Netml – A Language and Website for Collaborative Work on Networks and their Algorithms

Ron Addie, Stephen Braithwaite and Abdulla Zareer
Department of Mathematics and Computing
University of Southern Queensland
Toowoomba, QLD, Australia

August 19, 2006

Abstract

This paper describes a language, Netml, for describing networks, a web site which provides a collaborative working environment for use and development of network data files and software for networks. The language, Netml, is based on XML. An XML schema for this language has been developed. Algorithms for availability analysis, loss analysis, routing, traffic flow analysis, and link dimensioning have also been made available on the Netml web site and are described in outline in this paper.

A repository of networks has also being set up and is described here. In addition to the algorithms for network analysis and design, scripts for converting `dia` files to `Netml`, and for generating postscript from `Netml` have also been developed. The software and the web site are particularly directed to educational purposes and enable students of network analysis and design to specify networks and apply an increasing range of algorithms to these networks in the course of their studies.

1 Introduction

Communication networks are of increasing importance to every aspect of modern life. The design, planning, analysis and management of networks is therefore of increasing interest. Fortunately, many modern communication networks are quite robust, readily adapt to a great range of conditions, and therefore do not require detailed decision making to keep costs low while delivering satisfactory performance. However, it would be premature to suggest that our networks are so good at adapting to current conditions that we do not need to understand how they perform and how to manage their performance at all.

We therefore need to continue to develop, refine, and apply mathematical techniques for analysis and design of our communication networks, and to teach these techniques to students.

This paper is not about developing new techniques for design and analysis of networks but rather about a framework within such work can be conducted efficiently and with effective collaboration between interested parties.

There are many network planning, designing and analysis tools [1, 2, 3, 4] available in the market, most of which are commercial tools. Due to the complexity of network planning process, a single network planning tool does not have all the required features or algorithms a planner needs to complete their work. Since these planning tools use different formats for entering and storing network models, using multiple tools to perform planning tasks is a tedious undertaking. For these reasons having a single language to describe networks will be a great help. In this context it is important that such a language is independent of the planning tools used and that it is extensible.

We have developed an XML based language for describing networks - Network Modelling Language (Netml). Having such a language defined in XML will make it operating system and language independent making the transmission and sharing of network models easier via the Internet and across different platforms.

The purpose of the web site is to make network planning tools available that work with Netml, to provide a central registry for example networks and to link to other related resources. The site is implemented using Java Server Page technology. This site can be found at `http://cs.sci.usq.edu.au/netml2`. In this paper we first describe the Netml language that we have developed using XML schema language, then we describe the network analysis tools which have been implemented in the *Java* and *XSLT* languages and made available on the web site. Next we present the repository of networks. Finally, we present concluding remarks and future work.

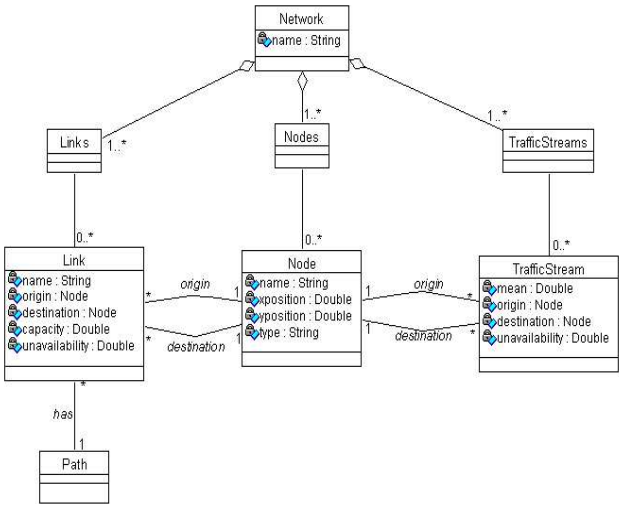

Figure 1: UML class diagram for network

2 Netml – An XML Language for Networks

A network description is a static representation of the network. The principal elements in a network are nodes and links. The nodes carry out routing, switching and/or processing on messages or signals. Traffic streams are another important element in the description of networks. Traffic streams are used to describe in general the statistics of the messages and/or signals which the network will carry.

Figure 1 is a UML class diagram showing how its elements are related and how together with other elements, they form a network object. A network is made of a collection of nodes, links and traffic streams. Nodes are identified by a name and have other attributes as shown in the diagram. Nodes are interconnected by links. A node does not necessarily correspond only to a router or an end-system, but may also represent an autonomous system. A node can have many links originating or terminating at itself. A link is also identified by a name and has one origin and one destination node. A link can be a point-to-point or multipoint link. The traffic stream object identifies the traffic flow information from an origin node to a destination node. A path is composed of a list of links. This is an object which is expected to be useful for describing routing and in algorithms for network design, but up to this point we have not made use of paths.

2.1 Netml Schema

The syntax of the Netml language has been specified by means of the XML schema language [5]. To develop the Netml language an XML document was derived from the UML class diagram shown in Figure 1. Figure 2 shows the definition for Netml root element. Inside the root element definitions for all other objects of the network namely *nodes*, *links* and *trafficStreams* are provided. Also the definition for *displaySetting* is defined. The Netml schema document also contain namespace related information such as default namespace, target namespace, etc. The grammar for the complete Netml schema is availabe on the website.

2.2 Netml Example

The example network shown in Figure 3 was described based on Netml. This picture was generated by using the tool for generating postscript from netml which is available on the Netml website. An extract from the netml file for this network is shown in Figure 2.2.

```
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://cs.sci.usq.edu.au/netml"
  xmlns:netml=
  "http://cs.sci.usq.edu.au/netml/netml">

<!-- Description of Network -->
<element name="network">
  <complexType>
    <all>
    <element name="nodes" minOccurs="1"
    maxOccurs="1">
      <complexType>
        <sequence>
          <element name="node"
            type="netml:nodeType"
            minOccurs="0"
            maxOccurs="unbounded" />
        </sequence>
      </complexType>
    </element>
    <element name="links" minOccurs="1"
    maxOccurs="1">
      <complexType>
        <sequence>
          <element name="link"
          type="netml:linkType"
          minOccurs="0"
          maxOccurs="unbounded" />
        </sequence>
      </complexType>
    </element>
    <element name="trafficStreams"
    minOccurs="1" maxOccurs="1">
      <complexType>
        <sequence>
          <element name="traffic"
          type="netml:trafficType"
          minOccurs="0"
          maxOccurs="unbounded" />
        </sequence>
      </complexType>
    </element>
    <element name="displaySetting"
    type="netml:displaySettingType"
      minOccurs="0" maxOccurs="1" />
  </all>
  </complexType>

</element>
```

Figure 2: Netml schema root element

3 Analysis and Design Tools

3.1 Availability Tool

In this section we explain a network planning algorithm we have implemented which makes use of the Netml markup language. We have chosen and implemented the availability calculation algorithm that was presented in [7]. The implementation reads a network described using the Netml language and calculates the total probabilities of all states considered and also the unavailability of each Origin-Destination($O-D$) pair of the network.

Based on the method explained in [7] the $O-D$ availability can be calculated to a desired degree of accuracy

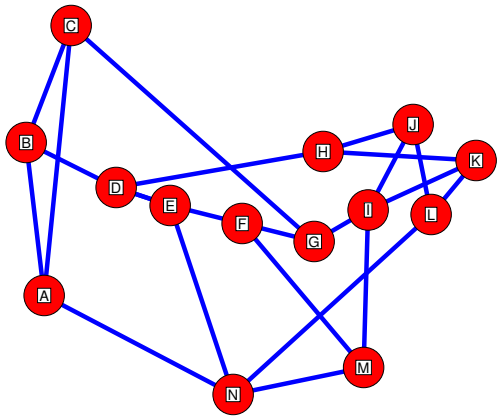

Figure 3: Example Network (source : [6])

```
<netml:network xsi:schemaLocation="...">
  <nodes>
    <node>
      <name>A</name>
        <xposition>70</xposition>
        <yposition>150</yposition>
      </node>
      <node>
        <name>B</name>
        <xposition>50</xposition>
        <yposition>320</yposition>
      </node>
.....
    <links>
      <link>
        <name>link1</name>
        <origin>A</origin>
        <destination>B</destination>
        <capacity>200</capacity>
      <unavailability>0.01</unavailability>
    </link>
.....
```

by enumerating the states of the network in “alphabetical order”, and terminating when the sum of probabilities is close to 1 or when the number of failures have reached the maximum number of failures specified.

The availability algorithm was designed and implemented in the `Java` programming language using Object Oriented methods. Figure 4 shows a UML class diagram

of the objects and their relationships for the system we developed to implement the algorithm. The implemented system mainly performs four basic process. That is:

1. It reads in a network specified in Netml, parses it creating a Document Object Model(DOM).
2. Converts the DOM document to an internal data structure to be used by the system.
3. Calculates the availability of network using the internal data structure which is optimised for this purpose.
4. Store the end-to-end availabilities in the DOM object of the XML document.
5. Show the results, and produce an xml file from the DOM object which now includes the results.

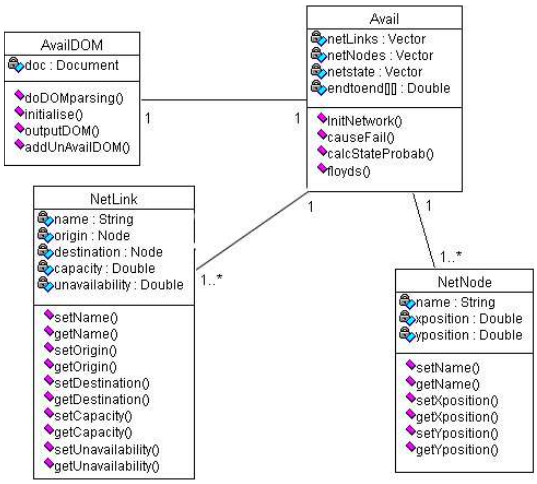

Figure 4: UML class diagram of the system

As shown in the Figure 4 there are four classes implemented in this system *Avail*, *AvailDOM*, *NetNode* and *NetLink*. The *Avail* class is the main class responsible for creating other objects and implements various methods for the calculation of unavailability. The *causeFail()* is a recursive function which implements the core algorithm. This function is responsible for enumerating all the possible states of the network and accumulating the probabilities. During the calculation of probabilities to determine whether there is a path or not between each $O - D$ pair, the Floyd [8] algorithm is implemented. Since we need to consider every O-D pair, this is more efficient than using Dijkstra's algorithm.

The *AvailDOM* class implements a DOM parser using the Xerces2 Java XML parser [9] based on the JAXP specification [10]. This class is responsible for creating a DOM tree of the XML network, converts the nodes in DOM object to the internal data structures used in the system. This class has methods to update the DOM object and also to write the modified DOM tree to a file in XML format. The internal data structures *NetNode* and *NetLink* are used to improve the efficiency of the system.

3.2 Traffic Analysis and Dimensioning

The website includes traffic analysis tools which carry out the following functions:

(a) routing tables are calculated, in accordance with the traditional philosophy of using the shortest path; these routing tables are stored in the nodes of the network;

(b) the mean and variance of the aggregate traffic on each link is calculated from the mean and variance of the traffic streams;

(c) the loss which would be experienced by traffic flows, on each' link, are calculated and stored as a parameter of each link.

(d) The capacity which would be required on each link in order that the network should be able to carry all the traffic even when one link hsa failed is calculated and stored in the capacity element of the links.

4 XSLT Scripts

4.1 Making Pictures

A tool developed using the XSLT transformation language[11] translates a description of a network in Netml into a picture of a network, in postscript. This tool allow us to display or emphasise certain features of a network.

4.1.1 Display Settings

The way that nodes, links and traffic are displayed is controlled by display attributes within the XML. The colour, width, labels of lines, shape of nodes, whether hidden and the labels may be controlled by the display settings. These display settings occur in two places. Display settings may be set globally for the entire network, and also locally for any individual node, link or traffic object.

Netml uses a flexible mechanism to decide whether to draw using a local setting or to draw using a global setting. Global and local display settings all have a priority setting and the display tool always uses the setting with highest priority.

This mechanism is more flexible than the mechanism of always having local settings override global ones. We can, for example, have a network with links of varying degrees of importance, and control whether the more minor links are displayed varying only the *priority* attribute of the global *hidden* setting.

Figure 5 shows an example settings of the *displaySetting* object which can be defined in the XML network. Only links that have a higher priority for the *hidden* setting than the one in the global display object are displayed. Figure 6 shows the output network created by the XSLT system developed based on the display setting shown in Figure 5.

```
<displaySetting>
  <nodeSetting>
    <shape priority="5">box</shape>
    <size priority="5">40</size>
    <colour priority="5">red</colour>
  </nodeSetting>
  <linkSetting>
    <hidden priority="5">yes</hidden>
    <label priority="2">unavailability</label>
    <width priority="1">5</width>
    <colour priority="2">blue</colour>
  </linkSetting>
  <trafficSetting>
    <hidden priority="3">no</hidden>
    <label priority="3"></label>
    <width priority="2">8</width>
    <colour priority="2">green</colour>
  </trafficSetting>
</displaySetting>
```

Figure 5: Display Setting

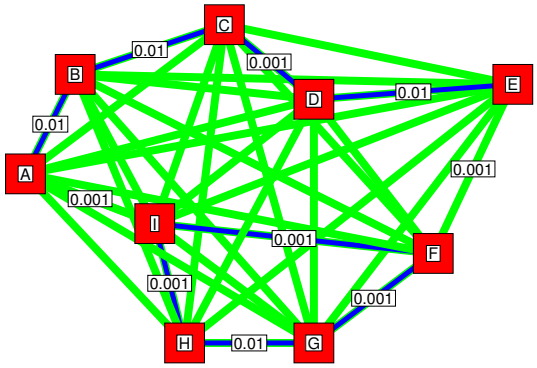

Figure 6: Network based on settings in display setting

4.1.2 Design of the Display Tool

To develop the system we have first created the postscript which is needed to draw a network. The postscript generated by the display tool includes a postscript header in which there are quite a number of definitions. These definitions effectively extend the postscript language, adding functions which will draw a node, link, traffic, box, circle and write a label. Once the postscript has been designed several XSLT transformation rules were created to transform XML networks described in Netml to postscript. In these rules we have used XPath expressions [12] to gain access to various parameters in the network.

The root template defines the postscript header including the definitions of all the functions used in the subsequent postscript. Also it has code to invoke the templates for nodes, links and trafficstreams and then writing labels and drawing the nodes. The node template, link template and traffic template define details to process each node, link and traffic of the network.

4.2 Dia to Netml translation

An *XSLT* script which converts a network created using the diagram editing tool *dia* has been developed. This makes assumptions regarding the drawing, such as that links are *joined* to nodes at each end and that nodes are drawn using certain standard objects.

4.3 Contributed Scripts

User contribution of *XSLT* scripts has recently been implemented. At first sight XSLT does not seem to be a natural language for analysing and designing communication networks. However, if additional functions are provided, from a library written in Java (e.g. functions for finding shortest paths), it turns out that many tasks which have previously been regarded as complex (e.g. deciding on the capacity all links) are quite easy to do.

An example of the use of XSLT for a network task is shown in Figure 7.

5 The Repository

5.1 A Database of Networks

The Netml Repository is an important feature of the Netml website. Users may select an existing network rather than create a new one from scratch. Tools may be applied directly to networks in the repository, or alternatively, a network from the repository may be download and edited before applying the tools.

Users may submit their own networks for inclusion into the repository. When a network file is submitted, it will be validated against the Netml schema before it is being accepted. Users may view the network as a picture once it is submitted.

5.2 A Database of Algorithms

It is also possible to submit software to the repository. The existence of a standard language for describing networks is essential to allow sharing of software. The characteristics of XML, as a data language, increases the benefits of sharing because of the easy access to data which it provides, and in particular because we can use the very powerful document transformation language, XSLT, to encode these algorithms.

The Netml web site already uses XSLT as the language for implementation of several algorithms already, namely the algorithms for conversion to postscript, the algorithm for conversion of dia files to netml files, a simple algorithm for costing a network, and an algorithm for adding traffic streams to a network.

For encoding simple scripts which carry out useful functions, XSLT is ideal. As an example, Figure 7 shows the key fragment of a script, available for use on the web site, which adds a complete collection of traffic streams to a network:

```
<trafficStreams>
  <xsl:for-each select="/nml:network/nodes/node">
    <xsl:variable name="origin" select="name"/>
      <xsl:for-each
           select="/nml:network/nodes/node">
        <xsl:variable name="destination"
           select="name"/>
        <xsl:if test="$origin != $destination">
        <trafficstream>
          <name>
            t_<xsl:value-of select="$origin"/>
             _<xsl:value-of select="$destination"/>
          </name>
          <origin>
            <xsl:value-of select="$origin"/>
          </origin>
          <destination>
            <xsl:value-of select="$destination"/>
          </destination>
          <mean>1</mean>
        </trafficstream>
      </xsl:if>
    </xsl:for-each>
  </xsl:for-each>
</trafficStreams>
```

Figure 7: XSLT code for adding traffic streams to a network

6 Concluding Remarks

In this paper we have presented an XML based language for describing networks called ”network modelling language (Netml)”. This language has been used to define a collection of networks, some based on existing networks, others artificial examples for educational purposes. A collection of algorithms for analysing, modifying, and designing these networks has been developed and made available on the web site. Shared use of the network data has now been in place for some timeand sharing of the algorithms has only recently been implemented. Development of additional algorithms is now greatly facilitated and it is expected that the range of algorithms and methods for manipulation of networks will increase steadily with time.

Acknowledgements

The authors would like to acknowledge the contribution of Peter Salmon in the development of the tool for translating `netml` into postscript.

References

[1] Aixcom. Network planning, network design and optimisation, 2003. http://www.aixcom.com.

[2] Comsof. WDMNetDesign - A tool to Design and Evaluate WDM networks, 2003. http://www.comsof.com.

[3] WANDL. Network Planning and Analysis Tool, 2003. http://www.wandl.com.

[4] J. Frings. Designing ATM Networks Using LINDA. *European Conference on Networks and Optical Communications, Antwer*, January 1996.

[5] W3C Consortium. XML Schema. W3C Recommendation, 2 May 2001. http://www.w3.org/TR/xmlschema-0/.

[6] M. Tornatore and G. Maier and A. Pattavina. WDM network optimization by ILP based on source formulation, 2002.

[7] R. Taylor and R. Addie. An Algorithm for Calculating the Availability and Mean Time to Restore for Communication Through a Network. In N.M. Van Dijk, P. Gerrand, W.T. Henderson, R.E. Warfield and R.G. Addie, editor, *Traffic Theories for New Telecommunication Services*, pages 109 – 114. North-Holland Elsevier Science Publishers B.V., 1990.

[8] R. W. Floyd. Algorithm 97: Shortest path. *Communications of the ACM*, 5(6):345, 1962.

[9] The Apache Software Foundation. Xerces2 Java Parser, 2002. http://xml.apache.org/xerces2-j/index.html.

[10] JAXP. Java API for XML processing. http://java.sun.com/xml/jaxp.

[11] W3C Consortium. XSL Transformation (XSLT) Version 1.0. W3C Recommendation, 16 November 1999. http://www.w3.org/TR/xslt/.

[12] W3C Consortium. XML Path Language (XPath) Version 1.0. W3C Recommendation, 16 November 1999. http://www.w3.org/TR/xpath.