UNDERSTANDING AND ENLIVENING AQM WORKINGS USING COMPUTER SIMULATION

Chong Shen shen@usq.edu.au
Zhongwei Zhang zhongwei@usq.edu.au
David Lai lai@usq.edu.au

Department of Mathematics and Computing
The University of Southern Queensland, Toowoomba, QLD

Abstract: Undeniably, computer simulation is an effective tool to help understand and analyze complex processes and systems in various areas. In recent years, many educators adopt computer simulation technology in the teaching of some topics or courses which include dynamic interactions between components. For years, many concepts of networking have been taught based on textual or other static visual materials. And many researcher have shown that illustrating dynamic scenario using static and lecture-based paradigms compromises the teaching effectiveness. This problem on computer network education prompted us to use graphical simulation. Courses related to computer communication and networking can be benefitted if computer simulation is wisely adopted.

In this paper, we describe a study in which we count on computer simulation to illustrate important and complicated algorithms of congestion control and queue management in the TCP/IP protocol suites. Comparing with current queue management techniques, Active Queue Management(AQM) is an innovative mechanism in router packet scheduling. We noticed that AQM is a promising technique and might be implemented in new generation routers. However, the concepts and internal workings of AQM schema are difficult for researchers and students to understand. Thus, we designed an interactive software to dynamically visualize the AQMs' principles and internal workings. The implementation of simulation package used Java technology due to that Java is an object-oriented programming language with extensive build-in graphical facilities and multi-threading mechanism.

In our software package, we have implemented the traditional Drop Tail(DT) and two representative AQM schemas: Random Early Detection(RED) and BLUE. It allows users to conduct their own experiments by entering different parameters for each of the algorithms, as shown in the figure below. The structure of simulation package follows the Model-View-Controller paradigm which separates the development of network models from visualization and control of the models. The animation visually describes the internal working process of the algorithms with a plot diagram, which displays the variations of the router queue size. We also compared the simulation results of the three queue management algorithms. From the animating simulation, we can easily see the difference of performance between AQM and DT. The main strength of our AQM simulator is ease of use when compared with other professional simulation tools such as NS2 or OMNet++, because users do not need solid programming capability to build the simulation from scratch. By using graphical animation, learners can directly access the internal process of the three queue management algorithms. We have received very positive feedbacks from network professionals and University lecturers for using this simulation software.

Key words: Educational computer simulation, Communication network, Active queue management and Java programming.

1 Introduction

Due to the characteristics of computer communication and networks, many concepts, algorithms and protocols are by no means easy for students to comprehend. Neither for the University educators to elucidate on the class. A traditional way of teaching and learning is to use the methods which are based on illustrative text, sketch or drawings. Educators like professors have complaint that this kind of static material is only effective for purpose of illustration of theoretical and simple principles. In most case, however, the network concepts are dynamic and complex process which make the traditional methods hard to cope with. Because of this problems, a paradigm shift has commerced from relatively static paradigm of teaching and learning to an interactive, dynamic paradigm [2]. Therefor the use of graphical simulation is adopted as assistance tools in this teaching area.

In this paper, we report a research is focused on providing an animation package which graphically simulate the working process of AQM in the router. The remaining of this paper is organized into 5 sections. In section 1, we look at some related works have been done in the similar areas for computer network. In section 2 we introduce our approach on using Java language to build the simulation package. In section 3 on the Implementation, a developed AQM animation package will be presented which helps students to understand how AQM works is described. In section 4, some analysis result generated by our graphical simulator is given. The paper is concluded by a summary and future works in section 6.

2 Related works

Computer networks are an innovative approach of communication and understanding the concepts of computer networking is essential for computer science students. Traditionally, dissemination of computer communication and networking techniques mainly rely on two complementary methods: one is transferring information from educators to learners via books and lectures which explain network concepts in a static method, or secondly, learners perform exercises while being supervised by educators [4]. Various pedagogy of teaching network protocols, TCP algorithm, queue management and other principles have been exclusively relying upon the text and sketches. As a consequence, some problems occurs when trying to explain dynamic and complex concepts of computer networking using those dull educational materials.

For years, Simulation has been proved to be a successful way of expressing and analyzing problems and recently has been adopted in various fields. An interesting method emerging in recent years is using computer simulation and animation technology. This way of disseminating has been proved to be effective by more and more educators and be adopted by many institutes. The advantage of teaching networking technology using simulation is that it can provide a interactive educational environment for knowledge transfer, requirement analysis, architecture design, performance optimization and system testing. The interaction is encouraged while user could provide different parameters to the simulation model and to see the affect through the simulation result. A statistical analysis conducted by Brian and Key [5] shows that students performed much more better in their assessment as the computer network concepts graphically simulated in the real world processes.

2.1 General network simulator

For past decades, some existing network simulator has been used widely by researchers mainly in discovering and testing new network principles. Among them, NS2 and OMNeT++ are two

examples of these tools.

NS2 is a discrete event simulator which supports the sophisticated simulation of TCP algorithms, packets routing and multi-cast protocol over network [3]. NS2 has been used in the area of network research for nearly a decade. Researchers add new protocols to the simulator by writing their protocols in Tcl/Tk script. This approach is extremely hard for students with limited programming experience in Tcl/Tk. Also, the animation of the simulation and the simulation result can only be presented after the completion of simulation process.

Similar to NS2, OMNeT++ [16] is a discrete-event simulator based on C++ and Tcl/Tk. Modules built in OMNeT++ communicate with each other by passing messages and the interface and functionality of the modules are separated which supports models reuse. The advantage of OMNeT++ is the network simulation can be executed in graphical user interface with graphics and animations, which makes simulation process visible to users. But again, like NS2, to using the simulator effectively, users have to have solid programming knowledge in C++.

2.2 Educational simulation

Beside the networking simulators used for research purpose, quite a lot of simulation tools has been developed in the context of educational purpose:

In the year of 1997, Shifroni and Ginat developed a simulation game to let the students understand the characteristics of network protocol [15]. The simulation game implemented a basic protocol of Stop and Wait in the data-link layer. They found the simulation game method are more effective than traditional lecture presentation. The understanding level and motivation of the students is also dramatically improved. But the concept of the protocol is simulated as a game rather than presenting animation to students. This is different from our simulation package since animation plays a very import role in our project.

In the paper [9], Guido and Bernd provide a animation tool called Animal for animation generation thus gives the lecturer much more flexibility in producing animations. The strengths of the Animal is its visual editing. User could design the animation from scratch to fit their requirement. Example provided in the paper shows that Animal is suitable for teaching programming algorithm with source code highlighting, but the simulation of network principle is not mentioned in their paper.

2.3 Use of Java-based simulation in education

In the past few years, the use of Java have made it possible to incorporate a broader range of material into computer science education, including software engineering, visualization, graphic user interface design, concurrency, parallelism, networking, and database connection. For instance, hundreds of Java simulations were created at the National Taiwan Normal University (NTNU) Virtual Physics Laboratory. Many physics teachers use these Java simulations in their teaching.

Holliday from the Western Carolina University recently implemented few Java Applets that illustrate some of important concept of network by using animation. The applets and accompanying materials addresses four network concepts: packet encapsulation, packet fragmentation, error control and media access. All these applets has been applied to the class of computer network. But the AQM concept is not mentioned in this paper [10].

3 Network simulation model

Our simulation package is based on the well-known Model-View-Controller(MVC) paradigm, which is widely used in the user interface programming field and has proved its worth. The basic idea behind MVC is to separate internal domain logic models from the user interface. The reason to do so is to allow the models to be reused and represented by various views without requiring any changes to the models of the application.

The MVC frame work was originally developed for use in Smalltalk-80 programming systems [12]. Indeed, since that time, the MVC design idiom has been adopted to many language and system including PHP, ASP, and Java. Particularly in Java programs, the MVC framework has been extensively used in the Swing toolkit for building interactive applications [13]. As the name indicate, there are three major components in the MVC paradigm, as shown in Figure 2.

In the Model-View-Controller architecture, the views do not have to know the existence of models. The views are read only represtation of the models state at certain given time. The views have free access to the model only via the query methods provided by the models. Methods calling from controller may change the state of the model. Consequently, this state change is reflected in associated views via notification from the models or periodical check on models. The view and controller are not always separated. For example, in Java Swing architecture, the view and controller are combined, which are called 'delegate'. But no matter how the view and controller are related, they are always separated with the models.

Based on the MVC architecture, each component or model in our simulated computer network is an autonomous entity, which could be an end-host, a router/switch, a traveling packets, or a link between two hosts. And each model contains a set of parameters which represents the component's status. Messages derived from the view are passed by the controller to the model through the methods within that model. The view of the model is generated based on model objects derived from the general classes, but the models have no knowledge on how the view going to represent themselves to the user. Same as the simulation controls applied to the models, no matter how the control panel interpret the user response, it only make change on the parameters of the models rather than the structure in the models.

The justification for doing this is to make the models reusable in the future. Because the development for each model in network simulation takes time, by applying MVC architecture to our simulation design, we actually separate models from their representation and behaviors. This can make our job easier by combining models together without considering the internal structure of a particular model.

4 Active queue management

Internet congestion control is one of popular and important topic in computer network and Active Queue Management is one of internet congestion control techniques. AQM stands for Active Queue Management, which is pro-active approach to detect data flow congestion on router or switch before the buffer overflow occurs [11]. Currently several AQM algorithms has been proposed and many continuous researches are still undergoing on this topics. The proposed technique will be applied on three algorithm: Drop Tail, Random Early Detection and BLUE.

4.1 Drop from tail - DT

Drop Tail algorithm is not a pro-active approach in detecting congestion, but it has been widely used in the existing networks. As shown in Figure 1,, when overflow happens, the routers imple-

mented with DT algorithm simply drops the incoming packet and inform the sender to reduce the sending rate.

4.2 Random early detection AQM - RED

The RED algorithm(shown in Figure 3) regards the congestion in a different way. The packets are dropped according to a probability if a predefined maximum level of capability is exceed, even the buffer is not overflowed. The goal of RED gateway is the ability to control the average queue size which acts as a indicator compared with the minimum and maximum threshold.

As shown in the RED algorithm of Figure 3, for each packet arrival at the RED gateway, the RED gateway calculates the average queue size(*avg*), using a low-pass filter with an exponential weighted moving average($EWMA$). Two thresholds(min_{th} and mam_{th}) is defined before setting up the RED gateway and compared with the current average queue size. When the average queue size is less than the minimum threshold, no packet is dropped. The dropping probability of the arriving packets is zero.

On the other extreme, when the average queue size exceeds the maximum threshold, which infers persistent and severe congestion, all arriving packets will be dropped even though the buffer may be not totally full. The dropping probability of the arriving packets is one in this situation.

When the average queue size is between minimum and maximum thresholds, the arriving packet is dropped randomly. The dropping probability is calculate as a function of three parameters: the maximum probability(max_p) that two consecutive packets will be dropped, the current average queue size and the *count*. The *count* tracks how many packets have been enqueued in buffer since the last packet was dropped.

The research conducted by Floyd [7] shows that RED do perform better than the normal DT algorithm. Although RED has some potential problem addressed by other AQM algorithms, most of them are base on RED. Therefore it's important for computer science student to have a clear understanding of RED algorithm.

4.3 Events based AQM - BLUE

Another approach called BLUE [6], which assigns packet dropping probability based on packet dropping events and the usage of the network links. The algorithm of BLUE is shown in Figure 4.

BLUE maintains a single probability p_m to drop packets when they are enqueued. The p_m is calculated every time an event occurs. When a packet loss event occurs due to buffer overflow, if the time interval between now and last update of p_m is greater than the predefined parameter $freeze_{time}$, the p_m is increased. Therefor, if the queue is continually dropping packets due to buffer overflow, the p_m keeps increasing. Conversely, if the buffer becomes empty, BLUE decrease the p_m, which means less arriving packets will be dropped in the buffer.

The advantage of using BLUE is the buffer size can be somehow self-adjusted base on the vary of dropping probabilities.

Network congestion control and queue management are relatively complicate and dynamic concepts in computer network. Without any animated tools, it's hard to throughly understand the idea of AQM. The computer simulation a wonderful tool to deal with this problem, but disscussed in Section 2, the current tools are either hard to use or not properly to applied in the simulation of AQM.

5 Structure of AQM simulator

In this section, we explain how our simulation system is structured by using the MVC architecture and Java Technology to visualize the internal workings of three AQM algorithms. We also propose a technique helpful in understanding these complex networking protocols or principles. We start with a relatively simple animation of active queue management.

All the three algorithms of interests has been implemented in our simulator. The objective is that user could test the different algorithms while watching the animations. The simulation package is designed based on the Model-View-Controller paradigm. Thus, the structure of the simulator is consisted of three main components that representing the Model, View and Controller respectively. The structure of the simulation system is shown in Figure 8.

The internal structure of the simulation package mainly consisted of four components: User Interface, Simulation Model, Animation Generator and Simulation Conroller. Visualization of AQM working includes two general aspects. First is the animation of process on the network which graphically simulate the whole picture of network concept. Second is the graphically representation of simulation result. The animation generator and result presentation of our system structure serves the role of visualization. Each components are explained as below:

- User Interface: The main inter-media between the user and the system. User manipulate the simulation model and input simulation parameters through the system interface. Animation and result representation are also displayed to the user by the interface.
- Simulation model: simulation entities and algorithm are build as simulation models which participate any given simulation scenarios.
- Animation generator: The simulation processes derived from interaction of simulation model is graphically displayed on the animation panel. The animation is fully controllable by the user via system interface. Simulation result is display in comparable diagrams, for example, a plot digram shows the number of packets dropped in a router based on different buffer size during certain period.
- Simulation Controller: Any input or user interaction to the system is received by the controller and direct to corresponding models. The controller and animation generator are combine together to perform simulation tasks.

6 Implementation techniques

Java technology is getting more attention from academics and commercial community. Java language is not only good in teaching object-oriented technique but also in Internet programming [1]. Java programming language is chosen to build the simulation package because its platform independence, graphical capability, class re-usability and Internet capabilities. In the future, the simulation package may be implemented in Java applet which could be accessed through the website.

6.1 Object-oriented design

The develop process of the simulation system is highly dynamic, different requirements keep adding to the system during the development. For modeling the software process, especially an

object-oriented development approach, we choose to use Unified Modeling Language(UML) [8], the object-oriented modeling language for software modeling. The benefits are:

- ease the communication among number of people because UML is a widely recognized modeling language.
- UML provide a rice model notations such as class diagram, activity diagram which facilitate both structure and dynamic modelling process.

Because of the limit of page space, it's impossible to show all the attributes and methods implemented in each class. Figure 5 shows the association of the classes defined in our simulation package without any specified attributes and methods.

- The visualization of the simulation is mainly handled by the MainFrame class. As mentioned earlier, the router simulates Drop Tail, RED and BLUE algorithm in its buffer. So in our system, we implement three AQM algorithms by creating three inherited classes in the AnimationBox class.
- The AnimationBox class provide a workspace for all the instance from other classes(like Host, Packets, Router and Connector) to work together by using Threads, which will be explained in detail later. As shown in Figure 8, the AnimationBox panel contains several sending and receiving host with a inter-connected router, packets are send from hosts in one side to hosts in another side. The number of the hosts is adjustable.
- The router buffer shows the instance queue size of the router. In a RED simulation, the animation also contains rectangle bar which indicate the current average queue size for the buffer.
- The PlotFrame class contains the plotPanel that generate the plot diagram which shows the variation of the queue size at run-time. We implement the plot diagram from scratch rather than using read-to-use plot generator, as most of them are hard to use due to the complexity or not suitable to our system.

6.2 GUI and animation features

The interface of the system is built on the Java Swing package. User provides simulation parameters through the interface. The parameters include topology of the network and other settings for different AQM algorithms. Alternatively, the parameters setting can also load from a predefined script. The simulation model carries on the simulation based on the parameters. The basic class used in the simulation model are host, connector and packet. The animation of the simulation is presented through animation generator which displayed on user interface. The plot generator take care of interpreting the statistic simulation result into plot diagrams and present to user through the interface. The animation and result presentation is developed on Java2D API and AWT package [17].

6.3 Parallel processing and concurrency

The computer networking are consist of dynamic and stochastic processes, and the factors that influence the the system keep changing during the system running. When our simulation system is operated in real-time, factors influence the processes are dynamic and random, and the concurrent

proceeding processes influence each other as time progresses. Thus, concurrency and parallel processing must be considered in our simulation system implementation. Figure 9 shows the threads created during the system running.

The main thread during the run time is the AnimationBox thread which paint the whole structure of the animation panel. The packets sent from each host operate as a individual thread. The router thread receiving the packets from different connector and put them into buffer for processing. The DT or RED buffer thread handles the packets based on its algorithm. At the same time, the plot digram calls the plot panel thread to paint the plot diagram.

7 Experiment results

The main frame of the simulation package has been developed with the queue management algorithms: Drop Tail, RED, and BLUE. The main structure for these three algorithms extends the same network frame which could be modified at high level , and the only differences of these simulations is in the simulated buffer. For each arriving packet from the network links, the buffers work differently base on current algorithm applied in the buffer, but the change of the algorithm in the buffer does not affect the working of other components. For now, the simulated buffer simply drop the packets if the criteria for dropping packets is satisfied. We now look at more detailed working results of the three algorithms.

7.1 DT AQM algorithm

Drop Tail is first implemented algorithm in our simulation package because it's simplicity. We marked color of the packets into red as the packets is dropped when the buffer is full(in Figure 10). The instant buffer size is recorded displayed in the plot diagram which is shown in Figure 11. For the time being, we have not consider the sending rate of the hosts, so when the router buffer is full, the sending host of that dropped packets is not notified and keep sending packets as normal. Therefor, in the plot diagram, the instance buffer size keep as maximum all the time after the initial period.

7.2 RED AQM algorithm

To illustrate the RED algorithm more clearly, beside the buffer used in DT simulation, we add one more indicator to display the average queue size, as shown in Figure 12. During the simulation, the forced dropped packets(when average buffer size exceed maximum threshold) are marked as red, and the random dropped packets(when average buffer size between minimum and maximum threshold) are marked as yellow.

Mentioned in paper [14], the wq is a very sensitive parameter which greatly influences the perform of RED algorithm. In the plot diagram of Figure 12, we tested the simulation with wq set to 0.2 and 0.8. With the $wq = 0.2$, when the instance buffer increase and decrease, the average queue size is responded relatively slowly, which makes the instance buffer size reaches higher. In another hand, with the $wq = 0.8$, the average queue size becomes much more sensitive to the variation of instance queue size. That's why both of them remains steady around the maximum threshold after a very short time running.

Compare with DT, with proper parameter settings, the RED buffer will never overflowed and the instance buffer size is maintain properly with high utilization.

7.3 BLUE AQM algorithm

The animation of BLUE simulation is similar with DT(Figure 13), except the packet is assigned a dropping probability base on the utilization of the buffer rather than average queue size as in RED.

Same as DT, the plot diagram of BLUE show the variation of the instance buffer size against the simulation time, as shown in Figure 14. Compare this figure with Figure 12, quite different from RED, the pattern of the instant queue size in BLUE are more unpredictable. It also shows a fluctuate pattern but the amplitude is much more wider than RED. This indicate a better buffer utilization because the buffer size is not increased and decreased dramatically, the variation of BLUE buffer is relatively flatter than RED.

The simulation package shows what we expected from the results presented. The main strengths of our AQM simulator is the ease of use, to manipulate with the simulator, user without solid network knowledge can easily understand basic principles of AQM. We received very positive feedback from lecturers during the system testing. The simulation package also shown the expected effect on improving the students motivations. Students like to explorer more about background AQM algorithms by modify settings to compare different scenarios. This makes students becomes active learner in the topic of AQM.

8 Conclusion

In this paper, we describe an approach of simulating computer networks concepts and animating many algorithms and protocols. The MVC control-view model was adopted which separate the display and control. Based on the MVC model, network simulation has modelled the networks as a group of autonomous entities which can pass the message and execute its own assigned tasks.

The resultant animation system which mainly visualize three AQM algorithms has been developed using the proposed model and Java technology. In addition to the lively display of the internal workings of DT, RED, BLUE, a set of tools are developed and implemented. Particularly, an animation of the router where packets queue up and dispatch and their dropping probability changing based on the buffers length has significantly reduced the difficulty and complexity of understanding these algorithms and protocols.

The research can be extended to visualize the TCP algorithm and new network model such as DiffServ. Future works include the further exploration into wireless networks and the correspond TCP, AQM and other algorithms.

References

[1] Joseph Bergin, Thomas L. Naps, Constance G. Bland, Stephen J. Hartley, Mark A. Holliday, Pamela B. Lawhead, John Lewis, Myles F. McNally, Christopher H. Nevison, Cheng Ng, and George J. Pothering, *Java resources for computer science instruction*, SIGCUE Outlook **26** (1998), no. 4, 14–34.

[2] Christopher M. Boroni, Frances W. Goosey, Michael T. Grinder, and Rockford J. Ross, *A paradigm shift! the internet, the web, browsers, java and the future of computer science education*, SIGCSE '98: Proceedings of the twenty-ninth SIGCSE technical symposium on Computer science education (New York, NY, USA), ACM Press, 1998, pp. 145–152.

[3] Lee Breslau, Deborah Estrin, Kevin Fall, Sally Floyd, John Heidemann, Ahmed Helmy, Polly Huang, Steven McCanne, Kannan Varadhan, Ya Xu, and Haobo Yu, *Advances in network simulation*, IEEE Computer **33** (2000), no. 5, 59–67, Expanded version available as USC TR 99-702b at `http://www.isi.edu/~johnh/PAPERS/Bajaj99a.html`.

[4] Cora Burger, Kurt Rothermel, and Rolf Mecklenburg, *Interactive protocol simulation applets for distance education*, Lecture Notes in Computer Science **1483** (1998).

[5] Brian H. Cameron and Kay Wijekumar, *The effectiveness of simulation in a hybrid and on-line networking course*, SIGCSE '03: Proceedings of the 34th SIGCSE technical symposium on Computer science education (New York, NY, USA), ACM Press, 2003, pp. 117–119.

[6] Wu-Chang Feng, Dilip Kandlur, Debanjan Saha, and Kang G. Shin, *Blue: an alternative approach to active queue management*, NOSSDAV '01: Proceedings of the 11th international workshop on Network and operating systems support for digital audio and video (New York, NY, USA), ACM Press, 2001, pp. 41–50.

[7] Sally Floyd and Van Jacobson, *Random early detection gateways for congestion avoidance*, IEEE/ACM Trans. Netw. **1** (1993), no. 4, 397–413.

[8] I. Jacobson G. Booth, J. Rumbaugh, *Unified modeling language user guide*, Addison Wesley, 1998.

[9] Guido and Bernd Freisleben, *Experiences in using animations in introductory computer science lectures*, SIGCSE '00: Proceedings of the thirty-first SIGCSE technical symposium on Computer science education (New York, NY, USA), ACM Press, 2000, pp. 134–138.

[10] Mark A. Holliday, *Animation of computer networking concepts*, J. Educ. Resour. Comput. **3** (2003), no. 2, 1–26.

[11] Gianluca Iannaccone, Martin May, and Christophe Diot, *Aggregate traffic performance with active queue management and drop from tail*, SIGCOMM Comput. Commun. Rev. **31** (2001), no. 3, 4–13.

[12] G. Krasner and S. Pope, *A description of the model-view-controller user interface paradigm in the smalltalk-80 system*, Journal of Object Oriented Programming **1** (1988), no. 3, 26–49.

[13] Scot F. Morse and Charles L. Anderson, *Introducing application design and software engineering principles in introductory cs courses: model-view-controller java application framework*, J. Comput. Small Coll. **20** (2004), no. 2, 190–201.

[14] Kevin Fall Sally Floyd, *Ns simulator test for random early detection queue management*, IEEE/ACM Transactions on Networking (1997).

[15] Eyal Shifroni and David Ginat, *Simulation game for teaching communications protocols*, SIGCSE '97: Proceedings of the twenty-eighth SIGCSE technical symposium on Computer science education (New York, NY, USA), ACM Press, 1997, pp. 184–188.

[16] A. Varga, *Using the omnet++ discrete event simulation system in education*, IEEE Computer **42** (1999), no. 4.

[17] Linda Wilkens, *A multi-api course in computer graphics*, CCSC '01: Proceedings of the sixth annual CCSC northeastern conference on The journal of computing in small colleges (, USA), Consortium for Computing Sciences in Colleges, 2001, pp. 66–73.

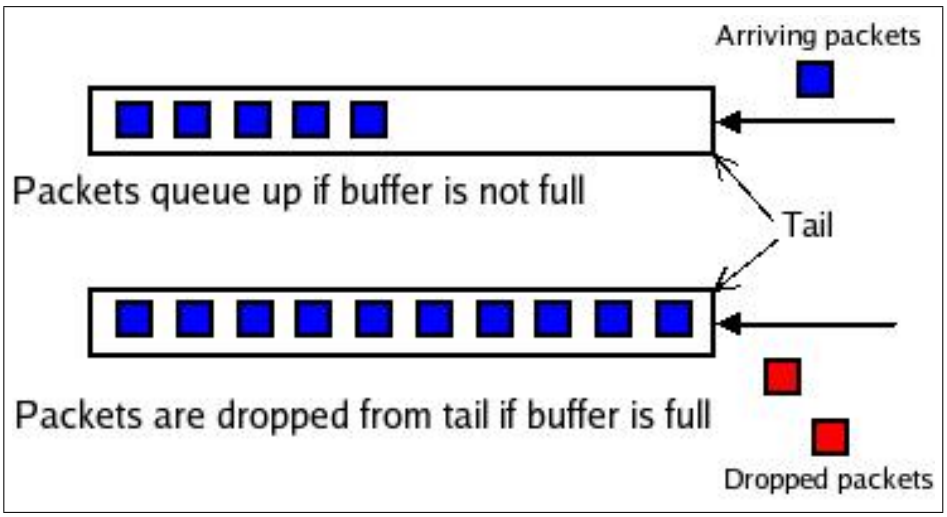


Figure 1: Drop from tail

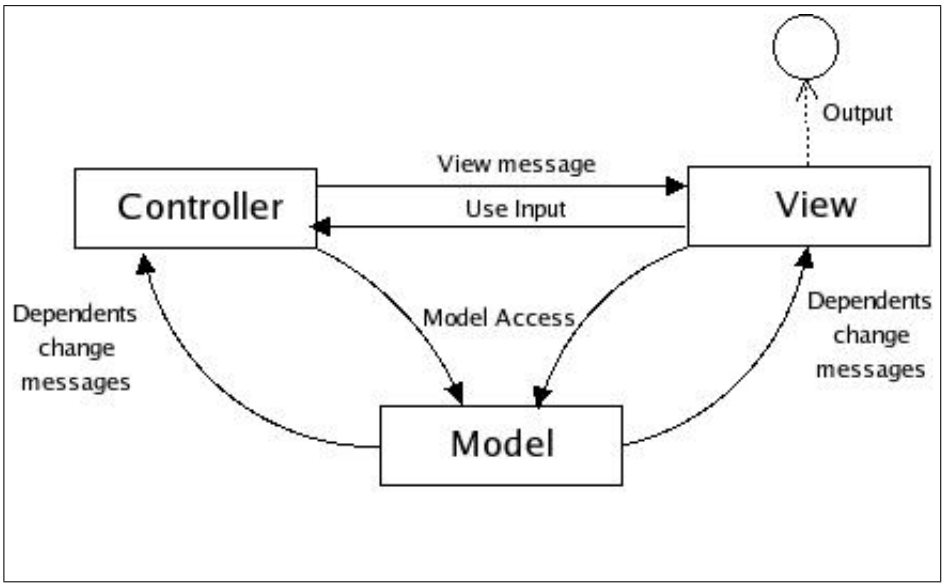


Figure 2: Model-View-Controller and message flows

Initialization
    avg = 0
    count = 1
end
for each packet arrival
    calculate the new average queue size 'avg'
        if the queue is nonempty
            $avg = (1-wq)*avg + wq*q$
        else
            $m = f(time - q_{time})$
            $avg = (1-wq)^{m}*avg$
        end
        if $min_{th} \leq avg < max_{th}$
            increment count
            calculate probability p_a
                $(p_b = max_p*(avg - min_{th})/(max_{th} - min_{th}))$
                $p_a = p_b/(1 - count*p_b)$
            end
            with probability p_a
                mark the arriving packet
                count = 0
            end
        end
        else if $max_{th} \leq avg$
            mark the arriving packet
            count = 0
        end
        else
            count = -1
        end
    end
end

when queue becomes empty
    $q_{time} = time$
end

Figure 3: RED algorithm

Upon packet loss(or $Q_{len} > L$) event:

if ($now - last_{update} > freeze_{time}$) then

$p_m = p_m + d_1$

$last_{update} = now$

Upon link idle event:

if ($now - last_{update} > freeze_{time}$) then

$p_m = p_m - d2$

$last_{update} = now$

Figure 4: BLUE algorithm

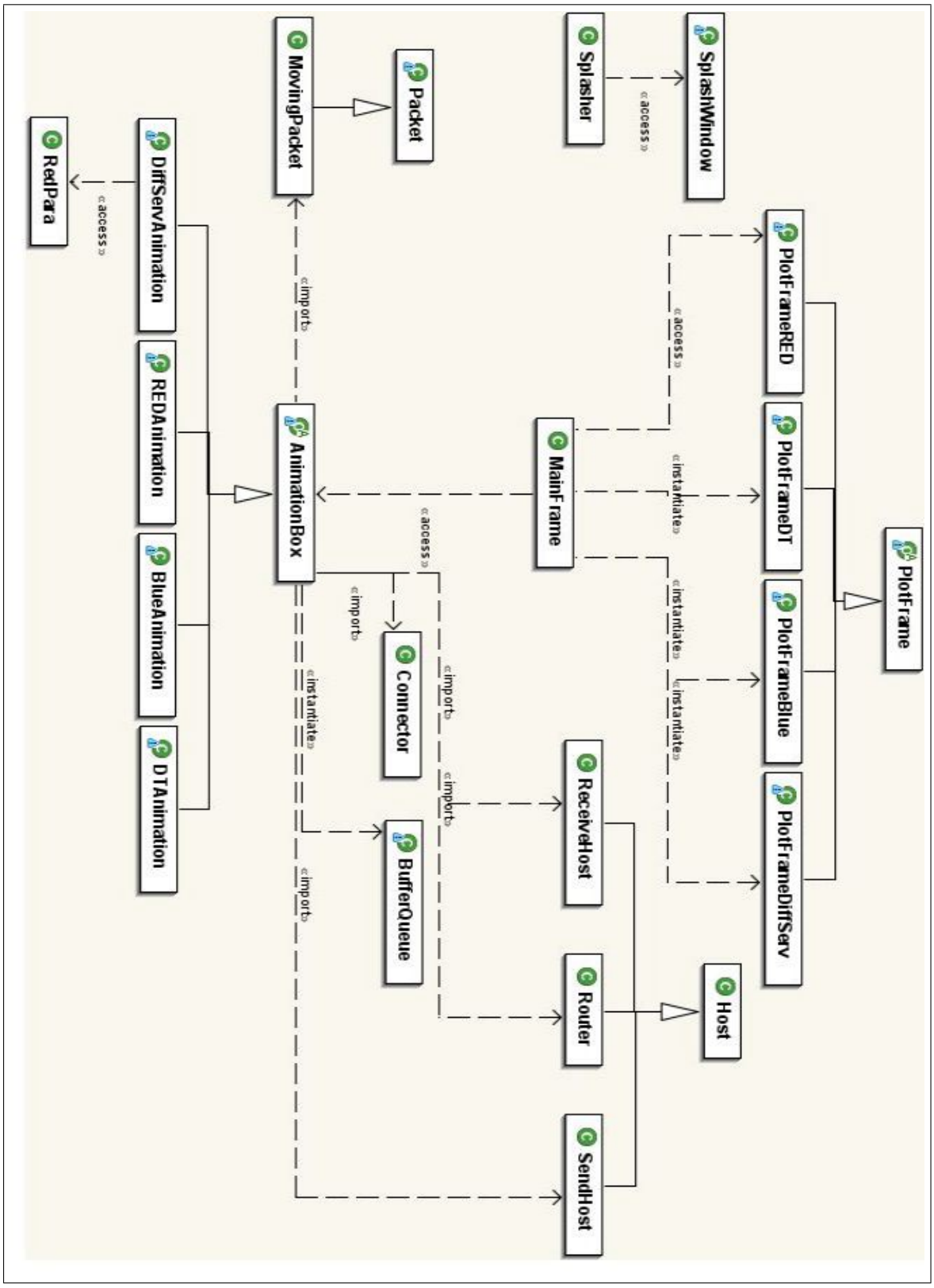


Figure 5: Class diagram of the simulation system

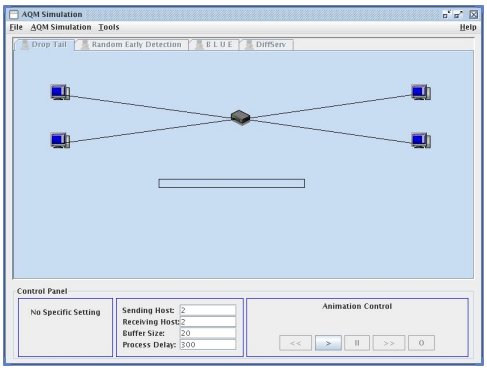


Figure 6: The animation panel

Figure 7: Splash screen

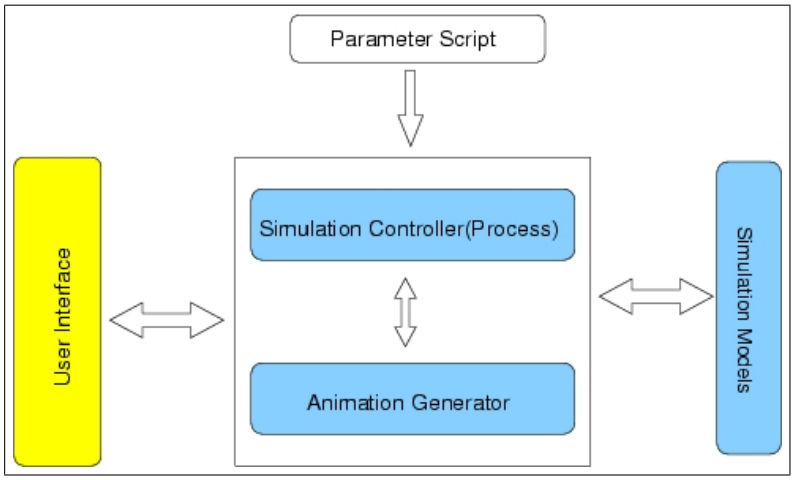


Figure 8: Basic structure of the simulator

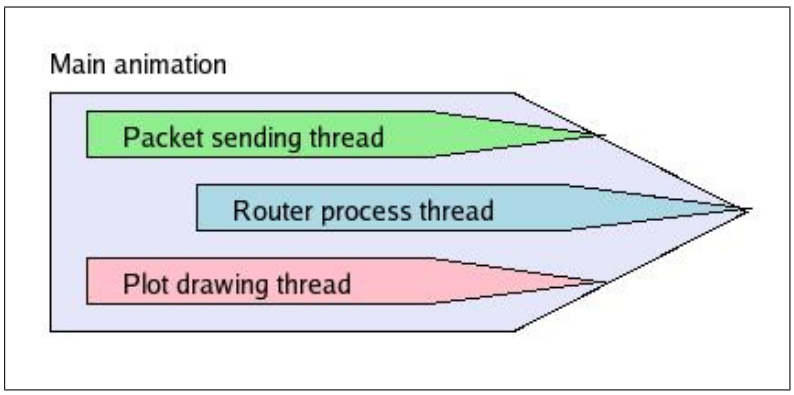


Figure 9: Different running threads in the simulation

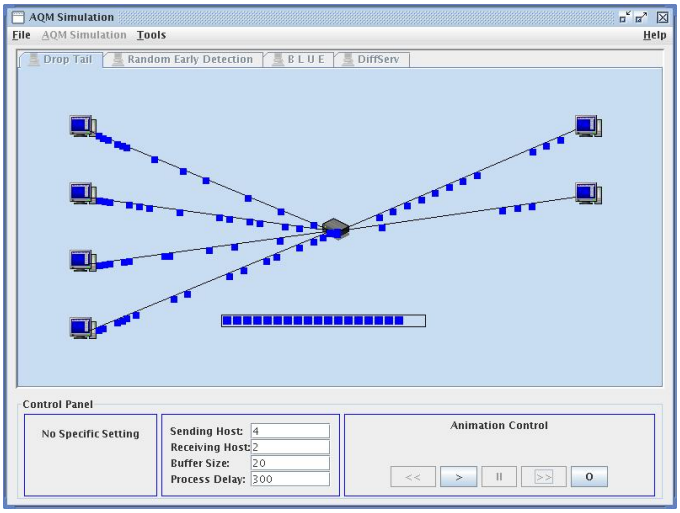


Figure 10: DT simulation

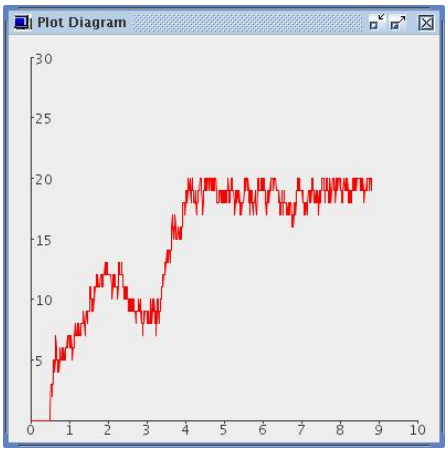
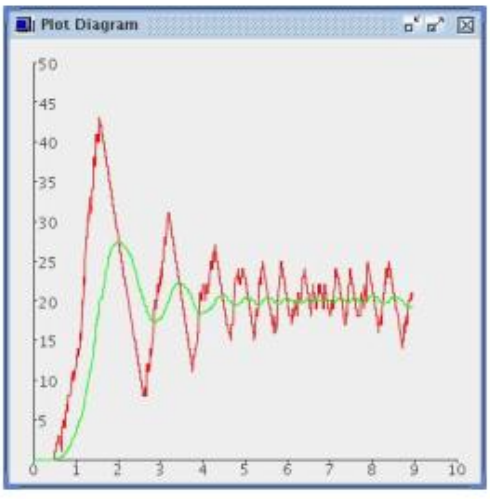
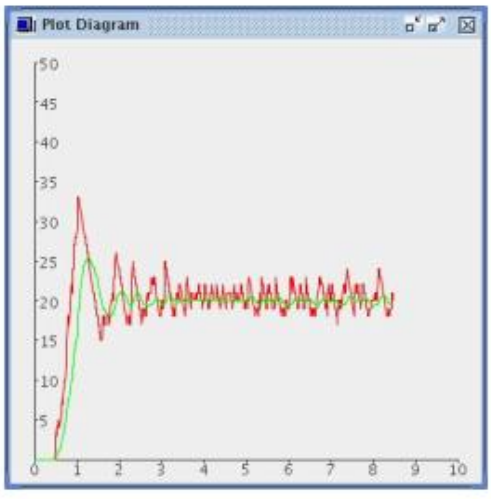


Figure 11: The plot diagram of the instant buffer size in DT Simulation



wq = 0.2



wq = 0.8

Figure 12: RED simulation

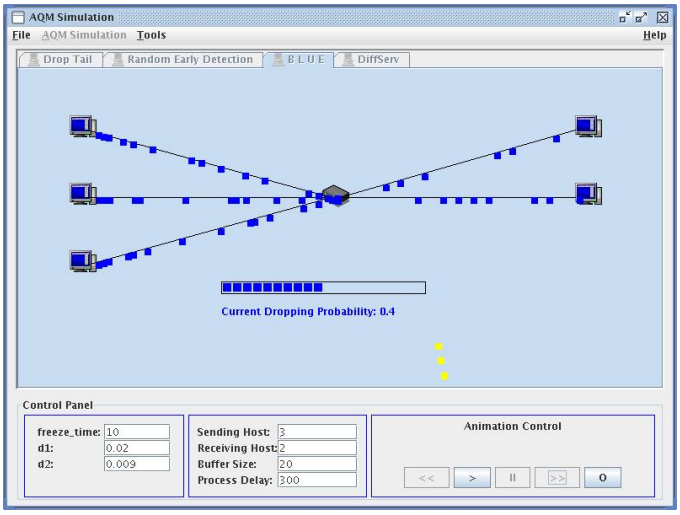


Figure 13: BLUE simulation

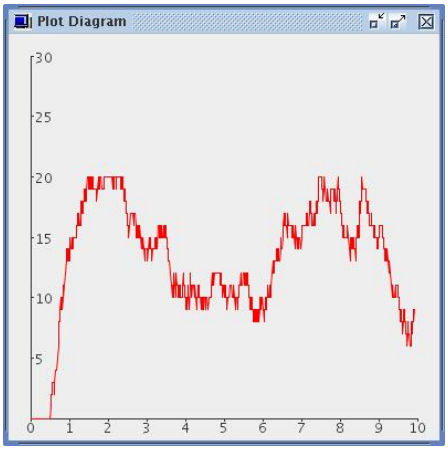


Figure 14: The plot diagram of the instant buffer size in DT Simulation