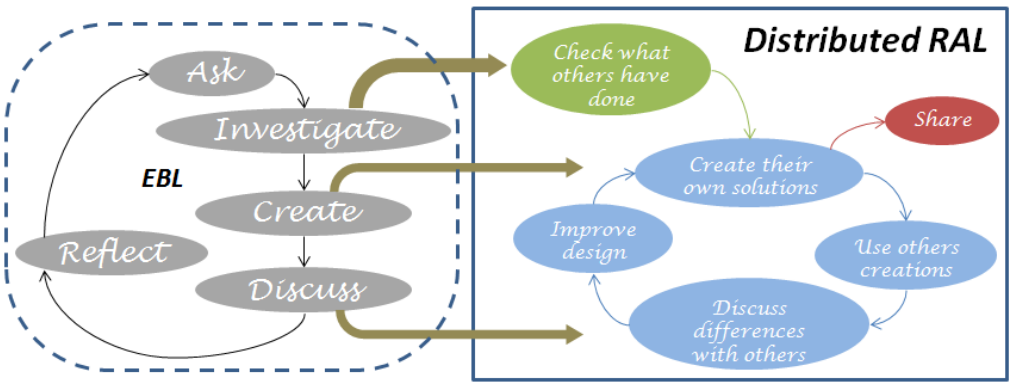


Figure 11.2. The phases of EBL for STEM (left side) extended to include the RAL features (right side)

to use.

Thus, the *create* and *discussion* phases of EBL can be easily incorporated into and improved by the P2P RAL system. Clearly, a simple client-server RAL cannot natively support these kinds of activities and thus both the dimensions i.e. Operational Autonomy and Pedagogy of RAL must be extended.

11.3 *Expanding RALs through a P2P Learning Approach*

Building on the technical P2P RAL method, the P2P approach also works in an educational sense. Students themselves become makers of the experiments and use each other's creations. Remote laboratories can be extended through increasing the scope of design and operation, providing flexibility in organising experiments and collaboration.

When increasing the design scope, students are allowed to plan and design their own experimental rigs for given problems. Designing a STEM experiment includes assembling an equipment setup, programming and running the experiment locally; and sharing the experiment with other students remotely via the Internet.

Students are exposed to several high-end technologies from a young age. So they become capable in learning and using simple electronic devices ranging from programmable robotics like LEGO Mindstorms to mobile phones. These are consumer electronics that are available easily. To run an experiment requires a student to program the different parts of the rig so that they can communicate with each other and the Internet. The only way to put a rig on the Internet is by using a network enabled computing device. This raises the question of student's capability to program a rig.

With computers fast becoming an integral part of our lives, there are several graphical programming languages being taught to young students today such as SNAP, Alice, Tynker and LEGO's LabVIEW based language. These programing languages are able to deliver the same capability of any high level language including multithreading, process communication and programming constructs. These languages can be easily used by students to create their own experiments. The entire scheme gives them full flexibility to think on the problem and come up with their own solutions.

Another problem with traditional RAL infrastructure is the way the experiments are handled. From a user's point of view, the experiments offered are static to-do lists. This limits the RAL system to provide the same capability as that of on-site laboratories. In a distributed form, the RAL experiment list is not static and users can upload whatever they want. As experiments are typically created following a curriculum, the nature of the experiments remains same but the way they are implemented differs from user to user. This provides a competitive or collaborative environment and this platform may be used to develop a game-based RAL pedagogy [218] where students can achieve certain levels and milestones for successfully creating and completing their own and others' experiments. One manner of continuing activities and providing context is through a series of quests [33]. These are a combined group of activities with a greater common goal, and may serve to create a learning path for a more abstract concept to be learnt.

Collaboration is another key aspect of learning. When both dimensions of RAL are increased, there is an added advantage of collaboration between students as peers. As students are running each other's experiments, they are capable of providing feedback on their peer's experiments and learning from each other. When experiments are designed and run collaboratively, these add to the learning outcomes [219, 220]. Students can also help each other by reciprocal teaching as stated in [193] i.e. each student upon completing an activity contributes their experiment and knowledge to the system which is then used by other new students in the system. 3D virtual world technologies have been used successfully to provide a hands-on technology to students via distance collaboration platform [221] and can be used here in a similar manner.

A P2P RAL system implemented can provide the tools and flexibility required for creating a STEM activity is practically available to average users [91]. However, the creation and hosting process have to be standardized according to the automaton models. Apart from constructing a rig, the makers need to create the Control program and the user interface of the experiment. These need to automatically fit into the RLMS. The following sections describe a web-browser based environment for doing this and its requirements.

Before describing in details the RALfie implementations, two notations need to be

defined based on the previous discussions. An experiment *activity* is the learning related materials associated with the experiment. An experiment activity does not contain any hardware but only the software i.e. UI, CPL and corresponding data in form of a *game*. An activity uses an experiments setup to create a learning task called *quest*. When a user performs an activity, they run the UI corresponding to it with the aims of that particular activity. This allows for the same experiments to be used for various purposes.

11.3 Joining Games and Experiments

A stated earlier, each experiment interface is designed as a game based on the concepts of SCRATCH [222]. These games provide an attractive motivation to use the experiments The UI components within this game provides the interaction with the users and makers and collects the inputs. The games and its logic is created and saved using a Web-Browser based environment which is a common platform for all makers and users. This platform implements the P2P RAL technologies and establishes the communication between the UI and the CU.

Computer-based *games* are fundamentally designed for quick, colourful and creative fun and entertainment. Other than entertainment, games have also been used to create environments for the students (players) to acquire knowledge and skills [223]. Gamification of learning environments can take many forms. In context of RALfie a quest-based approach is taken. Students access experiments through quests, which provide context and guidance. The content of quests is presented as a set of instructions and associated resources. It guides the interaction between the students and the UI of the experiment. Quests are organized into hierarchical groups as a larger game-based learning environment [33, 224] where individual users can accomplice bigger goals by completing multiple quests. In addition, experiments themselves can be designed as interactive games.

The creation of experiments by the *makers* involves programming to develop a user interface and to control the experiment. This often involves setting parameters and retrieving data. However, learning programming languages can be challenging for new users as they have their own syntax and semantics to describe complex functionalities.

Several methods have been suggested in previous work to teach programming to young learners using either library based or visual methods [223] which are discussed in the next section.

11.3.1 Related Work – Teaching Programming Languages and Robotics

A computer game has been used as a tool for teaching object-oriented programming methodologies and paradigms in a computer science course in [225]. This was a character based role-playing game where the player's character has to follow a storyline and clear some objectives. In doing so, the character (object) acquires traits (properties) and performs tasks (methods). The player gets experience points or rewards for finishing the given set of objectives. Game oriented procedures have been implementing in STEM fields [226]. Student motivation mainly includes intrinsic goals and tasks of the game.

Natural Language has been used to teach programming fundamentals [227]. It has been shown to be a good alternative to traditional programming languages defined by context free grammar. The natural language although attractive, may not be directly applied to RALs, due to its complex use of ports used to control peripherals. A visual drag and drop language like SCATCH [152] which is a simple language used to teach programming concepts to K12 students is more suitable. The drag enabled programing building blocks allows the pedagogical principles of teaching programming with a low threshold for entry.

Robotics and automation are integral parts of online laboratories. Robotics components are added to a localized version of the experiment setup to make it accessible from remote locations. LEGO based robotics is designed for teaching K12 students about robotics. These have been part of many school based STEM initiatives [228].

RAL programming uses various programming languages although often it is LabVIEW. Pastor et al [229] describe user based custom programming. This approach uses XML to specify the components and the corresponding functions which are then recompiled as Java programs. The students rely on using a XML based Laboratory Experimentation Description Markup Language for creating the laboratory modules and joining them to form experiments. This form of language is not suitable for

STEM students.

The following sections describe new implementation strategies and user related data about the application of P2P RAL in STEM.

11.3.2 P2P RAL Operation

The operation of the proposed P2P maker-learner experimental rig sharing is shown in Figure 11.3. The process starts with the maker identifying/given a STEM problem. Once it is decided on what is to be built, the corresponding experimental setup is prepared. The experimental rig uses automation components such as actuators into the experimental rig that enables its computer-based/remote control. The additional of the automation tools may require minor re-design of the rigs. These two steps are a repeated until a satisfactory control interface and the rig is setup.

Once the setup is ready, it needs to be stored as a *published* experiment in a repository where other users can search them. This storage mechanism id modelled around the quest-based learning [33].

After the experiments are published, it is available to the learners. They run the experiments, collect data and complete activities to gain experience points and collect badges in the quest based system. The creation to publishing affects the users experience with the system in the reverse order –

- Search is affected by storage policy,
- Experiments run and answering the questions is affected by automation and

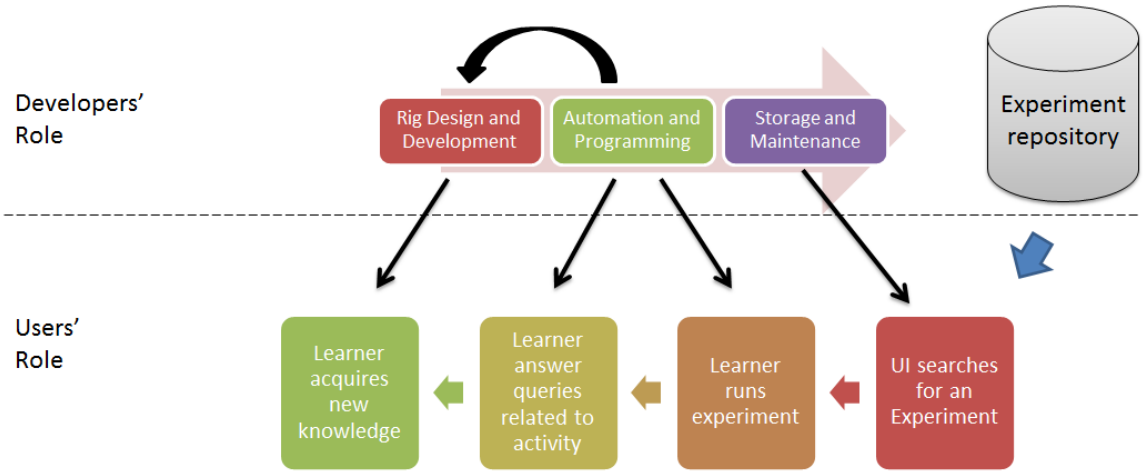


Figure. 11.3. The P2P experiment creation, storage and usage operational steps.

programming of the rig and its interface.

- The learning outcome and ‘game incentives’ in the form of badges, eXperience Points (XP) or achievements gained by the learner is dependent on the type of rig and the experiments chosen by the experiment makers.

In the current context, the focus is on the "Automation and Programming" and “Storage Mechanisms” from a maker’s perspective. In order to provide a unified, consistent, and easily understood programming interface to represent the states of the experimental *activity*, the following Sections 11.4 to 11.5 outlines the requirements of a programming language and supporting technical tools for a P2P RAL environment and evaluates the feasibility of using a graphical languages as the Integrated Development Environment (IDE) to create a Human-Machine interface for experiments.

11.4 P2P RAL Programming and Storage

Proper programming language and development environment must be used to enable users to connect the instruments to the Internet in a homogenous manner.

11.4.1 Role of Programming Language

Once an experimental rig has been assembled, it must be programmed to communicate with the UI through the Internet. From the perspective of young learners programming languages may be divided into several groups.

Procedural vs. Object Oriented Programming

The aspect differentiates between programs that have a simple flow control with programs that associates every data to a conceptual object. Experiments in RALs are usually operated by a small finite set of commands for a session. As such, it should be procedural in operation i.e. the code composed must start and end without initializing any object. Using objects adds higher overhead of associating each function with an object.

Text-Based vs. Visual Languages

This aspect differentiates between the styles of representing language components. A text-based language requires more typing of code, with the associate potential for

errors, while the visual languages are more colourful and primarily uses drag and drop methods. Visual languages are more appealing to the users with less to no programming background [230].

Declarative vs. Imperative Languages

This aspect differentiates between the structures of languages. The declarative strategy specifies the logic of the computation without specifying the manner in which it will be obtained (e.g. SQL). The imperative programming explicitly specifies the line of code. A former is more suitable for teaching young learners but requires high levels of computational flexibility for interpreting the users input.

Hence a declarative, visual and procedural language was chosen for RALfie. For a P2P RAL like RALfie, the fundamental capabilities required for its programming language are:

1. *Iterative and conditional abilities*: These are the two most commonly used programing constructs and needed to write any sort of program.

2. *Data logging abilities*: The language must be able to read and write with a range of sensors and actuators.

3. *Rapid user interface design capabilities*: A GUI and an IDE are also important to easily (re-) configure any program. The visual nature of a program is more appealing to young learners [152]. A GUI allows the users to be more expressive and it provides an easy way for setting up the actual user-interface for the experiment.

4. *Event capturing capabilities*: It must be event oriented. Capturing an event at the user interface and responding to that is vital to a remote laboratory experiment program. Thus events must be clearly defined and a wide variety of events must be supported.

5. *Browse- based*: the language and the corresponding IDE should preferably run in a web browser.

6. *Packaging*: Packaging refers to the capability of creating modular software and re-using code as much as possible. Users may share their codes and designs

with others.

7. *Network capabilities*: Obviously to communicate through the Internet the language must be equipped with the best Internet connectivity features. Note that this feature is not required for RALfie users. The users only create code and run it with the experiment. The underlying network infrastructure is hidden from the actual users of RALfie.

There are multiple graphical languages that fulfil some of these criteria, especially 1-4, like SCRATCH. However, Blockly and SNAP have the additional capabilities of being browser-based and supporting HTML5. They also allow packaging. SNAP has been chosen because of its similarity to SCRATCH which is a wide used language. The network capabilities are not sufficient in SNAP but an additional network module was added for RALfie and thus it forms the basis of the RALfie platform as described in the next section.

11.4.2 Activity as a Game

In order to present the *activity* to the learner, a *quest* is created. A *quest* is basically a game with an objective that must be achieved with in game mechanics provided by the makers. To make the quest interesting and hold the attention of the learners, it is presented as a story. The storyline follows a sequence of interactions between the learner and the interface which leads to a final solution where the interface tells the learner whether the user has reached a correct stage or not.

In case of RALfie a *narrative* approach **[**231] is taken where a character is used to first describe the UI environment i.e. the tools available on screen such as buttons, indicators etc. Then the learner is presented with the quest logic during which they are simply asked for a set of values through a set of questions. The answers to these questions are the input parameters to the experiments. The learners then observe any change in the experiments site through the video feedback or data feedback on the UI. At the end of the quest the learner is presented with quest questions. The answer to these final set of questions lies within the previous interactions with the UI and will indicate the learning outcome of the quest.

11.4.3 Storage in the Content Management System

Once an experiment is created, it must be hosted as part of structured hierarchy so that users are able to search for them and access them in the appropriate sequence. For ease of use and ubiquitous access *Content Management Systems* (CMS) are often associated with RALs. These provide the learning materials and task instructions that give the context for the experiment. Traditionally these would form lessons delivered by a Learning Management System such as Moodle or Blackboard.

In order to increase communication and collaboration between learners, RALfie deploys a non-traditional, gamified approach. Content in RALfie is delivered within a *quest*. In the RALfie system, there are a series of *quests* at different levels that must be completed in series to gain knowledge about a particular topic. One lower level quest may be required for multiple subsequent higher level *quests.* Also, multiple lower level quests may be required to be completed to get access to a higher level *quest*. Learners receive eXperience Points (XP) for completing a quest that accumulates to earn badges that indicate competency. Learners are members of guilds that provide an online learning community. This gamified approach has implications for the design and delivery of content and learning experiences. However, the requirements of the distributed RALs described in this section remain constant whether a traditional lesson structure or a quest-based system is used in relation to a P2P network of user-generated RAL.

11.5 RALfie Implementations

This section presents the technical implementation regarding the programming environment, communication and user feedback for the RALfie.

11.5.1 The Instrument Programming Interface

The system components are shown in Figure 11.4. The backbone of the P2P RAL communication is the VPN or overlay connection between users. Especially designed/programmed routers i.e. RALfieBox connect each experiment node to the VPN. Each experiment setup has one such RALfiebox. One RALfieBox is ideally associated with one controller although it may connect to multiple controllers.

A web-browser based IDE of SNAP is used as the programming interface. SNAP is a

graphical programming interface that allows drag and drop of commands to form the program. The interface is exactly same in syntax and structure as that of SCRATCH . This allows quick understanding of the user interface. The only difference between SNAP and SCARTCH are that SNAP is written in JavaScript allowing it to be executed on any browser. SNAP also allows creating custom blocks which are

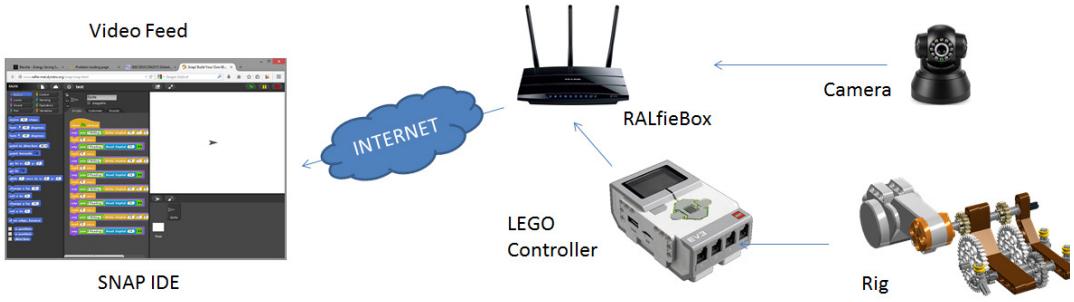


Figure 11.4. The RALfie Communication System Architecture

essentially subroutine or custom functions.

The RALfie re-deploys these tools based on SNAP with the additional requirements of RAL hardware interaction. This adds to new programming paradigms that need to be implemented and used by the makers.

The controllers for the experiments are low-cost microcontrollers units for example, LEGO, Arduino etc. with multiple ports/pins for controlling sensors and actuators. These MCUs have the IEM and associated tools that run the command coming from the SNAP based UI. One controller can potentially run multiple setups that are part of different experiment activities.

The message flow in the system

Figure 11.5 shows the message flow in the system. All experiment CUs when they become online registers with the P2P RLMS. The P2P RLMS assigns an unique ID to each CU and creates corresponding web links or URIs for the experiments. All such experiment are stored in a list in the P2P RLMS. The list is updated in the following events:

1. A CU enters the system: The CU sends a joining message and is recorded into the system list. Initially, all CUs are marked as not engaged in the list.

2. A CU is engaged by a learner: The CU sends a message about their status: *true*

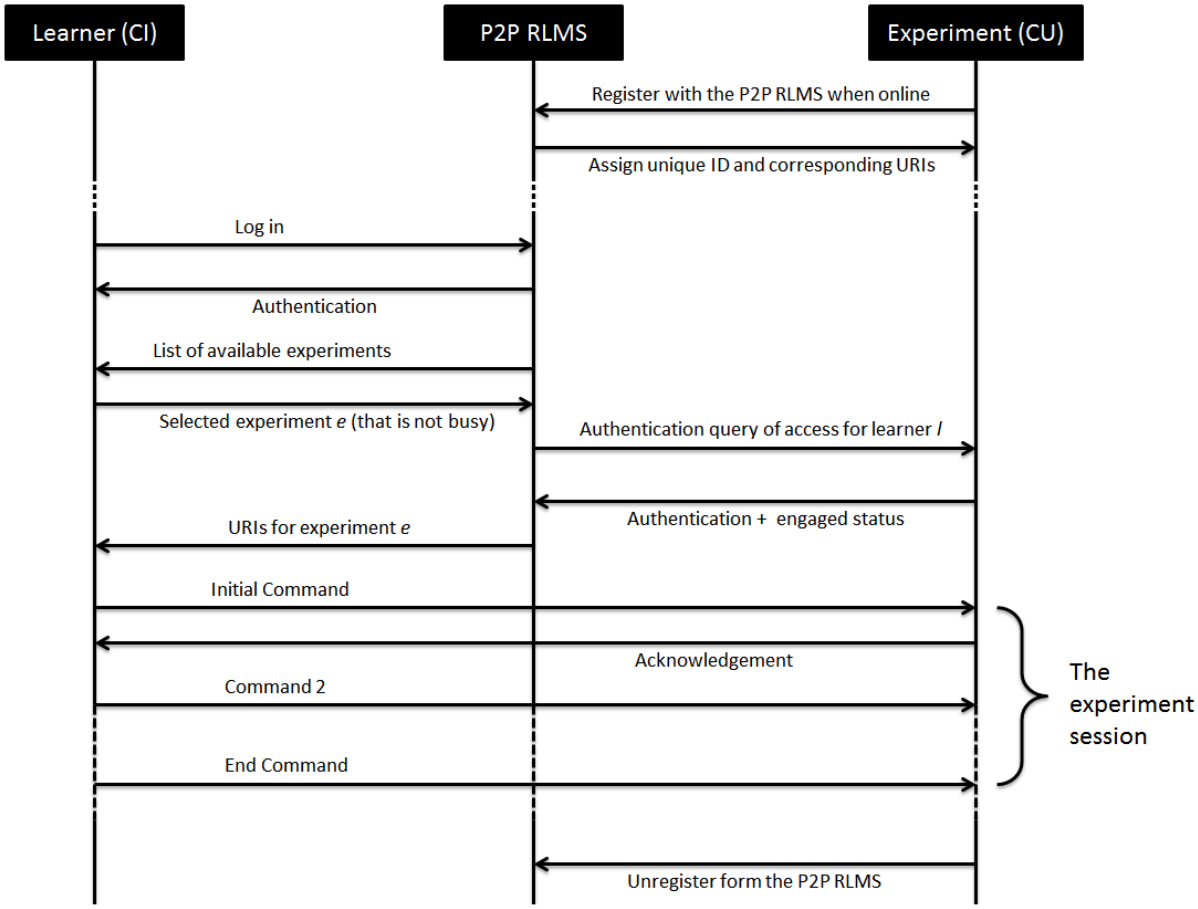


Figure. 11.5 The message flow chart

which is recorded in the list.

3. A CU is released by learner: The CU sends a message about their status: *false* which is recovered in the list.

4. A CU leaves the system: The P2P RLMS removes the CU from its active list of experiments.

In the beginning of the session, the learner opens the webpage (ralfie.net) and logs in to the system. The P2P RLMS verifies the log in and broadcasts a message to all known experiment CUs.. The CUs that are engaged are marked as such and the users cannot request that experiment for a window of time. The experiment sessions are limited to 15 minutes in the RALfie system, but this may be changed by the makers.

The program to query the P2P RLMS runs from within the learner's Web browser. The P2P RLMS checks whether the CUs are online and there status every 2 minutes

and update the learners list on their web browser accordingly. When the P2P RLMS finds a new CU or a CU with status as true, the learners list is updated. The learners can choose an experiment that they want to do. The P2P RLMS authenticate the learner against the CU and then allows the learner to download the CI which is global and common to all experiments, the CPL and other files corresponding to the experiments.

The P2P RLMS returns the links for the experiment to the CI. There are at least two links -one for the CU WebSocket connects for commands and the other for the Camera feedback. There can be additional camera feedbacks links as well. The CI runs from the web browser and uses the links to pass the commands to the CU. When the WebSocket connection is opened the CU changes its status to *false*. Once the experiment session is over, either due to end of time limit or the learner closed the session, the CI sends the *end* message. If the *end* message is received or the WebSocket session ends, CU changes its status to *true*.

Programming Paradigms of RALfie Experiments

There are three main advanced features in SNAP that are used extensively for RALfie. First is the *Network Capabilities.* When the SNAP IDE is opened, it establishes a WebSocket connection to the target controller on the VPN. Henceforth, each new command for the rig is sent through a WebSocket.

The second is using *Sprites as Objects/Components.* SNAP uses specific images called ‘sprites’ that represent each component of the user interface. These represent aspects of ‘object oriented programming’. Each sprite in the interface may be regarded as objects with its associated code. But the program is written in a functional manner and no object is ever explicitly used.

Every object in the UI is a sprite that can initiate its own code execution or perform a particular function. This implicitly implements the concurrency between execution driven by user generated events such as clicks and key-press, but the concurrency need not be part of the program logic. The most common sprites in the UI are:

- *The Narrator*: This object tells the objective of the experiments (see Figure 11.6a). It does not take any input either for the UI or the experiment, but simply presents a set of instructions and waits for the users’ actions.

- *The input components*: These include anything like a button that may be clicked to generate an event (see Figure 11.6b). Any image file can be used as the input components. Upon an event, these take an input either as numeric or text value or the click itself.
- *The output components*: These are those components of the UI which simply change state depending on the output received from the experiment. The output components on the SNAP interface may be optional as there is always a video feedback and certain experiments may solely rely on the video for showing the output.

All other functional blocks available in SNAP are used related with the sprites.

The third feature is using *Ports as variables.* Each controller is equipped with ports/pins and each pin is connected to a sensor or actuator of the experimental rig. Additional READ and WRITE components have been written for RALfie for interacting with hardware at different ports of the MCU. These were created under the control and sensor block in the SNAP. The READ commands take an input of a port number to return the value of sensor at that port. The WRITE command takes a port and value parameter to be written at that port to operate an actuator. These commands are put into other command structures to create the program logic of the rig operation as depicted in Figure 11.7. It shows a program where an actuator that is connected to a port is issued a *write* command.

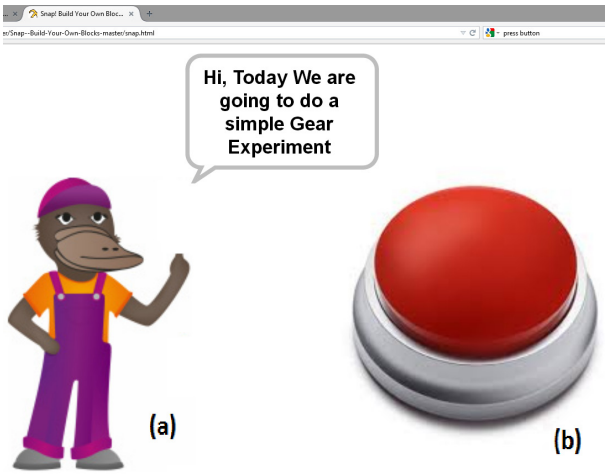


Figure. 11.6 (a) The Narrator of the activity (b) An example of an input component

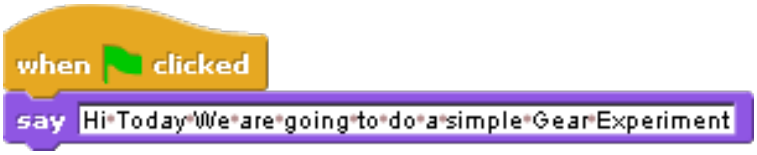


Figure 11.7 (a) Code example

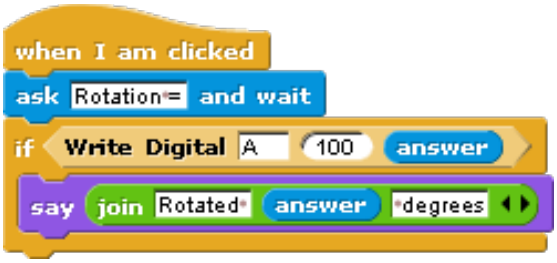


Figure.11. 7(b) Code Example

11.5.2 Lesson and Quest Management Interface

The IDE and its usage must be according to the characteristics described above. However, just satisfying the IDE requirement does not guarantee success of the system. The experiments must be stored in a cloud repository. RALfie uses a native Content Management System (CMS). Each quest is associated with a general description of the problem and related materials. The CMS also mentions the XPs and badges one can obtain for a particular quest. Also the CMS can store the pre-requisites of the experiments. The user's final set of answers to the activity are submitted to meet quest completion criteria. Other users' feedback on the quest, its due time and availability are also maintained by the CMS.

The SNAP programs can be converted to XML format including the images or sprites. Once the maker is ready with a fully functional experiment and UI, they can publish the experiment by saving it on the cloud. The corresponding XML file is stored in the cloud servers and associated with the activity in the RALfie. For the learners, the experiment xml file is downloaded and executed on the SNAP environment to run the experiment. They can only access the UI, but they do not have access to the associated code.

11.6 Example Experiments

This section presents selected examples of experiments that have been implemented in the P2P RAL environment RALfie. Both makers and users use the same portal to access the experiments. The makers can with between a maker interface where they

can create the program. The users do not have such capabilities and not authorised to see the maker's program in the interface. They can however download the program separately for sharing and further development.

11.6.1 Pendulum Experiment

The Pendulum experiment consists of a metal ball hanging by an extendible thread and a pushing mechanism to push the ball to generate a swing. LEGO Mindstorms EV3 components are used for this. There are three actuators for a) moving the ball up and down b) pushing the ball c) moving the pushing mechanism up and down.

The same experiment setup can address two alternative learning objectives: to determine the value of *acceleration due to gravity* (*g*) *constant* or the value of *length* of the swing thread. These objectives can be satisfied by moving the ball up and down and then measure the time taken for minimum 20 oscillations. Two alternative UIs with minor alteration have been built to support the two activities. This feature can encourage sharing of the program code among users and increase the number of activities as well.

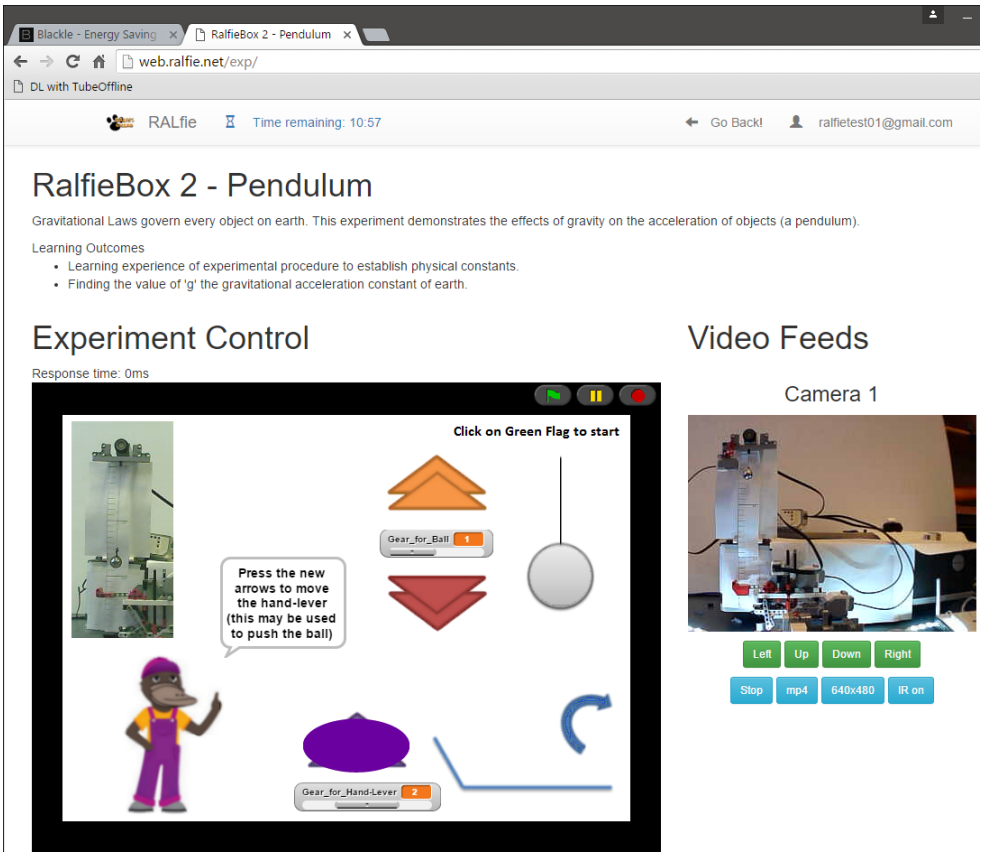


Figure. 11.8(a) The Pendulum example experiment UI in RALfie while initializing form a users' view.

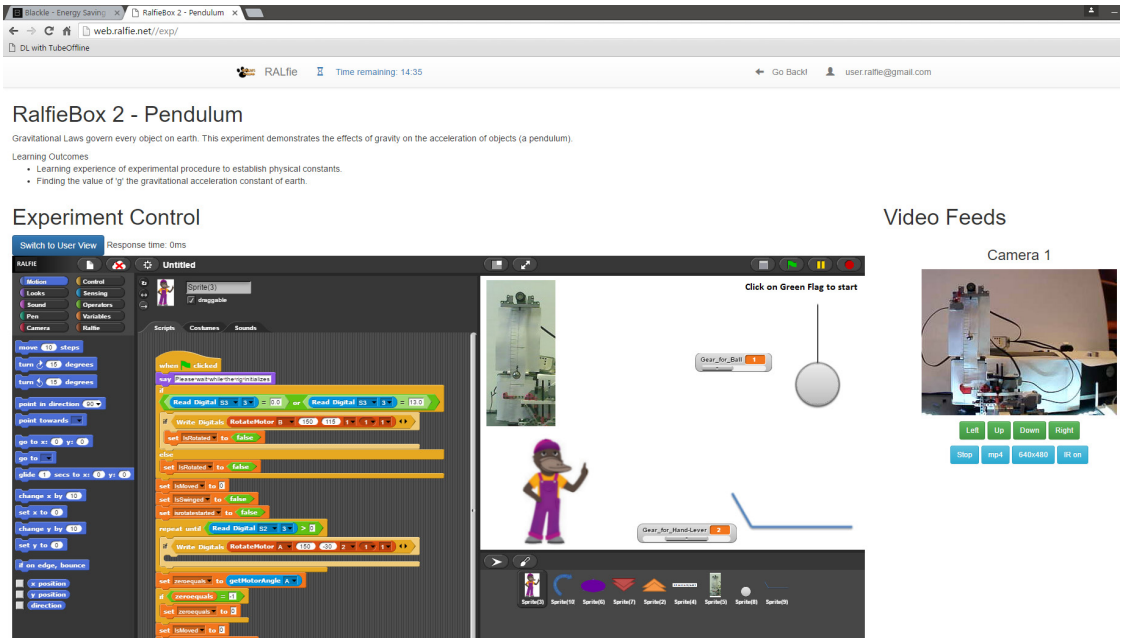


Figure. 11.8(b) The Pendulum example experiment UI in RALfie while initializing form a makers' view.

Figure 6.4 in Chapter 6 as well as the Figure 11.8 (inset) shows the pendulum experiment and it's UI in RALfie. The actuators are all controlled by the CU, which is a EV3 brick. The EV3 runs the corresponding IEM written in JAVA. Figure 11.8(a) shows the UI as developed by the makers and published in the RALfie website (https://ralfie.net) where other users can access it. The Figure 11.8 (b) shows the maker's view of the same experiment where they can edit the code. The UI consists of three control elements, one each for every actuator. Additionally, there is the narrator character, which describes the aim of the experiment and methods to operate the CI.

11.6.2 Gear Box Experiment

The Gearbox experiment consists of multiple gears that are interconnected. Each gear it connected to a particular coloured marker. Similar to the pendulum experiment, the LEGO components are used for the setup. Only one actuators is used for rotating the gears. The angle is the only input to the experiment.

The objective is to determine the ratio of the gears and understand that the relative speed of interconnect gears of different sizes. This is done by rotating the input gear, marked as red, by a certain angle as given by the user. The number of times the other gears rotate is observed to determine the ratio.

Note that this experiment setup may be used to run multiple experiment activities with

different narration, although the objective is always the same. For example, the same gear box experiments may be setup is multiple way with different sized gears, thus making a large variety of experiment setups. The same program however can be used to run all the rigs as long as the same port is used. This enables wide scale sharing of the CPL/UI among users.

Figure 11.9 shows the gearbox experiment and it's UI in RALfie. Similar to the pendulum experiment, the actuator is controlled by the CU which is a EV3 brick. The UI is developed by the makers and published in the RLMS website where other users can access it. The UI consists of one button for taking the user input. There is also the narrator character, which describes the aim of the experiment and methods to operate the experiment. Figure 11.9(c) shows a different gearbox setup that could be run with the same CI which may be shared from the previous example in 11.8(a & b).

11.6.3 Traffic Light

The traffic light experiment consists of LEDs that are placed on a printed paper map

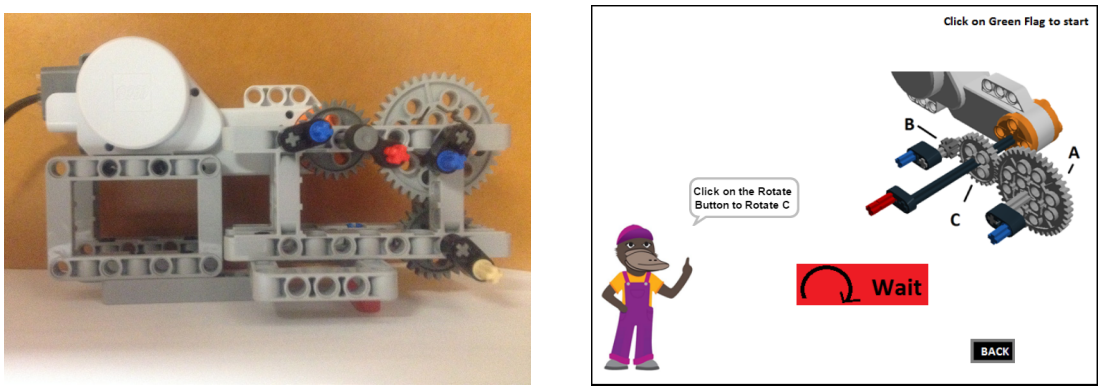


Figure. 11.9 (a) The GearBox example setup with LEGO Mindstorms and (b) it's UI in RALfie.

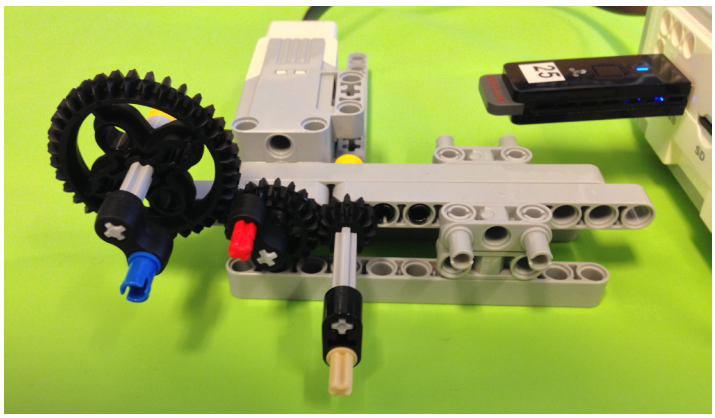


Figure. 11.9 (c) A different GearBox setup that can run with e same UI and CPL as the last one.

of road. Simple wires and cardboard are used create this rig.

The objective is to create a program that runs different aspects of the traffic light system. For example, create a rig consisting of 4 LEDs (Red, Green, Yellow and Blue for pedestrians). This activity consists of an infinite loop that turns on the red, green and yellow at definite intervals and checks the users inputs. The user can press a button for a pedestrian crossing on the UI that will turn on the blue LED when the red light is on next time. This activity runs with *Direct Access Control* by sending all instructions for operating the LEDs to the MCU from UIM on client computer for Red, Green and Yellow on the MCU (a BeagleBone in this case) and the UI sending only the 'button pressed' instruction.

Note that this experiment setup may be used to run multiple experiment activities. The

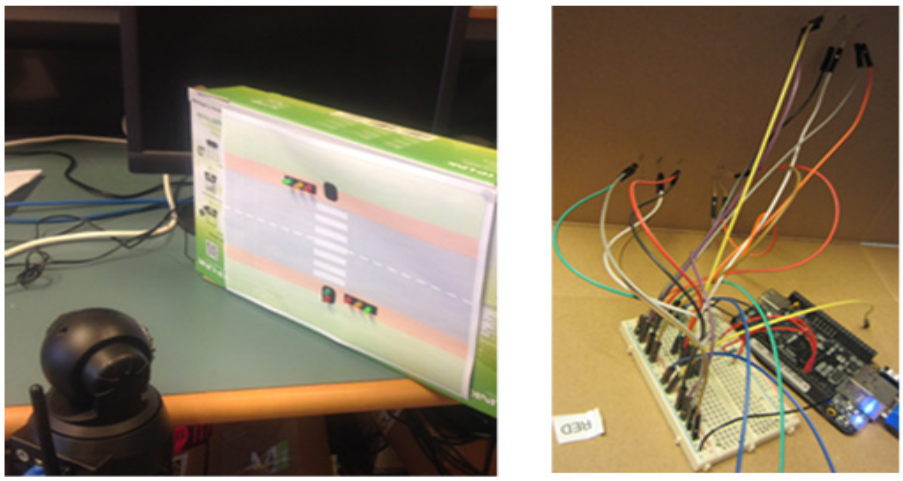


Figure. 11.10. The traffic light experiment example setup using a BeagleBone.

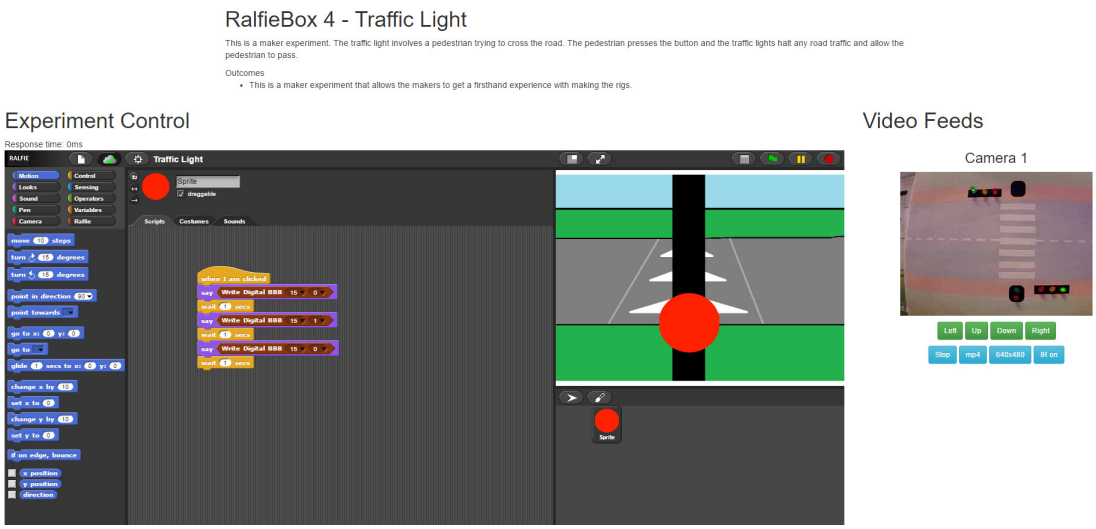


Figure. 11.11. The traffic light example UI in RALfie (maker's view).

main focus however is for initiating makers into creating a rig.

Figure 11.10 shows the traffic light experiment setup and it's UI in *RALfie* is shown in Figure 11.11. The actuators are all controlled by the CU which is a BeagleBone Black. The BeagleBone runs the corresponding IEM written in NodeJS. The CI is developed by the makers and published in the *RALfie* website (https://ralfie.net) where other users may access the default program for viewing only and then further develop the code. The UI consists of at least one button components for control, but the UI can vary depending upon the way the experiment is designed. Once published, the experiment can be used by other users.

11.7 User Trials and Feedback

This section presents the user feedback and results of using the RALfie system to create and/or use experiments. The primary aims of the trials were to determine whether the RALfie interface is usable and the concept of sharing experiments is feasible.

11.7.1 Trial 1 - Evaluation with Students

The proposed P2P RAL system is aimed to be used by students and teachers. It is necessary that they understand the concepts of constructing and programming a rig. To gauge the conceptual understanding of potential users, a group of students took part in a trial to create a traffic light experiment and access it online on the network.

The evaluation of the participants focused on these aspects. A total of ten participants took part in the activity. In order to ascertain the impact of the delays and port variable linkage, survey questions were asked of the participants. The results were:

1. All participants were able to assemble the experiments and plug them into the Internet within 10 minutes and then complete the activity as per the instructions provided.

2. They were able to change the values of the ports to make the necessary changes to the state of the peripheral devices connected.

3. They were able to change 'delay' or 'wait' values to modify the behaviour of the rigs i.e. slow down or speed up the rigs operation.

Whilst it required an extra compilation step to execute the code to communicate with the target MCU (a BeagleBone Black), this extra procedural step was easily understood and able to be performed without any interjection by staff. Collaborative team work with the activity was clearly evident as participants used differing pre-existing knowledge bases to satisfy overall knowledge and skill requirements to complete the activity. For linking the two parts of the activity together (UI and CPL), having a team was also particularly useful. Participants also spent a considerable amount of time reverse engineering the activity to ensure that connections were attached correctly.

11.7.2 Trial 2 – Evaluation with pre-service Teachers

A second trial was held with students from the University Education Faculty. They were pre-service teachers in an undergraduate education program doing a subject called EDP4130 Technology Curriculum and Pedagogy. This trial had 10 participants who had experience in classrooms. The aims of the trial was to

1. Establish whether hand on experience was essential.
2. Find out if the programming interface is suitable.
3. Determine the capabilities of making a physical experiment rig.
4. Understand the overarching architecture of publish and sharing experiments.
5. Finally, whether teachers would be interested in using these tools.

The following sequence of activities were conducted

- Users' preliminary proficiency with procedural programming in SNAP.
- Users' ability to create simple activity and the usability and effects of Procedural Programming for the purpose.
- Integrating a constructed hardware robot including a MCU and three Actuators into a small *activity*.
- Collaborating with each other to setup a activity
- Remotely using it to run the activity.

Participants were guided through the basics of the SNAP language and completed two sample example programs designed to familiarize participants with the development

environment, as well as the custom component to talk to the MCU in this case the LEGO Mindstorms EV3.

Each group was given an LEGO Mindstorms EV3 set along with the corresponding IEM installed on it, the RALfieBox, Cameras and Ethernet cables. The participants set up the RALfieBox which automatically connects to the Internet, and the RALfie RLMS. They then connected the EV3s to the respective RALfieBoxes.

Participants then constructed 3 wheel based robots. An activity was developed for this trial in which one of the groups robot was a goal keeper and the other two were competing robots trying to score a goal. This setup is shown in Figure 11.12.

The participants were asked to create the corresponding SNAP programs in RALfie website and save them. Figure 11.13 shows an example of a program. Once the robots were tested to run locally, the participants were taken to another room to run the activity remotely by viewing through the camera only on the RALfie website.

Observations

In the event that followed, all participants were able to gradually create the necessary program, having first established the networking to their robot, then creating the

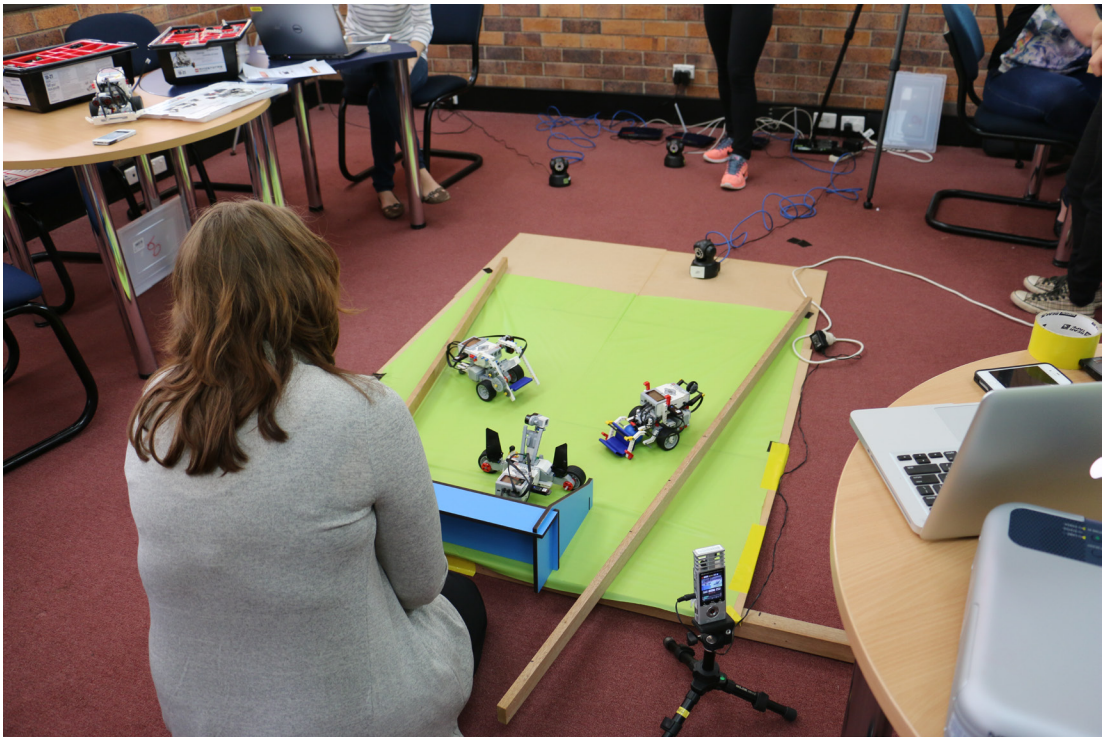


Figure. 11.12. The trail 2 of the RALfie system with three EV3 robots

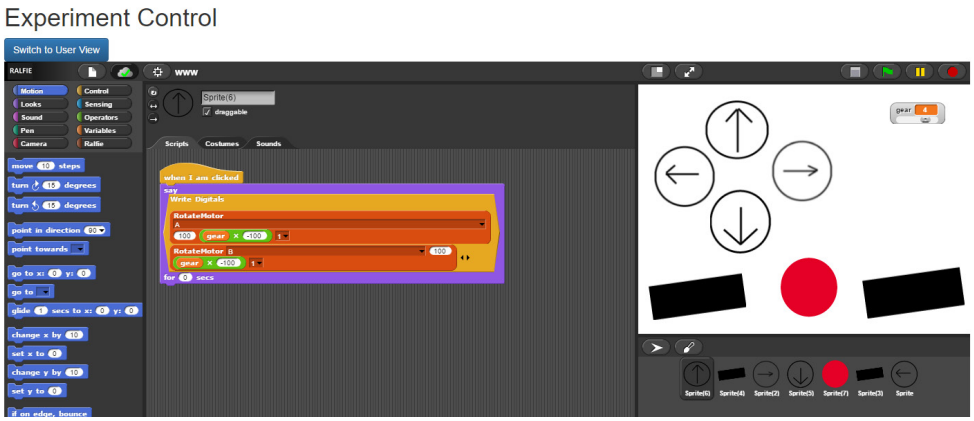


Figure. 11.13. An example program created by makers

sprites to which code would be related. Participants then built upon this with use of the SNAP output component to move each actuator in turn. This program was then built up until the robots were able to move in a controllable and predictable manner using skid steering.

All participants were successful in being able to move their robots through the field, whilst problem solving the skid steering, as well as the speed and loop parameters of their program.

During a focus group discussion afterwards, several key issues were identified:

Whilst participants themselves were aware of the objectives of the exercise, this was not reflected in their program sprites or control interface for the activity. Participants understood the link between the software “ports” and the hardware “ports”, however this was considered a threshold concept, where both ports needed to be synchronized, thus clear documentation and output component design is desirable.

It was also identified by participants that this could also cause confusion where LEGO Mindstorms (or other MCU) hardware faults were present, particularly poor wiring connections, or mechanical design flaws) would cause incorrect response to the SNAP program. As such debugging systems (although not present in the trial) are desirable within the SNAP interface.

With regards to instrumentation and sensors, participants were unsure what these devices or mechanisms were, and thus some examples or tutorials on sensors and instrumentation was requested, and although not specific to SNAP highlights the issue of open-ended hardware design with novice programmers.

Participants indicated that the organization of the SNAP interface was at first confusing, but related to familiarity with the interface. When creating the interface, participants felt a more interactive interface was required, where SNAP blocks showed or indicated what the physical object would do with any given SNAP block.

Participants felt the most appealing aspect was to have a quest, and achieve a level of operation or understanding about that quest. In this case moving the physical robot around.

With respect to the aims of the trial the results were:

1. The kind of hands-on-experience done in the trials is essential and suitable for school children.
2. All participants had successfully created the program.
3. All groups were able to create their own robot with various designs.
4. All the participants understood that they could use RALfie to demonstrate someone else's rig first, to understand the capabilities of the system before building their own. Participants indicated that a bank of example activities would considerably help their understanding of the concepts. Additionally, it was indicated that sharing of the activities with other participants was the most memorable aspect of the trial.
5. All participants indicated that this type of activity could be done at schools but may not be suitable for homes.

11.7.2 Trial 3 – Second Evaluation with pre-service Teachers

In order to test the feasibility of makers being able to make the experiments, further interviews were done. 10 Participants from another course were shown the 3 experiments described in the last section - pendulum, traffic light and gearbox. All the participants performed the 3 experiments remotely and were shown the components of the actual rig.

Table 11.1

Question	Yes	No
To run an experiment you need to have video cameras and robots. If we provide all the gear to you, do you think you	75%	25%

can host an experiment from home?		
If you build this experiment at home and share with the RALfie system, can you keep the experiment online for 24x7?	0%	100%
Will it (the time frame) depend on the nature of the experiment as well?	100%	0%

The participants were asked the following questions and their responses are shown in Table 11.1.

Most participants indicated that they can create the experiments and host them from home provided they have adequate online support materials. All of them indicated that a good communicate support is required where the process of making simple components related to the experiment they are trying to create is essential.

All of them agreed that keeping an experiment at home online for 24x7 is impossible. However, 50% indicated that in a school setting, the experiments can be kept available for a few months until a target group of students have all run it.

All participants indicate that the time for which an experiment can be kept online will depend on the nature of the experiment and how it can be constructed. If an experiment can be constructed with simple but sturdy components taking less space, they can be kept online for longer period of time.

The above result shows that individual makers can create and host experimental rigs ta their home.

11.8 Summary

This chapter has discussed the implementation of P2P RAL technologies for STEM education particularly for the RALfie project. It can enable individual makers to create and share their experiments. The experiment setups can be used for multiple experiment activities. Each of these activity could use the same rig but with different aims and narrations and learning outcomes. The activities were set up as web based SNAP programs. The concepts of P2P RAL such as *ports* and *delays* (wait command) are implemented in the RALfie system. The programming environment is graphical

with drag and drop components. A narrator-based approach is used where an animated character is used in the UI to describe the aim of the experiment and describe the methods to use the UI. This follows the principles of creating small games for learning purposes.

Finally three different examples of experiments/activities that are created and available on the RALfie system are discussed. Results of two user trails were reported where the usability of the online RALfie system along with the process of creating rigs and programs were positively established.

The RALfie implementation of the P2P architecture creates huge opportunities for makers including students and teachers in STEM education. The method of enabling makers to create their own rigs to share with others cannot only grow their own interest the STEM subjects but attract others as well. The P2P RAL does not implement a laboratory in the strictest sense of an experiment being part of a predefine curriculum that must be finished within a time period. Instead it is focused on enable makers to communicate and share as much as they can and as long as they can. Such principles are prevalent in the use of social media and could be expected to be used efficiently in STEM education as well. Further research is required to successfully deploy this system with appropriate pedagogies.

12

Other Issues –Augmented Reality

This chapter discusses preliminary work in an additional area - augmented reality tools for P2P RAL.

This chapter discusses an additional feature in the context of P2P RAL - augmented reality tools. Augmented Reality (AR) is technology to embed media information in video streams to create a rich interactive user interface. Embedded AR components can be text or highly complex graphics. AR technology reacts to the surrounding environment. It responds with AR components depending upon the visual inputs to the system. AR components must be updated in real time by recognising the input video frames' contents and processing it according to a pre-determined logic. Other common inputs apart from static objects are gestures from the users themselves which is also part of the environment. AR systems can also take conventional inputs such as mouse and keyboards.

AR is used in many areas of science and technology including computer games for recreational purposes, sports and entertainment, navigation and tourism. AR has also been used in education [232]. In this section, the aspects of integrating AR into RALs are discussed. A P2P RAL creates the challenge that the experiments must all be provided with a set of tools that are useful for all types of experiments. Thus a common set of requirements and conditions are considered. Four different levels of relationships between the real and virtual components in an AR application have been identified depending upon their activeness. Two different solutions of has been proposed as a part of the P2P RAL system to deal with these cases, super-imposing animated and interactive objects on video streams and identifying and tagging objects to corresponding sensor values. These solutions in form of AR tools can be used for multiple experiments independently designed by different makers.

The rest of the chapter is organized as follows: Section 12.1 discusses the current status of AR and its applications in education and in particular RALs. Section 12.2 discusses the P2P RAL system and identifies conditions and alternative approaches to AR in RAL context. The SNAP programming platform [118] and the AR tools are presented in Section 12.3. Section 12.4 discusses the implementation methods of the tools described. The contents of this chapter are based on related publication in [233].

12.1 Related Work – Augmented Reality

Augmented Reality can be perceived in multiple ways. Most commonly it is defined as a mixed environment that blends digital information with real world objects in a meaningful way [234]. The amount of real world entities in the environment should be more than the overlay information for it to be AR [235].

There are different classes of AR environment based on how immersive they are. One common form of devices includes a head mounted displays system and possibly hand gloves with feedback [236]. These are fully immersive environments that enable users to experience whole of the reality environment with augmented features. It also allows more accurate interaction with the AR environment. Fully immersive AR is achieved by using wearable devices such as smart glasses or head mounted displays. These devices have cameras mounted on them which are capable of running applications to process the video which is the visual area of the user. The view of the user is then enhanced with overlaid information.

The other type is desktop AR [237] which only covers partial portion of the surrounding of the user, in particular what can be shown in the desktop screen. The view is limited and interaction with the environment happens with regular input devices for example, mouse and keyboard. While full immersive AR is more attractive and advantageous, they have many problems. The hardware required is expensive for being a commonly available tool to be used for educational purposes. They require high precision to recreate the augmented feature. It requires expertise to set up and maintain the system. They are prone to errors [238].

Moreover, fully immersive AR works by augmenting the local environment with virtual objects. This by itself takes considerable cost, processing power and technology. It is more difficult and unnecessary to recreate a remote real environment

completely and then augment it with virtual objects. The ability to view remote real environment is much reduced and only a fixed set of views are available through cameras. Thus these video streams can be projected directly onto screens.

With immersive AR, it is difficult to obtain a generalized environment to create a number of interfaces from a single platform. This causes disparity between the interfaces of different systems with little in common. Also, immersive AR is not always necessary for good educational outcome [234].

AR in education mainly aims at providing rich educational experiences. Such systems usually concentrate on the desktop AR or mobile devices. Traditionally, some systems use markers for identifying the location in the real world stream to be replaced with the augmented information of objects as well as a unique identifier for what to display. The augmented objects are stored in a database, against a unique identifier and reproduced when the desired marker with that identifier appears on the screen. This also requires accurate computer vision techniques to correctly identify the marker and the encoded identifier within it.

This type of technology helps in understanding operation and models of the objects that are available in-place with real world learning materials. They present the users with a quick in-depth augmented multimedia experience during their interaction with real world environment.

Augmented Reality features have been added to RAL experiments before [232, 235, 239]. Usually, the AR is desktop type and mostly the augmentation is overlaid virtual components such as switches that can be manipulated by users. In [235] the virtual elements on FPGA boards, users can remotely interact with the real and virtual devices. The real devices are viewed through a camera video feedback. This approach leads to a very realistic environment, as the majority of what the users see as part of the user interface is real objects: here the FPGA board. Only small portions of the video feed back are overlaid with other information and graphics that takes users inputs.

The main limitations in the broader context of P2P RAL are that these examples are designed specifically for one experiment. Moreover, the co-ordinates of the virtual objects are directly tied to the co-ordinates of the hardware in the video feedback.

Any change in the hardware orientation may require change in the AR setup as well. As part of a P2P RAL, a common web-based instrumentation platform is used by multiple users with different hardware setups. Thus the P2P RAL systems must identify which objects in the real environment need to be supplemented. This enables makers to specify certain objects in the video feedback and associate the virtual components with them within the online environment.

12.2 Augmented Reality in RALs

This section describes the application areas of AR in RALs, and types of AR and constraints of applying them.

In general augmented reality can be used in many ways [234] but the most common approach is to draw virtual objects onto the real world video feed. Augmented reality in RALs can serve two key purposes: to show hidden or invisible views and to display additional information.

In certain experiments some objects/entities may not be visible to the camera. For example magnetic fields that attract magnetic materials generated and studied by using different electromagnets [240]. These entities which are part of the experiment, may be implanted into correct positions by using animations. This involves re-drawing certain objects such as arrows over the region to indicate the presence and orientation of the entities.

Another set of objects that needs to be presented is text information relating to certain real objects in the video. It is best to draw the text onto the video feedback close to the associated object. To do this however, the objects must be identified and tracked in real-time during the experiment. Overlaid text information must be updated in real time as well to reflect the change in the state of the object.

12.3 Levels of Augmented Reality

In the current context, AR is the process of overlaying virtual objects including scalar images or vector animations or both onto the video feedback. The video feedback has a definite frame rate and resolution and thus a fixed number of pixels (P) for each frame. For the AR, a pixel p in the feedback may contain either real-object or virtual object (or maybe fractionally both). Thus two measurements can be defined -

Virtual Pixels (P_V), the average number of pixels that relate to a virtual object. Then, ΔP_V and $\Delta \rho_V$ are two parameters that signifies average change in the P_V and ρ_V over time in the video feedback where ρ_V is the matrix representing the position of the virtual pixels.

Real Pixels (P_r), the number of pixels that relate to real objects. ΔP_r and $\Delta \rho_r$ signifies average change in the P_r and ρ_r over time in the video feedback where ρ_r is the matrix representing the positon of the real pixels.

This allows for different degrees of virtual and real objects to be blended. In the P2P RAL, AR may be implemented by having

Case 1. More virtual components with more active behaviour than that of the real objects. i.e.

$$P_V > P_r \text{ and } (\Delta P_V)(\Delta \rho_V) > (\Delta P_r)(\Delta \rho_r)$$

Case 2. More virtual components than the real objects but less active in behaviour than the real objects. i.e.

$$P_V > P_r \text{ but } (\Delta P_V)(\Delta \rho_V) < (\Delta P_r)(\Delta \rho_r)$$

Case 3. Fewer virtual components than the real objects but more or equal active in behaviour than the real objects i.e.

$$P_V \leq P_r \text{ but } (\Delta P_V)(\Delta \rho_V) \geq (\Delta P_r)(\Delta \rho_r)$$

Case 4. Fewer virtual components than the real objects. i.e.

$$P_V < P_r \text{ and } (\Delta P_V)(\Delta \rho_V) < (\Delta P_r)(\Delta \rho_r)$$

The first scenario is in the space of augmented virtuality [241] (not in scope of this section) where both virtual visibility and associated information are high and the real objects do not change their orientation much. In the second scenario real objects change their orientation more often compared to (or equally to) the induced visibility/information. In the 3rd scenario the users have fewer virtual components and the real world objects, both can be equally active. In the 4th scenario, users interact largely with the real components and only supporting information are displayed as visible information.

In the P2P RAL system, all AR capabilities are embedded in the online platform. Makers can use AR tools to incorporate virtual objects onto the UI along with a video feedback from the corresponding Camera.

12.4 Integrating AR in the P2P System

The role of AR comes into the programming part of the rig. The AR information includes two type of procedure Virtual Object Creation (VOC) and Real Object Identification (ROI).

For the VOC, the makers can create animations in the SNAP environment and these needs to be aligned correctly to the video feedback. This can be achieved by ensuring that the virtual objects and all of their re-orientations are within the bounds of the real objects coo-ordinates in the video feedback.

For the ROI, as mentioned earlier, makers are not expected to create markers [239] for AR objects. It is also not possible as the SNAP system does not know what the maker wants in the AR. Thus any real object that needs to be augmented with virtual objects must be identified by the SNAP system in the video feedback. Once the desired objects are identified, they may be associated with corresponding virtual objects or overlaid text information.

The maker has to perform these tasks as a part of building the rig which is supported

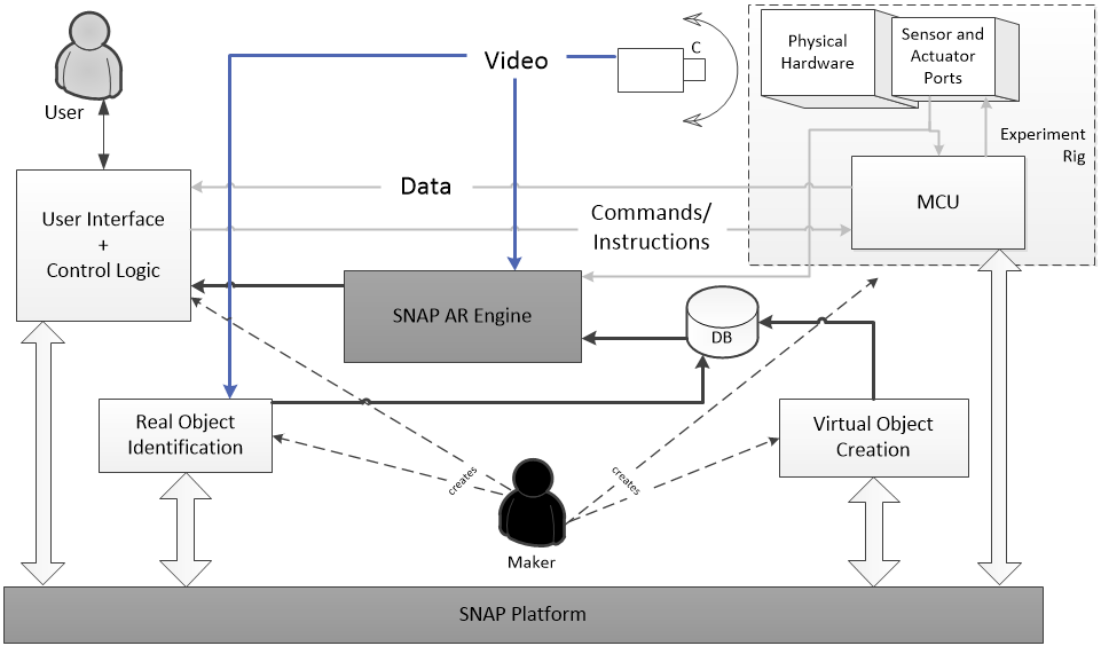


Figure. 12.1. The SNAP environment and the experiment rig

by the SNAP platform. The UI and CPL is also created in the SNAP Platform. A separate AR engine (also part of the SNAP platform) can combine the stored Virtual objects and corresponding Virtual objects/components into the video feedback during run-time for any user.

Figure 12.1 shows the SNAP system with the various AR related components. There are two separate streams that feed into the UI - the resultant data and the SNAP AR frames. The resultant data is the data obtained from the experimental rig i.e. sensor data of actuator success or failure data. The SNAP AR Engines generates the frames for the UI that shows the video frame received from the experimental setup modified with the AR components. The video feedback is received through an IP Camera as a MJPEG stream in the SNAP which is further processed according to the experiment and the corresponding objects saved in the database. The maker is responsible for both ROI and VOC both of which are optional for a given experiment. The objects identified and their associated media by ROI or created by VOC and their activities are stored in a database alongside the experiment. This database is used by the SNAP AR engine. The SNAP AR tools run on top of the SNAP execution engine that processes the users' program and communicates with the experiment.

For P2P RAL, the Cases 2, 3 and 4 as mentioned in Sections 12.3 can be addressed as follows.

The Cases 2 and 3 can be handled by super-imposing the desired virtual objects on to the camera feedback. This allows for the camera feedback to directly display the real objects without any alteration. This will work only if the amount of virtual objects pixels is less than or equal to the number of pixels for the real objects. This does not work well if ultimately the number of virtual object pixels is greater than that of real objects as a large number of pixels are required to be re-drawn in each frame of the video, reducing system performance. It also increases complexity of real-virtual object pairing and the ways to store them in the database and display them farther affecting performance and experience.

The Case 4 is displaying information associated with certain components of the rig. The virtual components are not special objects but only text that is updated in real time. In the rigs, each actuator will result in change of orientation. This change may be associated with a certain component in the rig and thus a corresponding sensor may

be able to read the changed values. These values can be shown in real time over or near the component in the video feedback.

These help makers in understanding key components of the design and identify any weakness. It also helps makers to quickly associate sensor value to any object with the need of displaying them explicitly. This saves screen space which is very important for mobile devices. For the user, the augmented components helps in identifying and understanding the changes in the experiment easily without having to look into detailed UI reports [242].

12.5 A Sample Implementation in RALfie

This section discusses the methods to implement the two solutions to the Cases 2, 3 and 4 as generic tools for the online SNAP platform in conjunction with a P2P RAL.

Super-Imposing the Camera View

The SNAP platform has a designated area of screen that is called a stage. The stage is where all the objects of animation and other output data are displayed. The simplest form of AR is to super-impose the cameras view below the stage. This is done by connecting the stage background to the camera stream. The camera steam may be resized and placed at any position on the stage or a full screen mode can be applied.

Makers must include a command to start the AR. If the AR is not started then the SNAP environment behaves like a typical non-AR setup. Once the AR mode is started, the camera is visible. Then the makers can include any object they wish on the stage that will appear on top of the video stream. The makers can make precise movements according to the underlying changes in the camera feedback.

Figure 12.2 shows an example of this type of AR. The experiment activity concerned is a traffic light system as shown in Figure 11.9. The rig has LEDs that go *on* and *off*. Virtual objects are cars which stop and move according to the LEDs status. The LEDs are connected to ports on the MCU and controlled with commands from the SNAP UI to the MCU. The video feedback shows the LEDs and the background roads as well as the car movements. These are controlled through SNAP depending upon the data received from the MCU.

This type of AR suffers from two problems: stability of the camera view and network

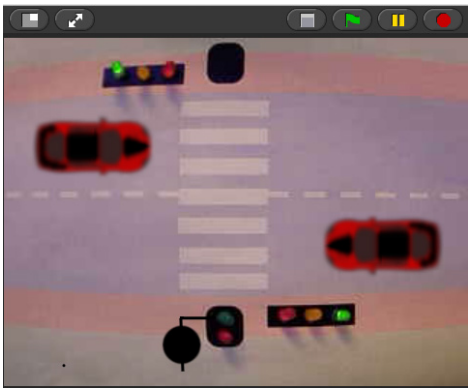


Figure. 12.2. A traffic light example in SNAP with real LEDs and virtual cars

response time. The camera view is assumed to be static. The problem may arise if the view changes due to the camera getting moved accidentally. This could be addressed by detecting changes in view and notifying makers to put the camera back in place. However, if AR objects are precisely programmed, they may require re-calibration of coordinates to ensure correct UI interactions. There is no object identification procedure to re-align the virtual objects accordingly.

The response time is the time taken to retrieve any data or video from the maker node to the user node. On the Internet this may be high. The animation frame rate will be typically faster than the frames from the video, thus it will create a lag in user interaction if every frame of the stage is attached to a new video frame. For this purpose, the video is handled by a separate process that runs parallel to the SNAP execution platform running the virtual objects activity. Whenever the video frame is retrieved, the stage background is updated accordingly. Thus the users' interaction with the UI components remains unaffected.

A second problem is the difference between the arrival rate of data and the SNAP animation frame rate. Due to response time, the data may arrive at a later time than the relevant frame where the data was supposed to have any effect. Thus, all SNAP execution including animation is suspended when a message or instruction a is issued from the SNAP to the MCU at time T_a. During this period the virtual objects do not move or operate thus creating a paused state until the data is received. With higher latencies, the number of paused states will increase in an interval of time, thus affecting quality of experience. But the data and SNAP animation will remain synchronized thus not affecting the learning objectives.

The last problem is the arrival time difference between the data and video stream. Typically, the command and sensor data exchanged between the nodes is very small and delivered at faster rate than the video as well. This causes the problem of de-synchronization between the video frame and the virtual objects. Thus, the SNAP execution engine is paused until a new video frame T_a^v is received after receiving a new data T_a^d after an instruction a is issued. With stable Internet conditions, there will be least effect on the performance and interaction of the users. Thus the paused time after an instruction a is issued to the experimental rig from the user interface is:

$$Paused\ Time(a) = \max\{\ T_a^v - T_a\ ,\ \ T_a^d\ -\ T_a\}$$

this means that the paused time is the greater of whichever arrives last, the video feedback or the data feedback.

Object Identification and Tagging

The second approach is the solution to Case 4 where the objective is to identify individual objects in the video stream and tag them. The objects cannot be mapped to any fixed global database in SNAP as there is no limit to what the users can use to create the rig. Thus the SNAP platform must be able to store these additional components alongside the control logic program for every experiment.

The following steps describe the process of identification and tagging:

- The SNAP AR engine identifies objects that change position (or shape) over time. This process is ROI.
- A record of the desired objects is created for the particular experiment based upon physical properties of the objects i.e. colour, contour, size or even an image of the object. The record is stored in the database for the experiment alongside the control logic.
- Once these objects are recorded, the makers can attach a sensor's value to the object, which is also stored in the corresponding record.

Actuators cause changes in the rig positions. The magnitude of the change can be measured by sensors. For example, Figure 12.3 (b) depicts the pendulum experiment. It the user has to drop the ball, the corresponding actuator has rotated by a certain

degree. The ball then changes position in the video stream. The length of the drop is a function of the rotation. This change in the balls position can be displayed in real time as augmented texts pointing towards the ball.

The maker is able to attach the sensor values *x* as a function *f(x)* which is constantly updated on the screen. The maker can also designate a particular area of the screen where the text is displayed. This should ideally be a space that does not have any meaningful object and the text should not overlap such objects. However, in certain cases where, the rig has massive change in position, no such suitable space on the screen may be available for the entire duration of the experiment. The SNAP AR engine must determine a suitable space to put the text. The user is also able to switch on and off the AR components to make them visible or invisible.

A prototype ROI mechanism has been developed in P2P RAL - SNAP as follows:

1. The initial image of a video stream when a session starts is stored as the background image (B).

2. Once any object moves in image Fi, it is isolated by subtracting Fi' = Fi - B. A residue of the object is left in B, which is identified using subsequent frames, as the residue will always remain static. The pixels of the object residue in B are replaced with the corresponding pixels in the current frame Fi.

3. A clustering mechanism is used to remove noise and get the actual objects.in the frame F'i. The clustering mechanism takes into account the potential radius (as specified by the maker) of the target object that needs to be tagged. Thus any object that is larger than the size is automatically put off the list. Figure

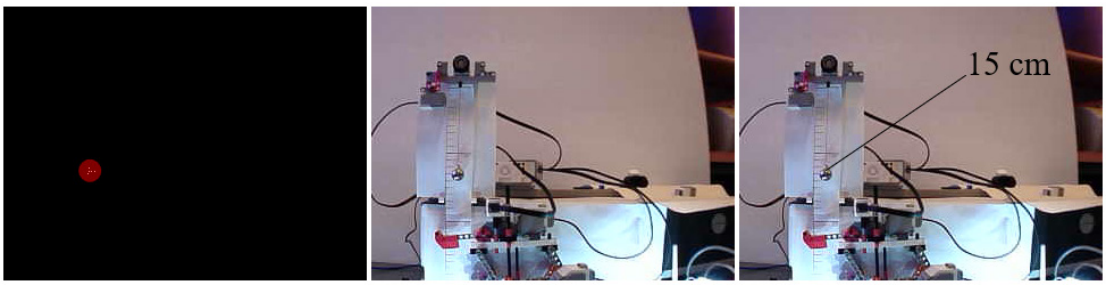


Figure. 12.3. The pendulum Experiment. (a) The difference in frames to identify the moving object (i.e. the ball) (b) The original video feedback of the pendulum experiment (c) The final video feedback with the sensor value as shown to users.

12.3(a) depicts an identified object.

4. The makers can then select the object(s) that need to be stored permanently and will be used for AR.

5. The maker then uses a SNAP block to associate a sensor value with the desired object and also mention its x, y coordinates on the stage.

6. Each object is stored in a database and marked with a unique identifier. The object does not need to be identified in real world as what it is, but only matched relatively in each experiment session.

It may be noted that the actual algorithms to realize each part can be implemented in multiple ways. For example, the DBSCAN algorithm [243] is used for creating clusters of right size.

When the users run the experiment, the AR module checks if any of the objects, stored in the database while creating the experiment is in the frame. If there is any such object, then the corresponding, sensor values are shown if AR tools are activated. Figure 12.3(c) shows the final output of the Object Identification and Tagging (OIT) process where the ball is tagged with the value of the sensor measuring its height. This image frame is placed in the stage of the SNAP environment.

The AR module in SNAP to create each frame of the video feedback to the user works in two steps as shown in Figure 12.4. First, the video feedback is analysed and for each frame, the objects are identified and tagged according to the makers' selection and function. Second, the virtual objects created by the makers are then placed in the video feedback.

The prototype system for ROI (or its implementation - OIT) is successful in principle

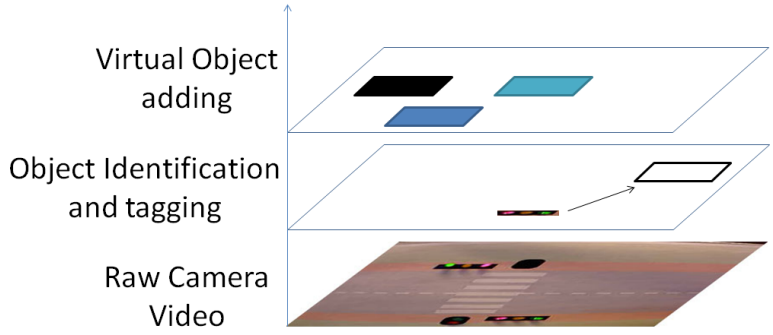


Figure. 12.4. The layers of AR components

to provide a generic tool to create AR interfaces for multiple experiments using the steps described earlier. But it is not able to support all types of experiments. There are major performance shortcomings that were noticed.

First, the JavaScript based SNAP environment runs in a web-browser. The OIT AR tools increase the CPU rate to more than 33% on a 2.5GHz, Intel i5 processor using the web browser Firefox version 41. This indicates that AR tools require considerable computational power which may not be available on mobile devices.

Second, in Step 3, a clustering algorithm the DBSCAN algorithm is used which has 2 inputs ε and p where ε signifies the radius around a point and p is the minimum number of points (or pixels) around a given point. The resultant clusters will have each point in the cluster surrounded by a minimum of p clusters within a radius of ε. This is an ideal way to determine objects and reduce noise in the video input. However, this also adds to parameters that need to be altered to an extent to identify the desired objects correctly. Further improved implementations of this have to either automatically adjust this or makers choose desirable values.

Third, in Step 4, the objects properties - average colour for red, blue and green along with a range of minimum and maximum heights and widths of the detected clusters are stored. While this is sufficient to identify small and mobile objects with uniform colour, it may fail in some scenarios with larger objects.

12.6 Limitations and Future Work

One of the limitations of the implementations described here includes the need for static camera positions. Ambient lighting changes can also affect the outcome. This will create larger differences between the background (B) and any subsequent frames. Thus the experiments with AR tools must be set up in a well-defined environment. Further work will look into minimizing the effect of response time on the performance of AR tools.

The OIT in this work assumes that there are fewer moving or changing components compared to static objects in an experiment view. At the moment issues such as occlusion are not addressed, i.e. when an object is covered by other objects. The OIT described here can be used only for moving or largely changing objects visible to the camera and if there are no two similar objects in the feedback.

12.7 Summary

A set of generic augmented reality tools has been discussed that can be integrated into a P2P remote laboratory architecture, as the P2P RLMS must provide generic tools for all makers to create a variety of experiments. AR tools are based on the activity level of the virtual and real components in the experiment video. Such tools can help users of the experiments to quickly identify the changing parameters of the experiment and help makers get acquainted with the relationship of the parameters and the rig operation. In short, augmented reality tools can help users and makers recognise the most important learning concepts in the experiment. It can highlight important data and help users to understand the experiment. The proposed methods are also applicable to other RAL experiments.

13

Other Issues – Scheduling

This chapter discusses preliminary work in an additional area - scheduling for P2P RAL.

This chapter discusses a new scheduling mechanism in the context of P2P RAL. Scheduling is important as the users in a remote laboratory can only use a device individually and the RLMS must organise access such that user sessions do not overlap with each other in harmful ways affecting learning outcomes. The scheduling in the RALfie system can be more complicated given a situation where multiple users could be trying to access the same experiment which may have been implemented in multiple sites and the makers having constraints on how long they can keep their rigs online.

The rest of the chapter is organized as follows: Characteristics of scheduling mechanisms used for traditional RALs are discussed in Section 13.1and 13.2 followed by an analysis of which of these is suitable in the context of P2P RAL and quest-based education. Requirements and assumptions regarding the new scheduling strategy are presented in Section 13.3. The terminal assignment problem and its application in this context are discussed in Section 13.4. Section 13.5 presents simulation results on the proposed scheduling method. The contents of this chapter are based on related publication in [211].

13.1 Scheduling

RAL environments enable users to control equipment and collect data from them without the need to be present in the laboratory or classroom. This means that users

work in a disjoint manner, unable to co-ordinate the usage of the experiment manually with other participants as it often occurs in a regular on-site laboratory. Generally, the remote laboratory management system (RLMS) handles scheduling and user access, as multiple users typically cannot control experimental instruments simultaneously.

The P2P RAL uses quest-based learning [33], a special instance of game-based learning where the players are given a set of targets or goals to achieve. These target objectives relate to particular learning objectives. The RALfie [244] project introduces quest-based learning to P2P RAL environment to allow school students to design and build their own experiments [245] and then run each other's experiments. Each experiment is a *quest* which in turn is part of a large hierarchy of the *quests* that are related to different fields of study. Completing a series of such *quests* means the player has gained the knowledge about a particular STEM topic. The use of *quest*-based learning adds new varieties of motivating factors and user requirements to achieve certain goals in the entire RAL based game and thus affect the scheduling.

It may be noted that without the unique requirement of the RALfie system's quest based learning, queueing or time reservation method can be easily used for the P2P RAL system.

Time scheduling, i.e. making sure that each user can access experiment effectively, has been addressed in the context of traditional RALs [55]. Scheduling users in a distributed environment that features a gaming approach poses new challenges: Firstly, experiments will usually not be online all the time, but be limited to specific time periods. This is in contrast to regular RALs where the equipment is typically available '24×7'. Secondly, due to the nature of quest-based learning, there are a number of prescribed sequences in which experiment have to be completed, i.e. users will have varying requirements while selecting experiments. In technical terms, both identify additional constraints for the user-scheduling problem.

13.2 Related Work – Scheduling

Time scheduling in remote laboratories has been investigated in the past [55,34, 35, 198]. Time scheduling in RALs is essentially solving a problem of resource allocation within a given set of equipment (resources) that can perform various operations. The solution will vary depending on the nature of rig operation for example, time taken,

amount of data etc. The goal is to maximize both the efficiency of the instruments' use as well as the quality of users learning.

Time Slotting or *time reservation* divides a given period of time into discrete slots according to the experiment nature. Each slot is then allotted to a user during which the user has full control over the equipment. One characteristic of the time slotting mode include that the user gets the full access and gives inputs at will. The user input gathering and processing does not have to be at periodic interval. The inputs may be given at random time within the time slot. This allows the users to apply a wait and see approach where they can take time and analyse their current position in the experiment and then move forward.

This feature is most suitable for experiments that are fast experiments involving a variable environment and the users making multiple decisions to get the required result. By allotting a time slot to individual users the utilization of the resources drops. A poorly designed time slot length can cause the users to finish the experiments too early. Also, the users may not use the equipment much within the time slots and spend more time on the decision-making component leading to internal slot fragmentation. Since the times at which the users may start the experiment are discrete, they are allotted the slots in a *First-Come-First-Serve* basis. This may result in multiple time slots being unassigned if none of the target users can use it at that time leading to external slot fragmentation. The number of slots limits the total number of users that can be served.

Queuing is the converse of time-booking. It creates a list of users' requests and processes one request at a time. The user gives a set of inputs for an experiment and each of these requests (with a set of inputs) are put in a *queue* by the experiment controller or the RLMS. The requests are then executed in order i.e. FIFO manner. The characteristics of the queuing mode include that users get full access but are unable to provide inputs at will. The users' inputs are bundled as one closed packet of information that is processed as whole at discrete intervals of time. The inputs may be given at any time but the response will depend upon the number of requests in the queue before that. This denies the users an interactive session and there is no scope for changing inputs once experiment processing starts.

This feature of reduced accessibility is most suitable for experiments that are slow and

can be done with automated instruments that perform experiments with user inputs. The environment variables can be set only once and the users have only one chance to get the required result. Obviously he RLMS allow the users to post another modified request in future, so the users can always get accurate results under the desired environment variables. By forcing each instruction set in a queue and executing them successively, the utilization of the resources is optimal. The instruments are never idle. There is no scope of internal or external fragmentation of time. But it is inapplicable for experiments that are fast and interactive.

Several attempts have been made to combine the two methods to exploit the advantages of each and overcome their disadvantages. In [16] a mechanism that is adaptive and can be used to optimize the usage of instruments depending upon the number of users to be supported and the nature of and time taken for the experiment is presented. By setting the parameters of timeslot length and number of users in it, this method can be used for almost all kinds of experiment.

The viability of RALs to support laboratory related experiences in Massively Open Online Courses (MOOCs) has been discussed in [246]. The parameters for determining scalability were: student numbers, laboratory activity duration, average laboratory sessions per student and the usage window of an experiment in a day. While these parameters are relevant for MOOC based laboratory systems, they do not provide a solution to the operational management of distributed P2P laboratories based on the students' immediate learning requirements or interests.

In a P2P RAL, users are typically unable to start an experiment session at any random time, as the limited capabilities, time and resource constraints of the makers must also be taken into account. The server side flexibility and unpredictable availability results in scarcer experiment availability where management of assigning experiments to users must be further optimized.

13.3 Suitable method for P2P RAL for STEM

STEM experiments have a number of unique characteristics compared to typical RALs used by tertiary undergraduate students. Experiments are typically composed of actuators and sensors that have visible locomotion as opposed to any integrated measurement system. The STEM RAL aims to develop the students' basic concepts

of science and technology by relating or representing them to real-world objects i.e. visible or audible phenomenon. The RAL for undergraduate experiments on the other hand may rely on raw measurement data (in text form) from a setup that does not change over the experiment duration.

Thus, the *time reservation* mode is most suitable for the STEM based laboratory because most STEM based experiments are highly interactive. Students require acute understanding of the experiment running by viewing or hearing the events in real time. Thus sufficient time must be allocated to each session.

It may be noted that queuing may also be successfully implemented, but this section focuses on the time reservation mode.

The regular time scheduling scheme followed by a centralised mechanism cannot be directly applied in the P2P RAL as the user are not expected to provide 24x7 service. A basic solution to the problem can be obtained by simply extending the time scheduling to two sets of users. This approach requires the following steps:

1. Hosts creates time block suitable for them during which the experiments will be available.
2. Each time block is then divided into slots depending upon the estimated average time for completion of the experiment.
3. The users book the slots within this time slots as suitable to them.

This approach suffers from the three major deficiencies. First, the time for which the experiments will be turn online has to be minimized. But with no information on when or how many users will be using the experiment, the makers may have to keep their equipment for long periods of time or short intermittent period without anyone actually using them. Second, while some users may get their best choice of experiments, others may get only the experiments that are not allotted already although they want something else. This may result in irregular learning pattern.

Also, it reduces the time for which the experiments are available to the users. If users do not turn up in their allotted slot it creates a lag in time for all users.

13.4 Identifying Constraints for Experiments and Users

The use of quest-based learning [33] in the context of RALs has an impact on scheduling strategies. Different types of incentives [33] in the quest based learning approach provides for a variable user requirements and interest in experiments available. Generally quests, related to the users' field of interest or courses, will be used to guide them through the relevant experiments within a short time period of a few days in order to gain experience points and badges within the quest game system. Therefore experiment clusters relating to quests are important to a particular user for that period of time.

The overall satisfaction score (W) of users in the system can be defined as the average of all users' satisfaction relating to what experiment they can perform in the duration of the next time period. This is based on the importance of the quest/experiment the user performs in a particular period of time with respect to their immediate goal (for example, to obtain a badge) in the game.

Why are experiments only available for a limited time?

The distributed nature of P2P RAL enables users to become makers of resources. But individual makers cannot guarantee or be obliged to serve each and every users request. The makers of experiment may wish to provide the experiments online periodically for two main reasons. First, the experiments are heavily dependent on video feedback for the entire duration of the experiments session. This video feedback, despite the best compression mechanisms, will consume a large amount of Internet data. Apart from that the rigs themselves may consume large amount of data. Periodic availability will ensure lower or uniform consumption of data over constant period of time compared to 24/7 availability. Second, the experiment may require vigilance during operation as these are prepared by individuals and could fail during operation.

Thus a limited exposure of the experimental rig will ensure lower cost and higher longevity of the rigs. The assumptions with regards to user-maker time scheduling in this work are that the users will be available for any actual duration in real time for which the experiments are online. This may be done through negotiations between the user and the maker. Another assumption is that the maker is able to keep their rigs

online according to the user demands at all reasonable time periods but would like to limit the actual time due to reasons specified earlier.

Input Parameters

Following are the inputs required by the system in order to perform the assignment of time slots to the users:

1. *Total Time Period*: In the RAL system makers will keep their rigs online for a particular period of time within a larger window of time. For example, 2 hours every week, where a week is the time period (T) and 2 hours is the value for d_e.

2. *The total number of experiments available* (n): This is the total of different experiments that are available in the next time period. Each experiment will have a duration for which the experiment will be available online d_e in T as fixed by the user.

3. *User Preference*: To increase the efficiency of the assignments between the users and makers, the users are asked about the preferred experiments. Each user rates n number of experiments according to preference that they would have time to do in the next T. Each user is however allowed to do only one among these experiments. The total number of users is U.

Cost and Choice values of User-Experiment relation

Once the users select the experiment there are $U \times n$ number of relations formed between each user and the experiments. Whilst calculating the assignments using the TAP solution the cost values *c(u, e)* for user u and experiment e is used and while calculating the satisfaction score (W), the original choice or priority values as considered. A *choice* or *priority* value, *i(u, e)*, for an experiment e is the users' entry to the system when they choose the priority of the experiments in order 1, 2, 3 ... n where a lower number indicates higher preference. The *cost c(u, e)* is then based upon the priority, depending upon the users' condition in the system compared to others. The user-experiment relationship may be valued as follows:

1. *Polynomial*: The costs may be assigned in quadratic or following a polynomial function For example

$$c(u, e_i) = f(i(u, e)) \quad \ldots(13.1)$$

where $i(u, e)$ is the priority value of the choice value that ultimately determines the value of the cost $c(u, e)$. $i(u, e)$ is a subset of the natural numbers 1, 2, 3, ..., n.

This gives different levels of priority to user-experiment relations and makes relevant experiments more important than the others compared to the linear approach. These may relate to other factors relating to assignments in the previous rounds. For example if some user is unassigned in the previous round due to unavailability, others' costs may be increased compared to them, so that while minimizing this particular user gets a better chance of getting the satisfactory allocation. These will lead to *polynomial* type relations.

2. *Linear*: The relationships are *priced* linearly i.e.

$$c(u, e) = i(u, e) \quad \ldots (13.2)$$

In a linear case no consideration is given to the users' condition in the prior assignments and the cost values for assignment is equal to the current priority of the experiment for the user.

The user-experiment relationship represents the importance of the experiment to the user with respect to the users' goal in the game based learning system. This relationship may not necessarily be determined by the students themselves but could account for other factors as well.

13.5 Matching of Users and Makers

This section discusses how the scheduling problem in P2P RAL can be described as a *Terminal Assignment Problem*, its solution and ways to use it with the P2P RAL.

Formulating the TAP

The scheduling problem can be described as a Terminal assignment problem as follows:

1. There are users (or *terminals*) that are to be assigned to the experiment sessions.

2. Each user has the equal weight (q_u) of 1 that is the number of experiment

they can perform in the next time period.

3. Each experiment has an average usage time t_e associated with it. The number of time the experiment can be done in the next time period T is obtained by using the makers available time d_e for an experiment as

$$a_e = \frac{d_e}{t_e} \quad \ldots (13.3)$$

Thus there are a_e copies of the experiment e that are to be assigned.

4. The total number of experiments sessions (E) comprising all experiments' sessions s, that can be performed in the next time period T is

$$E = \sum_{e=1}^{n} \frac{d_e}{t_e}$$

The total number of users that may be accommodated

$$U \leq E$$

5. The experiments (or *contractor*) are the other set of nodes to which the users are assigned to. The capacity of each experiment session (a_e) is variable.

6. The cost $c\ (u, e)$ between each user u and an experiment e is the based on Equation 13.1 or Equation 13.2 which represents the preference values of the experiments for the users.

Figure 13.1 shows as example of the scenario with 7 users and 3 active experiments, where a_e is the number of experiment session that may be run in the next time available period. In a sample analysis, for experiment 1, let this value be 3, and for experiment 2 the value is 1 and for experiment 3 it is 5. The users u_1 to u_7 have all requested the 3 experiments with their preferred choices. Users u_1 and u_2 have chosen experiment 1 as their first choice, u_3 and u_4 have chosen experiment 3 as their first choice while users u_5, u_6 and u_7 has chosen experiment 2 as first choice. Clearly, not all of them can be assigned to their first choice experiments in the next time period T.

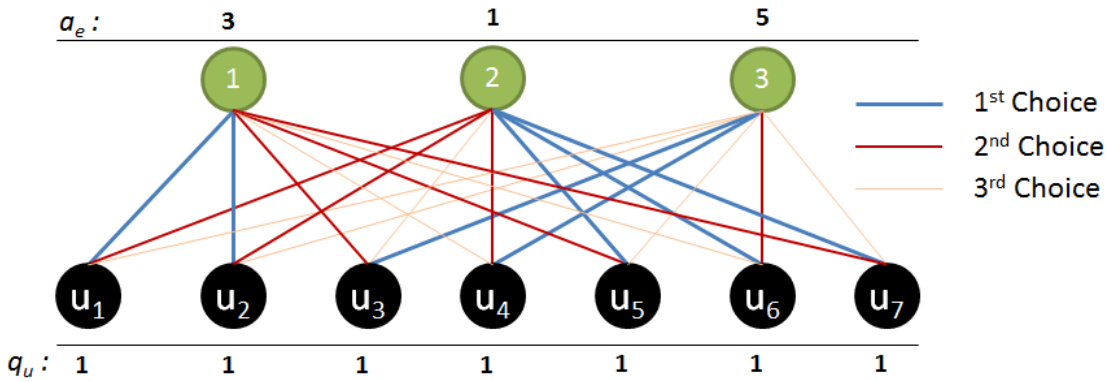


Figure. 13.1. An Example Scenario.

Solving this terminal assignment problem will assign the user with the lowest edge values i.e. the highest preference or lowest possible choice values for each users such that the

$$P = \left(min \sum_{u \in U,\ s \in E} c(u,s) \right)$$

and the satisfaction score (W) is then obtained as dividing the sum of all assignments choice values (P) by the number of users (|U|),

$$W = P \ / |U| \qquad \ldots (13.4)$$

The solution to the TAP will make sure that every user gets their most preferred choice as possible maintaining an optimized satisfaction level. One optimal solution to the example in Figure 13.1 is $\{u_1, u_2, u_5\} \rightarrow e_1$, $\{u_3, u_4, u_6\} \rightarrow e_3$ *and* $\{u_7\} \rightarrow e_2$. The lesser choices for u_5 and u_6 is because, they along with u_7 opted for e_3 which is available for only one time slot during the next T.

The TAP in general terms for variable requirements weights of the *terminals* and the variable capacities of *concentrators* are *NP-Hard* problems. However, for TAP with equal weights (q_u) for all *terminals* as in this case the augmenting path algorithm can solve the problem in polynomial time an provide a correct solution [247, 248]. The TAP solution implemented is based on the algorithm described in [248]. It splits the concentrators or each experiments session into individual nodes with capacity = 1.

Using the Predictor Model in P2P RAL

In P2P RAL, the global management server or the distributed P2P RLMS is

responsible for executing the assignment algorithm. The duration of a time period T is taken as a week or 7 consecutive days. The scheduling is done in the follow the steps:

Algorithm 13.1

1. The maker enters the d_e for the experiment in the P2P RAL system
2. Using equation 13.3, the value for a_e may be obtained as per t makers wishes and the value for t_e of their experiment.
3. The maker also inputs certain lengths of time for the availability of the experiment in the week.
4. If a maker does not wish to keep their experiment online at all, then the value d_e is marked 0. The list of experiments provided to users is the experiments with non-zero d_e values.
5. The users then assign the priority or choice values $i(u, e)$ from 1, 2, 3, ... n, for the available experiments.
6. After this the P2P RAL system does the assignment procedure and informs the user about the experiment that has been assigned.
7. The users then book a time slot from the makers list.

A specific function for determining the actual cost of each user-experiment relation $c(u, e)_t$ for a given time period t is deployed for an implementation to balance the users satisfaction (see Equation 13.5). This function however may vary from system to system depending upon how the users are related to their experiment. Very important user-experiment relations, For example an experiment that must be done by a user in the next Time period, may be pre-scheduled before starting to assign the experiments.

13.6 Implementation and Simulation

Assuming that a rig may be created by a group of up to 5 students, the example presented here considers a total of 20 experiments. The 20 experiments are to be kept online at different rates a_e depending upon the availability and feasibility of the hosting sites. The a_e is generated as a random number between 2 and 5. The number

of users is considered equal to the total number of experiment sessions ($U = E$) available. As discussed earlier, makers of an experiment are not obliged to accommodate every user that may be in the system. But, if makers are aware of the number of users in the system, they will keep the values of *de* accordingly so that U ≤ E always. The worst-case scenario of $U = E$ is considered here.

It may be assumed that these students belong to a certain cohort, say 10th standard students from 3 schools. Each of these users are also part of a small group of peers that creates at least one of the 20 rigs and hosts it. Each student will run all the 20 experiments personally as part of their quests in the game during a year. The RAL system now must assign the users (or students) with the experiment of their best choice. Choices for any experiment for any user will be depended on their requirements regarding the *class work* or their motivation in selecting quests related to their field of interest, all of which are variable with time.

Time period (T) is considered to be a week. Hence, at least 20 weeks will be required for all students to complete each of the experiment, if all users choose unique experiments each week as their first choice. For each user, a random priority values is generated for each week in the simulation. The experiments that have already been done are not expected to be done again. Thus, these experiments have no priority at all. The priority (or choice *i(u, e)*) of each user always starts with the value of 1and then vary according to the number of experiments that has been completed. The user's cost for any experiment *e* in for the time period T is given by,

$$c(u,e)_t = (i(u,e))^{1+y} \qquad \dots \ (13.5)$$

for,

$$y = CompletedExperiments(u)_t - \min_{x \in (U - D_t)} \{CompletedExperiments(x)_t\}$$

where, *i(u, e)* is the priority of the experiment assigned to the user *u* in week *j*. $CompletedExperiment(u)_t$ is the number of experiments that have been completed by user *u* by week *t*. D_t is a set of users who have completed all experiments by week *t*. Note that, from Equation 13.1 and Eq.13.4, the choice or priority *i(u, e)* must be minimized and the lower the value of choice, the higher the actual priority of the experiment for the user.

This ensures that if a user is unassigned for a week, then all other users' costs are increased to give the unassigned user a fair chance in the next week. The costs associated with the user-experiment relationship is thus dependent on the satisfaction score (*W*) in previous rounds of experiments assignments. The user array is randomly changed in the system to simulate the different orders the users may come in to the system.

The simulation measures the satisfaction score for each week which must be low to ensure good assignments and the time taken for completing experiments. While calculating W, if an user is unassigned in week *j*, the $i(u)_j$ is assumed to be equal to n (=20) to indicate total dissatisfaction. Thus the satisfaction score for assigned users in week *t* is

$$W_A^t = \left(\sum_{a \in A} i(a)_t \right) \Big/ |A| \qquad \ldots (13.6)$$

where $i(u)_t$ is the choice value of the experiment assigned to user u for week *t* and *A* is a set of active users who have been assigned to an experiment in week *t*, and Thus the satisfaction score for all incomplete users $(S = U - D_t)$ in week *t* is

$$W_S^t = \left(\sum_{a \in S} i(a)_t \right) \Big/ |S| \qquad \ldots (13.7)$$

For calculating W, the original choice number $i(u)_t$ is used instead of the modified cost of user-experiments relationship *c(u)*.

13.7 Results and Conclusions

Figure 13.2 to 13.5 shows the results where $U = E = 74$. Figure 13.7 shows the a_e assigned to each of the 20 experiments. Note that the three experiments with $a_e = 2$ have to be done for at least 37 weeks because with the current system only two users can use them in a week.

The allocation of experiments is done based on the user choices. For all users, the number of weeks taken for completing 20 experiments is between 20 to 37 weeks (as shown in Figure 13.3) with a mean of 26.7 weeks and a standard deviation of 5.7 weeks.

Figure 13.4 and 13.5 show the simulated performance of the scheduling scheme. The

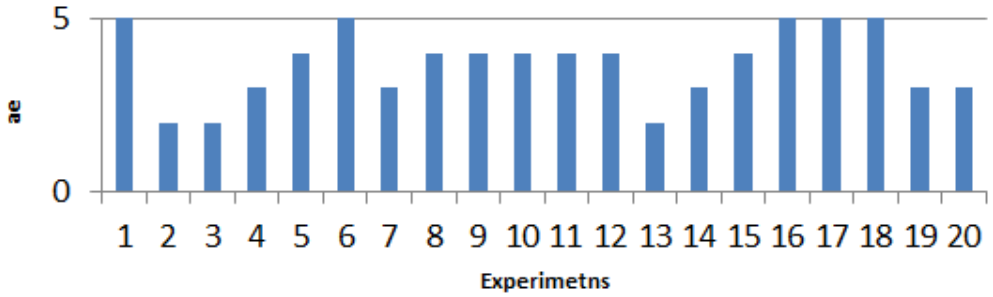


Figure.13.2. The a_e set for each experiment.

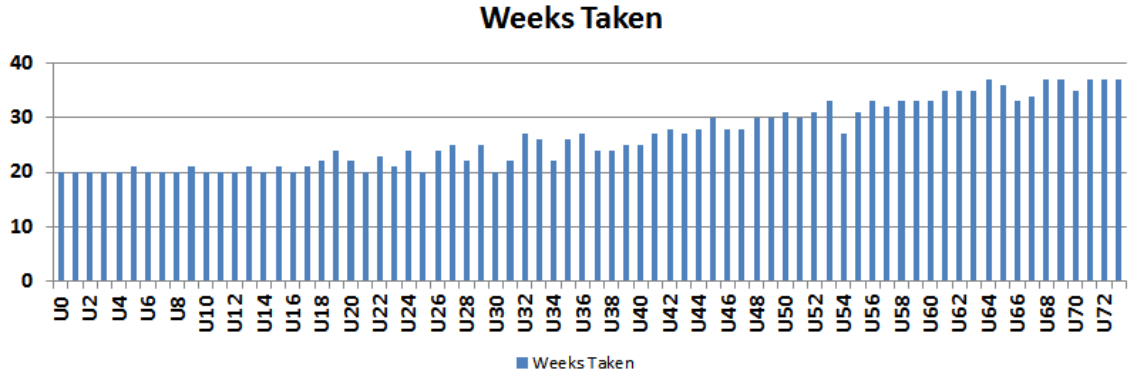


Figure.13.3. The completion time of all users.

observations can be summarised as follows.

The number of active users i.e. the users which have been assigned to at least one experiment in the next week starts with 74 and remains so for the first few weeks after which they start to fall as experiments that are available for more slots are already done by the users and the scarcer experiments resources are now assigned as much as possible.

Likewise the Satisfaction score (W_A for active users) is very low (< 2) for all weeks which indicates good assignments. The W_S, considering all users left (U - D_t) with D_t number of users having completed all the experiments, is similar to W_A for the first few weeks after which, it increases. As experiments start becoming scarcer, the algorithm makes compromises for every assignment in the smallest amount so that no one is disadvantaged. As there are more clashes of choices between the users after the initial periods, many users are unassigned for the following weeks largely increasing the values of W_S. The number of users completing all experiments by any given week increases since week 20.

After week 15, the availability of required or active experiments drops drastically as shown in Figure 13.4. The users who still need to do experiments declines after week

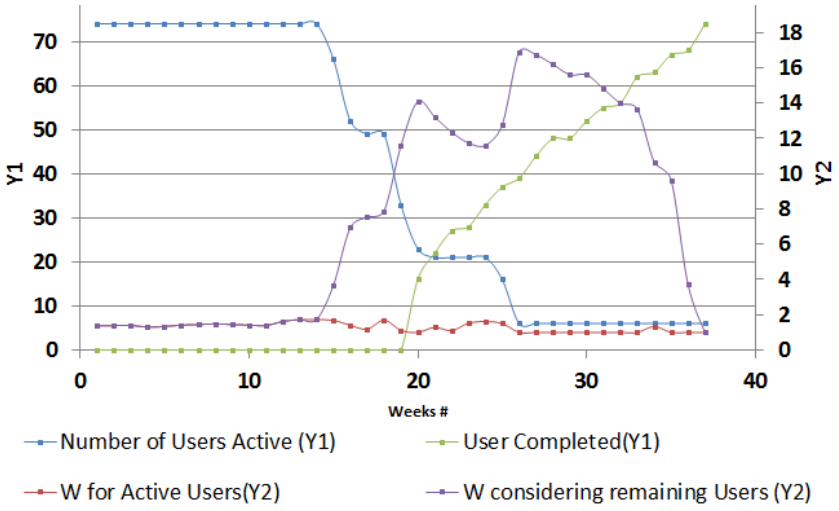


Figure. 13.4. The average Satisfaction Score (W) of all users in every week.

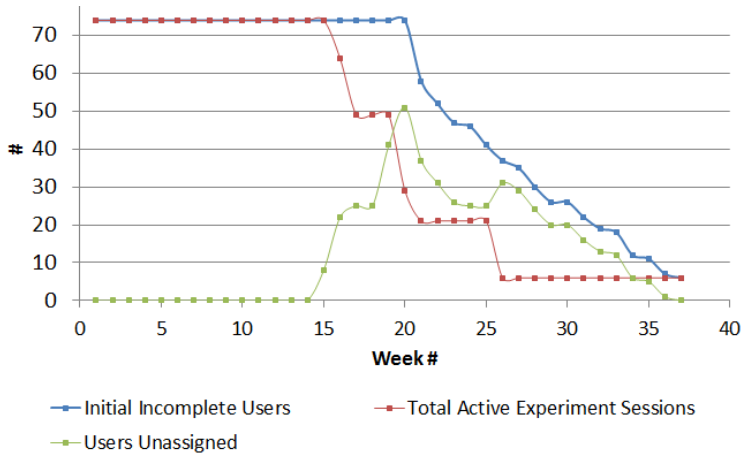


Figure. 13.5. The incomplete, unassigned users in every week.

20, but the number of active experiments drops earlier causing a large number of users remaining unassigned for the later weeks (> 15). The number of unassigned users again drops when experiments become more available compared to the number of users left. No user is assigned the same experiment more than once.

This shows that the proposed scheme can judicious assign the experiments to the users, such that all users have completed their experiments in the quickest possible way.

13.8 Summary

This chapter has discussed the scheduling problem in the remote access laboratory with a P2P approach where the makers have limited capabilities for hosting remote

experiments. The RAL experiments are set up as a quest-based hierarchy where users interests could widely affect the experiments they would like to perform. With limited resources in a P2P setup the users must be judiciously assigned to the most required experiments. Using the proposed scheme based on the solution to terminal assignment problem, users are accommodated as best as possible to the experiments depending upon the user-experiment relationships.

14

Conclusions and Future Work

This research has shown that a P2P RAL can be an alternative approach for creating distributed remote laboratories. Such a system can potentially provide high scalability and engage students in more hands-on-experience. The main aim of this research was to develop tools to enable makers to create and host an experiment based on everyday objects along with a microcontroller. It also focused on ensuring users accessibility in the P2P RAL to communicate to and control the experiment created by makers. A VPN overlay network is used to implement the network requirements. The P2P RAL architecture conceptually establishes point-to-point connections between makers and users. Reliability issues have been discussed and a method to measure reliability was presented. This method can be used to monitor the condition of experiments and prevent failure of accessibility of the experiment for users.

In terms of WoT or IoT, the CI-CU model presents a generic model that may be applied to any scenario where a supervisory system is required to monitor multiple master-slave combinations in a system. The semantics used for the P2P is unique to the control commands used here. But, the semantics may be altered and enhanced to be application specific for WoT or IoT systems.

In terms of control strategies, the model presented is largely master-slave. While the CI can send multiple and various types of messages to the CU, the CU is capable of only sending a fixed set of messages. However, this model can be further extended to include different messages being transferred from the CU to the CI instead of only a select few as described here. This gives greater flexibility in design, but brings in new issues with synchronization of the CI and CU.

In terms of MDP based modelling of the experiment, such a system is also applicable in IoT application where the devices have well defined states and transitions between them. The CU is capable of automatically creating the MDPs and uses them to identify user actions. A major challenge is to create the proper interface where the makers can manage the MDPs and their properties.

Clustering mechanisms were used to create various levels of composite commands. This also creates a profile of how an experiment is used and how it is accessed by the users. These profiles can be used to compare the different user sessions. Further work in this regard can look into identifying composite commands with advanced features such as conditional checks within them. The use of both these learning analytics tool can be used and modified to monitor users' performance in RALs and other similar systems.

Another area to expand the P2P tools is in pattern recognition possible with artificial intelligence. The MDP based experiment model can only evaluate the interactions based on the current and next state. The advanced evaluation tools based on clustering can only match two interaction sessions in a relative manner. The next level is to determine whether, the users' interaction has a sequence of state changes that matches the makers' state changes when the command sequence may not be equal. In terms of IoT, this will help identify similar patterns in the behaviour of master-slave communication when the semantics are different for different pair.

The proposed CI-CU model covers a wide variety of experiments, but is limited to the experiments that can be implemented with the port based architecture. For the P2P RAL for STEM Education this is sufficient. However, future works can identify other models and compare or relate them to the proposed model here.

The tools developed as a part of the P2P RAL can enhance STEM education. The P2P RAL architecture is suitable for the STEM subject as the teacher and students have little support with technical resources and knowledge. However, the practical knowledge of how to create the experiment setup and learn to measure and collect data is important for learning outcomes. If this practice is employed from an early age it could potentially grow interest in the STEM fields in higher studies. A graphical programming language is used that helps the users/makers to create the program easily. Such graphical programs have been used to teach programming at schools.

This research showed that certain aspects of programming such as ports and delays are possible to be explained to the makers in the target group of RAlfie and incorporated in the learning experience while creating the rig. The future work includes improving the tools for more practical use with better human usable interfaces as well as developing pedagogies for using the P2P RAL in the field. Also, the time scheduling algorithm presented is shown to be capable of handling the requirements of the P2P RAL, but it may need to be further improved in a real life scenario.

From a remote laboratory perspective, the main contribution of this thesis is the *white box* perspective against a traditional *black box* approach. Most experiments in traditional remote laboratories used for undergraduate or STEM education rely on human evaluators and developers to create, host, evaluate and monitor experiments. The white box approach proposed in this dissertation does not eliminate the need of human evaluation and maker roles, but greatly reduces the reliance on their capabilities to create the perfect experiment setup along with user interfaces. The tools discussed in here enable makers to create and host experiment with minimum deliberation. The P2P RAL aims to exploit these features to enable wide scale sharing and collaboration among the target users.

The most important research aspects that need to be addressed are the ways the tools can be used with respect to the context they are used in. The MDP and clustering algorithm offers methods to calculate the difference in user/maker interaction, but exactly how much deviation is acceptable for the application e.g. STEM education needs to be established from the relevant context. This research has focused on creating the tools and proving their usability only. Further research can also look into enhancing the performance of these tools or propose new ones. Also, the CI-CU model while being capable of providing a model for a large number of experiments, an improved model may be develop to incorporate experiments with more unique features.

References

[1] L. Johnson, S. Adams Becker, M. Cummins, et. al., "NMC Horizon Report: 2016 Higher Education Edition", Austin, Texas: The New Media Consortium, p.g. 18.

[2] J. García-Zubía and G. R. Alves (eds.), "Using Remote Labs in Education", University of Deusto, Bilbao, 2011.

[3] L. Gomes and J. Garcia-Zubia, Eds., "Advances on Remote Laboratories and e-Learning Experiences", University of Deusto, 2007.

[4] T. Kostulski and S. Murray, "Student Feedback from the First National Sharing Trial of Remote Labs in Australia," REV 2011, pp.203-211, 9, 2011

[5] A. Maiti, A. D. Maxwell and A. A. Kist, "An Overview of System Architectures for Remote Laboratories," in TALE 2013, pp. 661-666, 2013.

[6] M. Tawfik, E. S. Cristobal, A. Pesquera, R. Gil, S. Martin, G. Diaz, et al., "Shareable educational architectures for remote laboratories," in Technologies Applied to Electronics Teaching (TAEE), 2012, 2012, pp. 122-127.

[7] H. M. A. Andree, J. Habets, M. Koopmans, W. Koopmans, G. Kemmerling, M. Korten, et al., "Virtual control room, the REMOT project, networking pilot studies," IEEE Transactions on Nuclear Science, vol. 45, pp. 1999-2003, 1998.

[8] I. Mougharbel , A. El Hajj , H. Artail and C. Riman, "Remote lab experiments models: A comparative study", Int. J. Eng. Educ., vol. 22, no. 4, pp. 849-857, 2006

[9] V. J. Harward, J. A. Del Alamo, et al., "The iLab shared architecture a web services infrastructure to build communinities of Internet accessible laboratories," Proceedings of the IEEE, vol. 96, pp. 931-950, Jun 2008.

[10] D. Lowe, S. Murray, E. Lindsay, and D. K. Liu, "Evolving Remote Laboratory Architectures to Leverage Emerging Internet Technologies," IEEE Transactions on Learning Technologies, vol. 2, pp. 289-294, 2009.

[11] J. Garcia-Zubia, P. Orduna, et al., "Application and User Perceptions of Using the WebLab-Deusto-PLD in Technical Education," FIE 2011.

[12] M. Tawfik, E. Sancristobal, S. Martin, R. Gil, G. Diaz, A. Colmenar, et al., "Virtual Instrument Systems in Reality (VISIR) for Remote Wiring and Measurement of Electronic Circuits on Breadboard," IEEE Transactions on Learning Technologies, vol. 6, pp. 60-72, 2013.

[13] I. L. Hardison, K. DeLong, P. H. Bailey, and V. J. Harward, "Deploying Interactive Remote Labs Using the iLab Shared Architecture," Fie: 2008 Ieee Frontiers in Education Conference, Vols 1-3, pp. 1246-1251, 2008.

[14] M. Schulz, A. Rudd and L. Payne, "RESTlabs: Service broker architecture for remote labs," Remote Engineering and Virtual Instrumentation (REV), 2012 9th International Conference on , vol., no., pp.1,6, 4-6 July 2012, doi: 10.1109/REV.2012.

[15] J. Del Alamo, L. Brooks, C. McLean, J. Hardison, G. Mishuris, V. Chang, et al., "The MIT microelectronics WebLab: A web-enabled remote laboratory for microelectronic device

[16] A. Maiti, "A hybrid algorithm for time scheduling in remotely triggered online laboratories," in Global Engineering Education Conference (EDUCON), 2011 IEEE, 2011, pp. 921-926.

[17] G. N. Masters, "A shared challenge: Improving literacy, numeracy and Science Learning in Queensland Primary Schools," Australian Council for Educational Research, Camberwell, Vic2009.

[18] S. Thomson, N. Wernert, C. Underwood, and M. Nicholas, "TIMSS 2007: Taking a closer look at mathematics and science in Australia," Camberwell, Vic2008.

[19] P. J. Fensham, "Science education policy-making: Eleven emerging issues," UNESCO, Paris, France2008.

[20] K. Macpherson, "Digital technology and Australian teenagers: consumption, study and careers," The Education Institute, Canberra, ACT2013.

[21] http://www.innovation.gov.au/page/agenda

[22] L. D. Feisel and A. J. Rosa, "The role of the laboratory in undergraduate engineering education," Journal of Engineering Education, vol. 94, pp. 121-130, 2005.

[23] L. Johnson, "The future of education: The 2013 NMC Horizon Project Summit communique," New Media Consortium, Austin, TX2013.

[24] A. A. Kist and P. Gibbings, "Inception and management of remote access laboratory project," in 21st Annual Conference of the Australasian Association for Engineering Education, Sydney, Australia, 2010, pp. 5-8.

[25] V. J. Harward , J. A. del Alamo , V. S. Choudary , K. DeLong , J. L. Hardison , S. R. Lerman , J. Northridge , C. Varadharajan , S. Wang , K. Yehia and D. Zych, "iLabs: A scalable architecture for sharing online laboratories", Int. Conf. Eng. Educ. 2004, 2004.

[26] D. Lowe, P. Newcombe, and B. Stumpers, "Evaluation of the Use of Remote Laboratories for Secondary School Science Education," Research in Science Education, vol. 43, pp. 1197-1219, 2013/06/01 2013.

[27] A. A. Kist, A. Maxwell, P. Gibbings, R. Fogarty, W. Midgley, and K. Noble, "Engineering for primary school children: Learning with robots in a remote access laboratory," presented at the

SEFI 2011: Global Engineering Recognition, Sustainability and Mobility, Lisbon, Portugal, 2011.

[28] A. Maiti, A. D. Maxwell, and A. A. Kist, "Features, Trends and Characteristics of Remote Access Laboratory Management Systems," International Journal of Online Engineering, vol. 10, pp. 31-37, 2014.

[29] G. Eysenbach, J. Powell, M. Englesakis, C. Rizo, and A. Stern, "Health related virtual communities and electronic support groups: systematic review of the effects of online peer to peer interactions," BMJ, vol. 328, p. 1166, 2004-05-13 21:55:00 2004.

[30] S. J. H. Yang and I. Y. L. Chen, "A social network-based system for supporting interactive collaboration in knowledge sharing over peer-to-peer network," International Journal of Human-Computer Studies, vol. 66, pp. 36-50, 1// 2008.

[31] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim, "A Survey and Comparison of Peer-to-Peer Overlay Network Schemes," Ieee Communications Surveys and Tutorials, vol. 7, pp. 72-93, 2005.

[32] L. Orwin, A. A. Kist, A. D. Maxwell and A. Maiti, “Using Gamification to Create Opportunities for Engagement, Collaboration and Communication in a Peer-to-Peer Environment for Making and Using Remote Access Labs”, exp.at’15 — Online Experimentation, Portugal.

[33] C. C. Haskel, “Design Variables of Attraction in Quest-Based Learning”, May 2012.

[34] D. Lowe, "Integrating Reservations and Queuing in Remote Laboratory Scheduling," Learning Technologies, IEEE Transactions on , vol.6, no.1, pp.73-84, First Quarter 2013, doi: 10.1109/TLT.2013.5

[35] L. Yaoye, S. K. Esche, and C. Chassapis, "A scheduling system for shared online laboratory resources," in Frontiers in Education Conference, 2008. FIE 2008. 38th Annual, 2008, pp. T2B-1-T2B-6.

[36] Z. Nedic, J. Machotka, and A. Nafalski, "Remote Laboratory NetLab for Effective Interaction with Real Equipment over the Internet," Human System Interactions, Conference on, Vols 1 and 2, pp. 858-863, 2008.

[37] T. Fischer and J. Scheidinger, "VISIR - Microcontroller extensions," in Remote Engineering and Virtual Instrumentation (REV), 2015 12th International Conference on, 2015, pp. 177-179.

[38] Samuelsen, D. A H; Graven, O.H., "Virtual fences as protection against damage on physical equipment used in remote laboratories," REV 2012 9th International Conference on , 4-6 July 2012.

[39] P.Orduna, "Transitive and Scalable Federation Model for Remote Laboratories", PhD Thesis, May 2013.

[40] M. Auer, A. Pester, D. Ursutiu, and C. Samoila, "Distributed virtual and remote labs in engineering," IEEE Int. Conf. on Ind. Technology, pp. 1208-1213 Vol.2, 2003.

[41] K. Henke, S. Ostendorff, H.-D. Wuttke, and S. Vogel, "A grid concept for reliable, flexible and robust remote engineering laboratories," Int. Journal of Online Engineering (iJOE), vol. 8, pp. 42-49, 2012.

[42] J. García-Zubia, P. Orduña , I. Angulo, et. al., "Towards a Distributed Architecture for Remote Laboratories," Int. Journal of Online Engineering (iJOE), vol. 4(S1), pp. 11-14, 2008.

[43] M. Diponio, D. Lowe, and M. de la Villefromoy, "Supporting Local Access to Collections of Distributed Remote Laboratories," in AAEE 2012 Conference, Melbourne, Australia, 2012.

[44] H. Wenshan, L. Guo-ping, and Z. Hong, "Web-Based 3-D Control Laboratory for Remote Real-Time Experimentation," IEEE Trans. on Ind. Electron., vol. 60, pp. 4673-4682, 2013.

[45] C. Salzmann and D. Gillet, "Smart device paradigm, Standardization for online labs," in IEEE Global Engineering Education Conference (EDUCON), 2013, pp. 1217-1221.

[46] M. Tawfik, C. S., D. Gillet, D. Lowe, et. al., "Laboratory as a Service (LaaS): a Model for Developing and Implementing Remote Laboratories as Modular Components", in Int. Conf. on Remote Engineering and Virtual Instrumentation (REV) 2014, pp.11-20, 2014.

[47] L. Tobarra, S. Ros, R. Pastor, R. Hernandez, M. Castro, A. Al-Zoubi, et al., "Laboratories as a service integrated into learning management systems," in 2016 13th International Conference on Remote Engineering and Virtual Instrumentation (REV), 2016, pp. 103-108.

[48] A. C. Caminero, A. Robles-Gomez, S. Ros, L. Tobarra, et al., "Deconstructing remote laboratories to create Laboratories as a Service (LaaS)," in 2014 IEEE Global Engineering Education Conference (EDUCON), 2014, pp. 623-629.

[49] Y. Wei-feng, S. Rong-gao and W. Zhong, "Distributed Remote Laboratory using Web Services for Embedded System," in Proceedings of the 3rd WSEAS Int. Conf. on Circuits, Systems, Signal and Telecommunications (CISST'09), pp. 56-59, 2009.

[50] D. Ursutiu, D. T. Cotfas, M. Ghercioiu, et. al., "WEB Instruments," in IEEE Education Engineering (EDUCON) 2010, 2010, pp. 585-590.

[51] R. J. Costa, G. R. Alves, and M. Z. Rela, "Embedding Instruments & Modules into an IEEE1451-FPGA-based Weblab Infrastructure," Int. Journal of Online Engineering, vol. 8, 2012, pp 4-11.

[52] E. Bogdanov, C. Salzmann, and D. Gillet, "Widget-Based Approach for Remote Control Labs," IFAC Symposium on Advances in Control Education, 2012.

[53] D. Gillet, "Personal learning environments as enablers for connectivist MOOCs," in Information Technology Based Higher Education and Training (ITHET), 2013 Int. Conf. on, 2013, pp. 1-5.

[54] A. Melkonyan, A. Gampe, M. Pontual, et. al., "Facilitating Remote aboratory Deployments Using a Relay Gateway Server Architecture," IEEE Trans. on Ind. Electron., vol. 61, pp. 477-485, Jan 2014.

[55] P. Orduña and J. Garcia-Zubia, "Scheduling schemas among Internet Laboratories ecosystems", REV 2011, Brasov, Romania, July 2011.

[56] M. F. Schulz and A. Rudd, "Design and Implementation Issues in a Contemporary Remote Laboratory Architecture", in Remote Engineering and Virtual Instrumentation Conference 2011, Brasov, Romania, pp. 380-384, 2011.

[57] A. V. Fidalgo, G. R. Alves, et. al., "Using remote labs to serve different teacher's needs A case study with VISIR and RemotElectLab," REV2012,pp.1,6, 4-6 July 2012, doi: 10.1109/REV.2012.6293149

[58] A. A. Kist, A. Maxwell, P. Gibbings, R. Fogarty, W. Midgley, and K. Noble, "Engineering for primary school children: Learning with robots in a remote access laboratory," 1st World Engineering Education Flash Week, SEFI Annual Conference (European Society for Engineering Education), 2011.

[59] L. Bowtell, C. Moloney, A. A. Kist, V. Parker, A. Maxwell, and N. Reedy, "Using remote access laboratories in nursing education," Proceedings of the 9th International Conference on Remote Engineering and Virtual Instrumentation (REV 2012), pp. 1-7, 2012.

[60] L. A. Bowtell, C. Moloney, A. A. Kist, V. Parker, A. Maxwell, and N. Reedy, "Enhancing Nursing Education with Remote Access Laboratories," International Journal of Online Engineering (iJOE), vol. 8, 2012.

[61] A. A. Kist and B. Basnet, "Providing Equivalent Learning Activities with Software-Based Remote Access Laboratories," International Journal of Online Engineering (iJOE), vol. 9, pp. pp. 14-19, 2013..

[62] A. A. Kist, A. Maxwell, and P. Gibbings, "Expanding the Concept of Remote Access Laboratories " presented at the 119th ASEE Annual Conference and Exposition, San Antonio, Texas, 2012.

[63] G. R. Alves, et. al. "Using VISIR in a large undergraduate course: Preliminary assessment results," IEEE EDUCON 2011, pp.1125-1132, 4-6 April 2011.

[64] I. Santana, M. Ferre, E. Izaguirre, et al., "Remote Laboratories for Education and Research Purposes in Automatic Control Systems," IEEE Trans. on Ind. Informatics, vol. 9, pp. 547-556, 2013.

[65] L. Rodriguez-Gil, P. Orduna, L. Bollen, S. Govaerts, A. Holzer, et al., "The AppComposer Web application for school teachers: A platform for translating and adapting educational web applications," in Global Engineering Education Conference (EDUCON), 2015 IEEE, 2015, pp. 889-897.

[66] O. Dziabenko and J. García-Zubía, "Planning and Designing Remote Experiment for School Curriculum", Proceedings of the 6th IEEE Global Engineering Education Conference, EDUCON 2015, IEEE Computer Society, Tallin, Estonia, 2015.

[67] S. Freeman, S. L. Eddy, M. McDonough, M. K. Smith, N. Okoroafor, H. Jordt, et al., "Active learning increases student performance in science, engineering, and mathematics," Proceedings of the National Academy of Sciences, vol. 111, pp. 8410-8415, June 10, 2014 2014. M. W. Gilmore, "Improvement of STEM Education: Experiential Learning is the Key", Modern Chemistry & Applications, doi: 10.4172/2329-6798.1000e109.

[68] T. Richter, Y. Tetour and D. Boehringer, "Library of Labs - A European Project on the Dissemination of Remote Experiments and Virtual Laboratories," IEEE ISM 2011, pp.543-548, 5-7 Dec. 2011.

[69] Z. Nedic, J. Machotka, and A. Nafalski, "Remote Laboratory NetLab for Effective Interaction with Real Equipment over the Internet," Human System Interactions, Conference on, Vols 1 and 2, pp. 858-863, 2008.

[70] A. Nafalski, et. al., "International Collaboration in Remote Engineering Laboratories: an Approach to Development", REV 2011.

[71] A. Mujkanovic, D. Lowe, C. Guetl and T. Kostulski, "An architecture for automated group formation within remote laboratories", REV 2011, Brasov, Romania, pp. 91-100.

[72] J. A. Asumadu, R. Tanner, J. Fitzmaurice, M. Kelly, H. Ogunleye, J. Belter, et al., "A Web-based electrical and electronics remote wiring and measurement laboratory (RwmLAB) instrument," Instrumentation and Measurement, IEEE Transactions on, vol. 54, pp. 38-44, 2005.

[73] D. A. H. Samuelsen and O. H. Graven, "Design of a general purpose platform for easy setup of low-cost remote laboratories in electronics," in Remote Engineering and Virtual Instrumentation (REV), 2013 10th International Conference on, 2013, pp. 1-6.

[74] P. Garaizar, M. Á. Vadillo, and D. Lopez-de-Ipina, "Benefits and Pitfalls of Using HTML5 APIs for Online Experiments and Simulations," International Journal of Online Engineering (iJOE), vol. 8, 2012.

[75] D. Costa, G. Alves, P. Ferreira, J. Silva, "Remote Labs Accessible through 3D environments A Case Study with Open Wonderland," in REV 2011 Proceedings, pp. 191-196.

[76] B. Scheucher, P. H. Bailey, C. Gütl, and J. V. Harward, "Collaborative Virtual 3D Environment for Internet-Accessible Physics Experiments," International Journal of Online Engineering (iJOE), vol. 5, 2009.

[77] R. Marcelino, J. B. Silva, V. Gruber, and M. S. Bilessimo, "3D virtual worlds using open source platform and integrated remote experimentation," in REV 2012, Bilbao, 4-6 July 2012.

[78] J. Garci'a-Zubia, J. Irurzun, et.. al., "SecondLab: A remote laboratory under Second Life," Education Engineering (EDUCON), 2010 IEEE , vol., no., pp.351,356, 14-16 April 2010.

[79] T. Kostulski and S. Murray, "Student Feedback from the First National Sharing Trial of Remote Labs in Australia," REV 2011, pp.203-211, 9, 2011.

[80] Lindsay, E.D. and Murray, S.J. and Lowe, D.B. and Tuttle, S.W., "Derivation of Suitability Metrics for Remote Access Mode Experiments", in REV 2010, Stockholm, Jun 29 2010.

[81] E. D. Lindsay and M. C. Good, "Effects of laboratory access modes upon learning outcomes," IEEE Transactions on Education, vol. 48, pp. 619-631, Nov 2005.

[82] Z. Nedic, J. Machotka and A. Nafalski, "Enriching student learning experiences in remote laboratories", 2nd Annual Conference on Engineering and Technology Education, pp. 9-14, January 2011.

[83] C. M. Paiva, P. Nogueira, et. al., "A Flexible Online Apparatus for Projectile Launch Experiments," iJOE, vol. 9, 2013.

[84] J. García-Zubia, U. Hernandez-Jayo, et al., "LXI Technologies for Remote Labs: An Extension of the VISIR Project," iJOE, vol. 6, 2010.

[85] M. J. Callaghan, J. Harkin, M. El Gueddari, T. M. McGinnity, and L. P. Maguire, "Client-server architecture for collaborative remote experimentation," 3rd International Conference on Information Technology and Applications, Vol 2, Proceedings, pp. 125-129, 2005.

[86] D. Gillet, A. V. N. Ngoc, and Y. Rekik, "Collaborative web-based experimentation in flexible engineering education," Ieee Transactions on Education, vol. 48, pp. 696-704, Nov 2005.

[87] C. Salzmann and D. Gillet, "Remote labs and social media: Agile aggregation and exploitation in higher engineering education," in EDUCON 2011 IEEE, 2011, pp. 307-311.

[88] Center for Science Mathematics and Engineering Education. Committee on Development of an Addendum to the National Science Education Standards on Scientific Inquiry., Inquiry and the National Science Education Standards : a guide for teaching and learning. Washington, D.C.: National Academy Press, 2000.

[89] S. W. Tuttle, D. B. Lowe, and B. Moulton, "A Survey of Issues and Approaches to Remote Laboratory Adoption by Teacher-Academics," 2011 Frontiers in Education Conference (Fie), 2011.

[90] O. Dziabenko and J. Garcia Zubia, "Secondary School Needs in Remote Experimentation and Instrumentation," iJOE, vol. 8, 2012.

[91] A. Maxwell, R. Fogarty, P. Gibbings, K. Noble, A. A. Kist, and W. Midgley, "Robot RAL-ly international - Promoting STEM in elementary school across international boundaries using remote access technology," in Remote Engineering and Virtual Instrumentation (REV), 2013 10th International Conference on, 2013, pp. 1-5.

[92] L. Atzori, A. Iera and G. Morabito, "The Internet of Things: A survey", Computer Networks, vol.54, 2010, pp.2787–2805.

[93] T. L. Koreshoff, T. Robertson, and T. W. Leong, "Internet of things: a review of literature and products," presented at the Proceedings of the 25th Australian Computer-Human Interaction Conference: Augmentation, Application, Innovation, Collaboration, Adelaide, Australia, 2013.

[94] A. Whitmore, A. Agarwal, and L. Xu, "The Internet of Things--A survey of topics and trends," Information Systems Frontiers, vol. 17, pp. 261-274, 2015.

[95] D. Singh, G. Tripathi, and A. J. Jara, "A survey of Internet-of-Things: Future vision, architecture, challenges and services," in Internet of Things (WF-IoT), 2014 IEEE World Forum on, 2014, pp. 287-292.

[96] Y. Benazzouz, et al., "Sharing user IoT devices in the cloud," in Internet of Things (WF-IoT), 2014 IEEE World Forum on, 2014, pp. 373-374.

[97] http://clout-project.eu/

[98] P. Desai, A. Sheth, and P. Anantharam, "Semantic Gateway as a Service Architecture for IoT Interoperability," in 2015 IEEE International Conference on Mobile Services, 2015, pp. 313-319.

[99] T. L. Koreshoff, T. W. Leong, and T. Robertson, "Approaching a human-centred internet of things," presented at the Proceedings of the 25th Australian Computer-Human Interaction Conference: Augmentation, Application, Innovation, Collaboration, Adelaide, Australia, 2013.

[100] D. Bandyopadhyay and J. Sen, "Internet of Things: Applications and Challenges in Technology and Standardization," Wireless Personal Communications, vol. 58, pp. 49-69, 2011.

[101] B. Guo, D. Zhang, Z. Wang, Z. Yu, and X. Zhou, "Opportunistic IoT: Exploring the harmonious interaction between human and the internet of things," Journal of Network and Computer Applications, vol. 36, pp. 1531-1539, 11// 2013.

[102] N. Dlodlo and A.C. Smith, "The Internet-of-things in remote-controlled laboratories", Annual Conference on World Wide Web Applications, Johannesburg, 14-16 September 2011, pp. 5-13.

[103] D. Guinard, V. Trifa, F. Mattern and E. Wilde, "From the Internet of Things to the Web of Things: Resource Oriented Architecture and Best Practices", Springer, New York, Dordrecht, Heidelberg, London (2011) (Chapter 5).

[104] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac, "Internet of things: Vision, applications and research challenges," Ad Hoc Networks, vol. 10, pp. 1497-1516, 9// 2012.

[105] D. Guinard, V. Trifa, and E. Wilde, "A resource oriented architecture for the Web of Things," in Internet of Things (IOT), 2010, 2010, pp. 1-8.

[106] J. Garcia-Zubia, P. Orduna, D. Lopez-de-Ipina, and G. R. Alves, “Addressing software impact in the design of remote labs,” IEEE Trans. Ind. Electron., vol. 56, no. 12, pp. 4757–4767, Dec. 2009.

[107] L. Gomes and S. Bogosyan, "Current Trends in Remote Laboratories," IEEE Trans. on Ind. Electron., vol. 56, pp. 4744-4756, Dec 2009.

[108] A. A. Kist, A. Maiti, A. D. Maxwell, et. al., “Overlay Network Architectures for Peer-to-Peer Remote Access Laboratories”, in Int. Conf. on Remote Engineering and Virtual Instrumentation (REV) 2014, 26-28 Feb. 2014, Porto, Portugal, pp.274-280.

[109] S. Manske, T. Hecking, L. Bollen, T. Göhnert, A. Ramos, and H. U. Hoppe, "A Flexible Framework for the Authoring of Reusable and Portable Learning Analytics Gadgets," presented at the Proceedings of the 2014 IEEE 14th International Conference on Advanced Learning Technologies, 2014.

[110] D. Gillet, T. de Jong, S. Sotirou, and C. Salzmann, "Personalised Learning Spaces and Federated Online Labs for STEM Education at School: Supporting Teacher Communities and Inquiry Learning," presented at the 4th IEEE Global Engineering Education Conference (EDUCON), Berlin, Germany, 2013.

[111] S. Govaerts, Y. Cao, A. Vozniuk, A. Holzer, D. Zutin, E. Ruiz, et al., "Towards an Online Lab Portal for Inquiry-Based STEM Learning at School," in Advances in Web-Based Learning – ICWL 2013. vol. 8167, J.-F. Wang and R. Lau, Eds., ed: Springer Berlin Heidelberg, 2013, pp. 244-253.

[112] Michael E. Auer, D. G. Zutin, and M. Amir, "A Toolkit to Facilitate the Development and Use of Educational Online Laboratories in Secondary Schools," Seattle, Washington.

[113] A. Maiti, A. D. Maxwell, A. A. Kist and L. Orwin, “Merging Remote Laboratories and Enquiry-based Learning for STEM Education”, International Journal of Online Engineering (iJOE), vol. 10. Iss 6, pp. 43-49, 2014.

[114] H. Park, J. Yang, J. Park, S. G. Kang, and J. K. Choi, "A Survey on Peer-to-Peer Overlay Network Schemes," in Advanced Communication Technology, 2008. ICACT 2008. 10th International Conference on, 2008, pp. 986-988.

[115] M. Amoretti, "A survey of peer-to-peer overlay schemes: effectiveness, efficiency and security," Recent Patents on Computer Science, vol. 2, pp. 195-213, 2009.

[116] H. Park, J. Yang, J. Park, S. G. Kang, and J. K. Choi, "A Survey on Peer-to-Peer Overlay Network Schemes," in Advanced Communication Technology, 2008. ICACT 2008. 10th International Conference on, 2008, pp. 986-988.

[117] G. Faraut, L. Pietrac, and E. Niel, "Formal Approach to Multimodal Control Design: Application to Mode Switching," Industrial Informatics, IEEE Transactions on, vol. 5, pp. 443-453, 2009.

[118] A. Maiti, A. D. Maxwell, A. A. Kist, and L. Orwin, "Joining the Game and the Experiment in Peer-to-Peer Remote Laboratories for STEM Education," in exp.at'15, Portugal, 2015, pp. 213-218.

[119] A. Maiti, A. A. Kist and A. D. Maxwell, "Real-Time Remote Access Laboratory with Distributed and Modular Design", IEEE Trans. on Ind. Electronics, Jun 2015, pp. 3607-3618 .

[120] P. Orduna, A. Almeida, D. Lopez-de-Ipina, and J. Garcia-Zubia, "Learning Analytics on federated remote laboratories: Tips and techniques," in IEEE EDUCON 2014, pp. 299-305.

[121] A. Maiti, et. al., "Integrating enquiry-based learning pedagogies and remote access laboratory for STEM education," in Global Engineering Education Conference (EDUCON), 2014 IEEE, 2014, pp. 706-712.

[122] D. Adami, S. Giordano, and M. Pagano, "Dynamic Network Resources Allocation in Grids through a Grid Network Resource Broker," in Grid Enabled Remote Instrumentation, F. Davoli, N. Meyer, R. Pugliese, and S. Zappatore, Eds., ed: Springer US, 2009, pp. 115-130.

[123] C. Kotsokalis, T. Ferrari, et. al., "Grid-Enabled Instrument Representation and Reservation," in IEEE Fourth Int. Conference on eScience, 2008, 2008, pp. 16-22.

[124] M. Prica, R. Pugliese, A. Del Linz, et.al., "Adapting the Instrument Element to Support a Remote Instrumentation Infrastructure," in Remote Instrumentation and Virtual Laboratories, F. Davoli, N. Meyer, R. Pugliese, and S. Zappatore, Eds., ed: Springer US, 2010, pp. 11-22.

[125] Z. Nedic and A. Nafalski, "Suitability of SCADA for development of remote laboratories," in Remote Engineering and Virtual Instrumentation 2013, pp. 1-4, 2013.

[126] R. Marques, J. Rocha, S. Rafael and J.F. Martins, "Design and implementation of reconfigurable remote laboratory, using oscilloscope/PLC network for www access", IEEE Trans. on Ind. Electron., vol. 55, pp. 2425-2432, June 2008.

[127] H. Wenshan, L. Guo-ping, and Z. Hong, "Web-Based 3-D Control Laboratory for Remote Real-Time Experimentation," IEEE Trans. on Ind. Electron., vol. 60, pp. 4673-4682, 2013.

[128] G. Patricio and L. Gomes, "Smart house monitoring and actuating system development using automatic code generation," in IEEE Int. Conference on Ind. Informatics 2009, pp. 256-261, 2009.

[129] P. Kaushik, A. K. M. Azad and K. C. Vakati, "Customizing Household Mobile Robot for Remote Laboratories", in Int. Conf. on Remote Engineering and Virtual Instrumentation 2014 (REV), pp.144-150.

[130] J. B. da Silva, W. Rochadel, R. Marcelino, et. al., "Mobile remote experimentation applied to education", IT Innovative Practices in Secondary Schools: Remote Experiments, Olga Dziabenko and Javier García-Zubía (eds.), University of Deusto, pp.281-302, 2013.

[131] A. Gardel Vicente, I. Bravo Munoz, et. al., "Remote Automation Laboratory Using a Cluster of Virtual Machines," IEEE Trans. on Ind. Electron., vol. 57, pp. 3276-3283, 2010.

[132] S. Mubeen, J. Mam-Turja, and M. Sjodin, "Extending response-time analysis of Controller Area Network (CAN) with FIFO queues for mixed messages," in Emerging Technologies & Factory Automation (ETFA), 2011 IEEE 16th Conference on, 2011, pp. 1-4.

[133] S. Corrigan, "Introduction to the Controller Area Network (CAN)", Application Report, Aug 2002 – Rev. Jul 2008.

[134] J. M. D. Pereira, O. Postolache, and P. S. Girao, "HART protocol analyser based in LabVIEW," Ieee International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, pp. 174-176, 2003.

[135] P. Brooks, "Ethernet/IP-industrial protocol," in IEEE Intl. Conf. on Emerging Technologies and Factory Automation, 2001, pp. 505-514.

[136] D. Hristu-Varsakelis, M. Egerstedt, and P.S. Krishnaprasad, “On the structural complexity of the motion description language MDLe”, In IEEE CDC 2003, 4, pp. 3360–3365 vol.4, Dec. 2003.

[137] N. Dantam and M. Stilman, "The Motion Grammar: Analysis of a Linguistic Method for Robot Control," Robotics, IEEE Transactions on, vol. 29, pp. 704-718, 2013.

[138] J. Burgner, et al., "A Telerobotic System for Transnasal Surgery," Mechatronics, IEEE/ASME Trans. on, vol. 19, pp. 996-1006, 2014.

[139] J. Yunyi, X. Ning, L. Shuang, Z. Huatao, and B. Sheng, "Multi-objective optimization for telerobotic operations via the Internet," in IEEE Intelligent Robots and Systems (IROS), 2012, pp. 5197-5202.

[140] K. B. Lee and R. D. Schneeman, “Distributed Measurement and Control Based on the IEEE 1451 Smart Transducer Interface Standards”, in IEEE Transactions on Instrumentation And Measurement, vol. 49, no. 3, June 2000, pp. 621-627.

[141] S. Bendel, T. Springer, D. Schuster, A. Schill, R. Ackermann, and M. Ameling, "A service infrastructure for the Internet of Things based on XMPP," in Pervasive Computing and Communications Workshops (PERCOM Workshops), 2013 IEEE International Conference on, 2013, pp. 385-388.

[142] M. Kirsche, R. Klauck, "Unify to bridge gaps: Bringing XMPP into the Internet of Things," Pervasive Computing and Communications Workshops, IEEE International Conference on, pp. 455-458, 2012 IEEE International Conference on Pervasive Computing and Communications Workshops, 2012.

[143] F. S. Mark and L. Phillip, "Possible Futures for Remote Laboratories," in Internet Accessible Remote Laboratories: Scalable E-Learning Tools for Engineering and Science Disciplines, K. M. A. Abul, E. A. Michael, and V. J. Harward, Eds., ed Hershey, PA, USA: IGI Global, 2012, pp. 493-510.

[144] D. G. Zutin, M. Auer, P. Ordu, and C. Kreiter, "Online lab infrastructure as a service: A new paradigm to simplify the development and deployment of online labs," in 2016 13th International Conference on Remote Engineering and Virtual Instrumentation (REV), 2016, pp. 208-214.

[145] J. Dupuis and F. Zhun, "Evolved finite state controller for hybrid system in reduced search space," in Advanced Intelligent Mechatronics, 2009, IEEE/ASME International Conference on, 2009, pp. 833-838.

[146] A. Khatkhate, A. Ray, E. Keller, S. Gupta, and S. C. Chin, "Symbolic time-series analysis for anomaly detection in mechanical systems," Mechatronics, IEEE/ASME Transactions on, vol. 11, pp. 439-447, 2006.

[147] K. Jezernik, R. Horvat, and J. Harnik, "Finite-State Machine Motion Controller: Servo Drives," Industrial Electronics Magazine, IEEE, vol. 6, pp. 13-23, 2012.

[148] M. Garcia Valls and P. Basanta Val, "Usage of DDS Data-Centric Middleware for Remote Monitoring and Control Laboratories," Industrial Informatics, IEEE Transactions on, vol. 9, pp. 567-574, 2013.

[149] Y. Tipsuwan, M. Chow, Control methodologies in networked control systems, Control Engineering Practice, Volume 11, Issue 10, October 2003, Pages 1099–1111

[150] Z. Jinhui, X. Yuanqing, and S. Peng, "Design and Stability Analysis of Networked Predictive Control Systems," Control Systems Technology, IEEE Transactions on, vol. 21, pp. 1495-1501, 2013.

[151] A. Malinowski and Y. Hao, "Comparison of Embedded System Design for Industrial Applications," Industrial Informatics, IEEE Transactions on, vol. 7, pp. 244-254, 2011.

[152] P. Gruenbaum, "Undergraduates Teach Game Programming Using Scratch," Computer, vol. 47, pp. 82-84, 2014.

[153] https://scratch.mit.edu/

[154] K. Baraka, M. Ghobril, S. Malek, et al., "Low Cost Arduino/Android-Based Energy-Efficient Home Automation System with Smart Task Scheduling," in CICSNC 2013, pp. 296-301, 2013.

[155] R. J. Costa, G. R. Alves, and M. Z. Rela, "Embedding Instruments & Modules into an IEEE1451-FPGA-based Weblab Infrastructure," Int. Journal of Online Engineering, vol. 8, 2012, pp 4-11.

[156] R. Marin, G. Leon, R. Wirz, et al., "Remote Programming of Network Robots Within the UJI Industrial Robotics Telelaboratory: FPGA Vision and SNRP Network Protocol," IEEE Trans. On Ind. Electron., vol. 56, pp. 4806-4816, 2009.

[157] S. A. Baset and H. Schulzrinne, "An analysis of the Skype peer-to-peer Internet telephony protocol," Proc. of the INFOCOM '06, 2006.

[158] Wide Area Network Emulator, 2008, Avaiable at: http://wanem.sourceforge.net/

[159] A. Maiti, A. A. Kist and A. D. Maxwell, “Building Markov Decision Process Based Models of Remote Experimental Setups for State Evaluation”, Computational Intelligence, 2015 IEEE Symposium Series on, Cape Town, 2015, pp. 389-397.

[160] M. A. Kolobov, and D. S. Weld "A theory of goal-oriented MDPs with dead ends", In Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI’12), 2012.

[161] S. Adlakha, R. Madan, S. Lall, and A. Goldsmith, "Optimal control of distributed Markov decision processes with network delays," in Decision and Control, 2007 46th IEEE Conference on, 2007, pp. 3308-3314.

[162] M. A. Wiering and E. D. de Jong, "Computing Optimal Stationary Policies for Multi-Objective Markov Decision Processes," in ADPRL 2007. IEEE International Symp. on, 2007, pp. 158-165.

[163] C. Boutilier, T. Dean, and S. Hanks, “Decision-theoretic planning: Structural assumptions and computational leverage,” J. Artif. Intell. Res., vol. 11, pp. 1–94, 1999.

[164] O. Madani, "Polynomial value iteration algorithms for detrerminstic MDPs", Proc. of the 18th UAI, pp.311–318, 2002.

[165] S. Temizer, et. al, "Collision Avoidance for Unmanned Aircraft using Markov Decision Processes", AIAA Guidance, Navigation, and Control Conference 2 - 5 Aug. 2010, Toronto, Canada. http://hdl.handle.net/1721.1/61341

[166] J. Boger, J. Hoey, P. Poupart, C. Boutilier, et.al., "A planning system based on Markov decision processes to guide people with dementia through activities of daily living," Information Technology in Biomedicine, IEEE Transactions on, vol. 10, pp. 323-333, 2006.

[167] E. G. Guimaraes, E. Cardozo, D. H. Moraes and P. R. Coelho, "Design and Implementation Issues for Modern Remote Laboratories," IEEE Transactions on Learning Technologies, vol. 4, pp. 149-161, 04/01 2011.

[168] A. Maiti, A. A. Kist and A. D. Maxwell, “Components Relationship Analysis in Distributed Remote Laboratory Apparatus with Data Clustering”, IEEE International Symposium on Industrial Electronics, Rio de Janeiro, Brazil, 3-5 Jun 2015, pp. 861-866.

[169] R. Xu and D. Wunsch, "Survey of clustering algorithms," Ieee Transactions on Neural Networks, vol. 16, pp. 645-678, May 2005.

[170] Z. L. Lin, G. Chen, X. X. Bai, H. R. Lv, W. J. Yin, and J. Dong, "Customer Clustering Using Semi-supervised Geographic Information," IEEE Intl. Conf. on Service Operation, Logistics and Informatics, pp. 465-470, 2009

[171] J. M. Pena, J. A. Lozano, and P. Larranaga, "An empirical comparison of four initialization methods for the K-Means algorithm," Pattern Recognition Letters, vol. 20, pp. 1027-1040, Oct 1999.

[172] S. Zhu, D. Wang, and T. Li, "Data clustering with size constraints," Knowledge-Based Systems, vol. 23, pp. 883-889, 12// 2010.

[173] A. Maiti, A. A. Kist and A. D. Maxwell, “Variable Interactivity with Dynamic Control Strategies in Remote Laboratory Experiments”, in International Conference on Remote Engineering and Virtual Instrumentation 2016 (REV 2016), Madrid, Spain Feb 24-26 2016.

[174] A. Maiti, A. A. Kist and A. D. Maxwell, “Latency-Adaptive Positioning of Nano Data Centers for Peer-to-Peer Communication based on Clustering”, IEEE International Conference on Communications 2015 - Workshop on Cloud Computing Systems, Networks, and Applications (CCSNA), London UK, 8-12 Jun 2015, pp. 9981-9987.

[175] A. Maiti, A. A. Kist, and A. D. Maxwell, "Estimation of round trip time in distributed real time system architectures," in Telecommunication Networks and Applications Conference (ATNAC), 2013 Australasian, 2013, pp. 57-62.

[176] S. A. Baset and H. Schulzrinne, "Reliability and relay selection in peer-to-peer communication systems," presented at the Principles, Systems and Applications of IP Telecommunications, Munich, Germany, 2010.

[177] L. Caviglione and L. Veltri, "A P2P Framework For Distributed And Cooperative Laboratories," in Distributed Cooperative Laboratories: Networking, Instrumentation, and Measurements, F. Davoli, S. Palazzo, and S. Zappatore, Eds., ed: Springer US, 2006, pp. 309-319.

[178] A. Bestavros and S. Jin, "OSMOSIS: Scalable Delivery of Real-Time Streaming Media in Ad-Hoc Overlay Networks," Distributed Computing Systems, 23rd Intl. Conf., 2003, pp. 214-219.

[179] F. Cantin, B. Gueye, M. A. Kaafar, and G. Leduc, "A Self-Organized Clustering Scheme for Overlay Networks," Self-Organizing Systems, Proceedings, vol. 5343, pp. 59-70, 2008.

[180] S. Lederer, C. Müller, and C. Timmerer, "Dynamic adaptive streaming over HTTP dataset," presented at the Proceedings of the 3rd Multimedia Systems Conference, Chapel Hill, North Carolina, 2012.

[181] A. Mittal, A. K. Moorthy, and A. C. Bovik, "Visually Lossless H.264 Compression of Natural Videos," The Computer Journal, vol. 56, pp. 617-627, May 1, 2013.

[182] A. A. Kist and A. Maxwell, "Performance and Quality of Experience of Remote Access Laboratories," presented at the IEEE TALE 2012 Hong Kong, China, 2012. doi: http://dx.doi.org/10.1109/TALE.2012.6360346

[183] V. Valancius, et.al., "Greening the internet with nano data centers," presented at the Proceedings of the 5th international conference on Emerging networking experiments and technologies, Rome, Italy, 2009, pp 37-48.

[184] I. Kurniawan, et. al., "Cost-Effective Content Delivery Networks Using Clouds and Nano Data Centers," in Ubiquitous Information Technologies and Applications. vol. 280, Y.-S. Jeong, Y.-H. Park, C.-H. Hsu, and J. J. Park, Eds., ed: Springer, 2014, pp. 417-424

[185] W. Peng and L. Yan, "The real-time computing model for a network based control system," in Control, Automation, Robotics and Vision Conference, 2004, pp. 310-315, vol. 1, 2004.

[186] V. Paulsamy and S. Chatterjee, "Network convergence and the NAT/Firewall problems," in System Sciences, 2003. Proceedings of the 36th Annual Hawaii Intl. Conf. on, 2003, pp. 1-10.

[187] Z. Liu, M. Lin, A. Wierman, S. Low, and L. L. H. Andrew, "Greening Geographical Load Balancing," Networking, IEEE/ACM Transactions on, pp. 1-1, 2014. doi: http://dx.doi.org/10.1109/TNET.2014.2308295

[188] B. Ford, P. Srisuresh, and D. Kegel, "Peer-to-peer communication across network address translators," Proceedings of the annual conf. on USENIX Annual Technical Conf., Anaheim, CA, 2005.

[189] J. Rosenberg , Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols, RFC 5245, April 2010.

[190] J. Maenpaa, V. Andersson, G. Camarillo, and A. Keranen, "Impact of Network Address Translator Traversal on Delays in Peer-to-Peer Session Initiation Protocol," in IEEE GLOBECOM 2010, pp. 1-6.

[191] Z. Duan, Z.-L. Zhang, and Y. T. Hou, "Service overlay networks: SLAs, QoS, and bandwidth provisioning," IEEE/ACM Trans. Networking., vol. 11, pp. 870-883, 2003.

[192] M. Zheng, Q. Hong, and Y. Shoubao, "A Flexible Load-Balancing Traffic Grooming Algorithm in Service Overlay Network," in Cloud Computing and Big Data , 2013 Intl. Conf. on, 2013, pp. 211-216.

[193] L. Blazevic, S. Giordano, J. L. Boudec, "Self-Organizing Wide-Area Routing", In Proceedings of SCI 2000/ISAS, 2000.

[194] R. Jain, A. Puri, and R. Sengupta, "Geographical routing using partial information for wireless ad hoc networks," Personal Communications, IEEE, vol. 8, pp. 48-57, 2001.

[195] Z. Abichar, A. E. Kamal, and J. M. Chang, "Planning of Relay Station Locations in IEEE 802.16 (WiMAX) Networks," in Wireless Communications and Networking Conf., 2010 IEEE, 2010, pp. 1-6.

[196] Z. Liu, M. Lin, A. Wierman, S. Low, and L. L. H. Andrew, "Greening Geographical Load Balancing," Networking, IEEE/ACM Transactions on, pp. 1-1, 2014.

[197] A. Beloglazov, R. Buyya, Y. C. Lee, and A. Zomaya, "A taxonomy and survey of energy-efficient data centers and cloud computing systems," Adv. Comput., vol. 82, no. 2, pp. 47–111, 2011.

[198] D. Lowe, "Impacts of Scheduling Algorithms on Resource Availability", ACSILITE 2012, 25-28 November 2012.

[199] Google Geocoding API, http://developers.google.com/maps /documentation/geocoding

[200] Regional Population Growth, Australia, Australian Bureau of Statistics, http://www.abs.gov.au/AUSSTATS/abs@.nsf/DetailsPage/3218.02011?OpenDocument.

[201] S. Laki, et.al., "A detailed path-latency model for router geolocation," in Testbeds and Research Infrastructures for the Development of Networks & Communities and Workshops. TridentCom 2009. pp. 1-6..

[202] M. L. Shooman, "Appendix B: Summary of Reliability Theory," Reliability of Computer Systems and Networks: Fault Tolerance, Analysis, and Design, pp. 411-474, 2002.

[203] A. L. Reibman and M. Veeraraghavan, "Reliability Modeling: An Overview for System Designers," Computer, vol. 24, pp. 49-57, 1991.

[204] M. L. Shooman, "Networked Systems Reliability," in Reliability of Computer Systems and Networks, John Wiley & Sons, Inc., 2002, pp. 283-330.

[205] H. G. Msuya and A. J. Mwambela, "Integration of a low cost switching mechanism into the NI ELVIS iLab Shared Architecture platform," in REV 2012 9th International Conference on, 2012, pp. 1-5.

[206] K. S. Wang, F. S. Hsu, and P. P. Liu, "Modeling the bathtub shape hazard rate function in terms of reliability," Reliability Engineering & System Safety, vol. 75, pp. 397-406, 3// 2002.

[207] I. Bazovsky, Reliability theory and practice: Courier Dover Publications, 2004.

[208] J. B. da Silva, W. Rochadel, R. Marcelino, et. al., "Mobile remote experimentation applied to education", IT Innovative Practices in Secondary Schools: Remote Experiments, Olga Dziabenko and Javier García-Zubía (eds.), University of Deusto, pp.281-302, 2013.

[209] A. Bogdanov, S. Chiu, L. U. Gokdere, and J. Vian, "Stochastic Optimal Control of a Servo Motor with a Lifetime Constraint," in Decision and Control, 2006 45th IEEE Conference on, 2006, pp. 4182-4187.

[210] Department Of Defense, "Military Handbook Reliability Prediction Of Electronic Equipment", 1991 Avail : http://www.sre.org/pubs/Mil-Hdbk-217f.Pdf.

[211] A. Maiti, A A Kist, A. Maxwell, "Time Scheduling Scheme in P2P Remote Laboratories", IEEE TALE 2014, pp 152-158.

[212] M. Popa and T. Slavici, "Embedded server with Wake on LAN function," in EUROCON 2009, IEEE, 2009, pp. 365-370.

[213] A. Zollman, "Learning for STEM Literacy: STEM Literacy for Learning," School Science and Mathematics, vol. 112, pp. 12-19, 2012.

[214] J. Ma and J. V. Nickerson, "Hands-on, simulated, and remote laboratories: A comparative literature review," ACM Comput. Surv., vol. 38, p. 7, 2006.

[215] M. Kostelníková, M. Ožvoldová, "Remote Experiments in Primary School Science Education," International Journal of Online Engineering (iJOE), vol. 9(5), pp. 39-44, 2013.

[216] D. C. Edelson , D. N. Gordin , R. D. Pea, "Addressing the Challenges of Inquiry-Based Learning through Technology and Curriculum Design", The Journal of the Learning Sciences, vol. 8, pp. 391-450, 1999.

[217] R. F. Owens, J. L. Hester, and W. H. Teale, "Where do you want to go today? Inquiry-based learning and technology integration," Reading Teacher, vol. 55, pp. 616-625, 2002.

[218] S. Barab, M. Thomas, T. Dodge, R. Carteaux, and H. Tuzun, "Making learning fun: Quest Atlantis, a game without guns," Educational Technology Research and Development, vol. 53, pp. 86-107, 2005

[219] O. A. Herrera and D. A. Fuller, "Collaborative model for remote experimentation laboratories used by non-hierarchical distributed groups of engineering students," Australasian Journal of Educational Technology, vol. 27, no. 3, pp. 428–445, 2011.

[220] T. Luis de la, "Providing Collaborative Support to Virtual and Remote Laboratories," IEEE Transactions on Learning Technologies, vol. 99, pp. 1-1, 06/04 2013.

[221] L. Divine and R. Williams, "STEM collaboration in virtual world academy," in Collaboration Technologies and Systems (CTS), 2013 International Conference on, 2013, pp. 569-575.

[222] B. Kaucic and T. Asic, "Improving introductory programming with Scratch?," in MIPRO, 2011 Proceedings of the 34th International Convention, 2011, pp. 1095-1100.

[223] I. F. de Kereki, "Scratch: Applications in Computer Science 1," in Frontiers in Education Conference, 2008. FIE 2008, pp. T3B-7-T3B-11.

[224] S. Villagrasa and J. Duran, "Gamification for learning 3D computer graphics arts," presented at the Proceedings of the First International Conference on Technological Ecosystem for Enhancing Multiculturality, Salamanca, Spain, 2013.

[225] W. Yoke Seng, M. H. M. Yatim, and T. Wee Hoe, "Use computer game to learn Object-Oriented programming in computer science courses," in IEEE EDUCON 2014, pp. 9-16.

[226] S A. Nikou and A. A. Economides, "Transition in student motivation during a scratch and an app inventor course," in Global Engineering Education Conference (EDUCON), 2014 IEEE, 2014, pp. 1042-1045.

[227] O. L. Oliveira, A. M. Monteiro, and N. Trevisan Roman, "Can natural language be utilized in the learning of programming fundamentals?," in Frontiers in Education Conference, 2013 IEEE, 2013, pp. 1851-1856.

[228] I. Rothe, "Organization of a Lego-robots contest offered to high school kids by engineering students within a project based learning environment," in IEEE EDUCON 2014, pp. 36-39.

[229] R. Pastor, R. Hernandez, S. Ros, D. Sanchez, A. Caminero, A. Robles, et al., "Online laboratories as a cloud service developed by students," in Frontiers in Education Conference, 2013 IEEE, 2013, pp. 1081-1086.

[230] J. Maloney, M. Resnick, N. Rusk, B. Silverman, and E. Eastmond, "The Scratch Programming Language and Environment," Trans. Comput. Educ., vol. 10, pp. 1-15, 2010.

[231] B. Mott and J. Lester, "Narrative-Centered Tutorial Planning for Inquiry-Based Learning Environments," in Intelligent Tutoring Systems. vol. 4053, M. Ikeda, K. Ashley, and T.-W. Chan, Eds., ed: Springer Berlin Heidelberg, 2006, pp. 675-684.

[232] A. Cardoso, et al. C. Marques, et al., "Online experimentation: Experiment@Portugal 2012," in Remote Engineering and Virtual Instrumentation (REV), 2014 11th Intl. Conf. on, 2014, pp. 303-308.

[233] A. Maiti, A. A. Kist and M. Smith, Key Aspects of Integrating Augmented Reality Tools into Peer-to-Peer Remote Laboratory User Interfaces, in International Conference on Remote Engineering and Virtual Instrumentation 2016 (REV 2016), Madrid, Spain Feb 24-26 2016.

[234] H.-K. Wu, S. W.-Y. Lee, H.-Y. Chang, and J.-C. Liang, "Current status, opportunities and challenges of augmented reality in education," Computers & Education, vol. 62, pp. 41-49, 3// 2013.

[235] J. M. Andujar, A. Mejias, and M. A. Marquez, "Augmented Reality for the Improvement of Remote Laboratories: An Augmented Remote Laboratory," Education, IEEE Trans. on, vol. 54, pp. 492-500, 2011.

[236] H. Kaufmann and D. Schmalstieg, "Mathematics and geometry education with collaborative augmented reality," Computers & Graphics, vol. 27, pp. 339-345, 6// 2003

[237] J. Camba, M. Contero, and G. Salvador-Herranz, "Desktop vs. mobile: A comparative study of augmented reality systems for engineering visualizations in education," in Frontiers in Education Conference (FIE), 2014 IEEE, 2014, pp. 1-8.

[238] D.W.F. van Krevelen and R. Poelman, "A Survey of Augmented Reality Technologies, Applications and Limitations", The International Journal of Virtual Reality, 2010, 9(2):1-20.

[239] S. Abu Shanab, S. Odeh, R. Hodrob, and M. Anabtawi, "Augmented reality internet labs versus hands-on and virtual labs: A comparative study," in Interactive Mobile and Computer Aided Learning (IMCL), 2012 International Conference on, 2012, pp. 17-21.

[240] M. B. Ibáñez, Á. Di Serio, D. Villarán, and C. Delgado Kloos, "Experimenting with electromagnetism using augmented reality: Impact on flow student experience and educational effectiveness," Computers & Education, vol. 71, pp. 1-13, 2// 2014.

[241] S. Ternier, R. Klemke, et. al., "ARLearn: Augmented Reality Meets Augmented Virtuality", Journal of Universal Computer Science, vol. 18, no. 15, 2012, 2143-2164.

[242] Z. Nedic, J. Machotka, and A. Nafalski, "Remote laboratories versus virtual and real laboratories," in Frontiers in Education, 2003. FIE 2003 33rd Annual, 2003, pp. T3E-1-T3E-6 Vol.1.

[243] V. Estivill-Castro, "Why so many clustering algorithms: a position paper," SIGKDD Explor. Newsl., vol. 4, pp. 65-75, 2002.

[244] A. D. Maxwell et.al., "An inverted remote laboratory - makers and gamers", Proceedings of the 2013 AAEE Conference, Gold Coast, Australia, 2013.

[245] L. Orwin, "Maker, Hacker, Gamer, Learner: Exploring the digital future of Remote Accessed STEM Labs", Queensland Society for Information Technology in Education conference, September, 2013. doi : http://dx.doi.org/10.1109/REV.2014.6784219

[246] D. Lowe, "MOOLs: Massive Open Online Laboratories: An analysis of scale and feasibility," in REV, 2014, pp. 1-6.

[247] S. Khuri and T. Chiu, "Heuristic algorithms for the terminal assignment problem," presented at the Proceedings of the 1997 ACM symposium on Applied computing, San Jose, California, USA, 1997.

[248] A. Kershenbaum, “Telecommunications network design algorithms” : McGraw-Hill, 1993.