A FUZZY ENABLED EDGE NETWORK DEVICE

SIMULATION SOFTWARE APPLICATION

A Project

Presented

To the Faculty of

California State University, Chico

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

in

Interdisciplinary Studies:

Science Teaching

by

© Eric W. Yocam 2004

Fall 2004
A FUZZY ENABLED EDGE NETWORK DEVICE
SIMULATION SOFTWARE APPLICATION

A Project
by
Eric W. Yocam

Fall 2004

APPROVED BY THE INTERIM DEAN OF THE SCHOOL OF GRADUATE, INTERNATIONAL, AND INTERDISCIPLINARY STUDIES:

_________________________________________________________________
Susan E. Place, Ph.D.

APPROVED BY THE GRADUATE ADVISORY COMMITTEE:

_________________________________________________________________
Seung-Bae, Ph.D., Chair

_________________________________________________________________
Benjoe Juliano, Ph.D.
PUBLICATION RIGHTS

No portion of this thesis may be reprinted or reproduced in any manner unacceptable to the usual copyright restrictions without the written permission of the author.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of Tables</td>
<td>v</td>
</tr>
<tr>
<td>List of Figures</td>
<td>vi</td>
</tr>
<tr>
<td>Abstract</td>
<td>vii</td>
</tr>
<tr>
<td><strong>CHAPTER</strong></td>
<td></td>
</tr>
<tr>
<td>I. Introduction to the Project</td>
<td>1</td>
</tr>
<tr>
<td>Purpose of the Project</td>
<td>1</td>
</tr>
<tr>
<td>Scope of the Project</td>
<td>1</td>
</tr>
<tr>
<td>Significance of the Project</td>
<td>4</td>
</tr>
<tr>
<td>Limitations of the Project</td>
<td>4</td>
</tr>
<tr>
<td>Definitions of Terms</td>
<td>5</td>
</tr>
<tr>
<td>II. Review of Relevant Literature</td>
<td>7</td>
</tr>
<tr>
<td>Fuzzy Logic Background</td>
<td>8</td>
</tr>
<tr>
<td>The Edge Device</td>
<td>9</td>
</tr>
<tr>
<td>The Quality of Service Principle</td>
<td>12</td>
</tr>
<tr>
<td>Differentiated Services on Networks</td>
<td>12</td>
</tr>
<tr>
<td>Network Topologies</td>
<td>14</td>
</tr>
<tr>
<td>Centralized Network</td>
<td>14</td>
</tr>
<tr>
<td>Distributed Network</td>
<td>15</td>
</tr>
<tr>
<td>Combined Centralized and Distributed Network</td>
<td>16</td>
</tr>
<tr>
<td>MPLS Routing</td>
<td>17</td>
</tr>
<tr>
<td>Enhanced MPLS Routing</td>
<td>20</td>
</tr>
<tr>
<td>The Fuzzy-enabled Edge Device</td>
<td>20</td>
</tr>
<tr>
<td>Blurring the Network’s Edge</td>
<td>24</td>
</tr>
<tr>
<td>Current Fuzzy-enabled Edge Device Uses</td>
<td>26</td>
</tr>
<tr>
<td>III. Methodology</td>
<td>28</td>
</tr>
<tr>
<td>IV. Results</td>
<td>34</td>
</tr>
</tbody>
</table>

iv
<table>
<thead>
<tr>
<th>CHAPTER</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>V.  Summary</td>
<td>41</td>
</tr>
<tr>
<td>VI. Conclusions</td>
<td>43</td>
</tr>
<tr>
<td>VII. Recommendations</td>
<td>44</td>
</tr>
<tr>
<td>References</td>
<td>45</td>
</tr>
<tr>
<td>Appendices</td>
<td></td>
</tr>
<tr>
<td>A.  Fuzzy Priority Control Scheme</td>
<td>48</td>
</tr>
<tr>
<td>B.  Data Structures Used</td>
<td>53</td>
</tr>
<tr>
<td>C.  UI Specification</td>
<td>56</td>
</tr>
<tr>
<td>D.  Max-Min Inference and Defuzzification Equations Based on COG Approach</td>
<td>73</td>
</tr>
<tr>
<td>E.  Minimum FCL Compliance and a Listing of Production Rules</td>
<td>75</td>
</tr>
</tbody>
</table>
# LIST OF TABLES

<table>
<thead>
<tr>
<th>TABLE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. World Market Breakup for the Public Telecommunications Infrastructure in 2001</td>
<td>8</td>
</tr>
<tr>
<td>2. Core and Edge Device Characteristics</td>
<td>10</td>
</tr>
<tr>
<td>3. Results When No Fuzzy Logic Was Applied To the System</td>
<td>35</td>
</tr>
<tr>
<td>4. Results When Fuzzy Logic Was Applied To the System</td>
<td>36</td>
</tr>
<tr>
<td>5. Results When No Fuzzy Logic and High Priority QOS Set at 80%</td>
<td>37</td>
</tr>
<tr>
<td>6. Results When Fuzzy Logic Was Applied and High Priority QOS Set at 80%</td>
<td>38</td>
</tr>
<tr>
<td>7. Results When No Fuzzy Logic Was Applied and Network Congestion Set at 20%</td>
<td>39</td>
</tr>
<tr>
<td>8. Results When Fuzzy Logic Was Applied and Network Congestion Set at 20%</td>
<td>40</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>TABLE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Simulation Screen with No Fuzzy Logic Controls</td>
<td>2</td>
</tr>
<tr>
<td>2. Simulation Screen with Fuzzy Logic Controls</td>
<td>3</td>
</tr>
<tr>
<td>3. Edge Switch Configuration within a Network</td>
<td>11</td>
</tr>
<tr>
<td>4. Example of a Centralized Network</td>
<td>15</td>
</tr>
<tr>
<td>5. Distributed Network Example</td>
<td>16</td>
</tr>
<tr>
<td>6. An MPLS Network Example</td>
<td>19</td>
</tr>
<tr>
<td>7. An Example of a Reference Model that Incorporates a Fuzzy Priority Control Device Where Linguistic Variables are the Inputs and Action is the Output Variable</td>
<td>21</td>
</tr>
<tr>
<td>8. An Example of a Membership Function for the N Input Variable</td>
<td>23</td>
</tr>
<tr>
<td>9. An Example of a Membership Function for the DN Input Variable</td>
<td>24</td>
</tr>
<tr>
<td>10. An Example of a Membership Function for the CLR Input Variable</td>
<td>25</td>
</tr>
<tr>
<td>11. An Example of a Membership Function for the Action Output Variable</td>
<td>26</td>
</tr>
</tbody>
</table>
ABSTRACT

A FUZZY ENABLED EDGE NETWORK DEVICE SIMULATION SOFTWARE APPLICATION

by

© Eric W. Yocam 2004

Master of Science in Computer Science

California State University, Chico

Fall 2004

The objective of this master’s project was to build a simulation model represented through the creation of an interactive software application. The simulation model demonstrates that a conventional network edge routing device can be enhanced to produce a high degree of performance achieved with the application of fuzzy logic in contrast to the device performance achieved that does not have fuzzy logic applied to it. The simulation model demonstrates two different (yet simplified) scenarios, where one scenario exhibits routing of network traffic found in a conventional network edge routing device (also called an intelligent edge device) and the other scenario exhibits routing of network traffic, but with an enhancement made to it that incorporates a number of fuzzy logic principles to the priority control mechanism of the intelligent edge device.

The resulting contrast between these two scenarios demonstrates that by
incorporating fuzzy logic principles in a conventional network edge routing device, the result is a much higher degree of performance obtained with no fuzzy logic implementation. Even though the simulation model just emulates two very simplified and conceptual scenarios, the software application enables the user to experiment by making changes in the control parameters and displays how these changes affect the overall simulation model.

The simulation model demonstrates that an intelligent edge device that incorporates fuzzy logic not only blurs the network’s edge toward ubiquitous network topology, but also enables enhanced support of complex protocols in order to deliver a new breed of services to the consumer. Further, the enhanced support is developed by combining fuzzy logic, the principle of Quality of Service, and an enhanced version of Multi-protocol Label Switching that can be used in combination to address many of the challenges facing today’s networks.
CHAPTER I

INTRODUCTION TO THE PROJECT

Today’s network device has not reached its maximum potential in terms of ensuring Quality of Service (QoS) for network services delivered to network users—or, from the Internet service provider perspective, to subscribers.

Current TCP/IP congestion control algorithms cannot efficiently support new and emerging services needed by the Internet community. In fact, one could argue that network congestion control still remains a critical and high priority issue, especially given the growing size, demand, and speed (bandwidth) of the network [17].

Purpose of the Project

The purpose of this project was to build a simulation model represented through the creation of a software application used by a software user (Figure 1). The simulation model demonstrates that a conventional network edge routing device can be enhanced to produce a high degree of performance with the application of fuzzy logic in contrast to the device performance that does not have fuzzy logic applied to it.

Scope of the Project

By combining MPLS protocol, fuzzy logic, and an edge network device, I purport that network traffic can be prioritized between high and low priority traffic and then routed more effectively, thereby increasing the total throughput at the edge of the network.
Fig. 1. Simulation screen with no fuzzy logic controls.

network while maintaining a QoS goal for high traffic demand. This will be shown through the development of simulator software where packets will be randomly prioritized, packets will be transferred along a preconfigured path, and at each stop the packet will be compared to a device policy. Each device policy will control the acceptance or rejection of the packet based on whether fuzzy logic control is enabled or not enabled. The fuzzy logic control is based on a fuzzy logic class library developed by a company called Louder Than a Bomb (http://ffll.sourceforge.net). The simulator provides output after each trial is run by the user. This output is divided between results from enabling fuzzy logic and disabling it and then compared to support the purported conclusion of this paper that, in combination with MPLS and an edge network device using fuzzy logic, translates into a slight increase in throughput or network performance over not using fuzzy logic.

The simulation model demonstrates two different (but simplified) scenarios where one scenario exhibits routing of network traffic found in a conventional network.
edge routing device (also called an intelligent edge device) and the other scenario exhibits routing of network traffic, but with an enhancement that incorporates fuzzy logic to the priority control mechanism of the intelligent edge device.

The contrast between these scenarios’ results demonstrates that this simulation model (by incorporating fuzzy logic in a conventional network edge routing device) results in a much higher degree of performance than with one found where no fuzzy logic was implemented. Even though the simulation model just emulates two very simplified and conceptual scenarios, the software application does enable the user to experiment by making changes in the control parameters and seeing how these changes affect the overall simulation model (Figure 2).

Fig. 2. Simulation screen with fuzzy logic controls.
The simulation model demonstrates that an intelligent edge device that incorporates fuzzy logic leads to each device’s throughput to increase the number of high priority packets within the network. Also, by demonstrating that the enhanced support found by combining fuzzy logic, the principle of QoS, and an enhanced version of Multi-protocol Label Switching (MPLS), the simulation model can be used for experimenting with setting different combinations and analyzing the results. Additional User Interface (UI) specification details can be found in Appendix C.

Significance of the Project

Applying fuzzy logic to edge network devices combined with other enhancements is an exciting field with many possible directions. I have a good grasp of the state of the art from my work experience; combined with research in past courses and with this project, I hope to generate and encourage more interest in this field of study.

An interactive simulator software application allows a user to explore and ask whether or not the application of fuzzy logic has any effect on network performance.

Limitations of the Project

There are a number of limitations of the project that must be identified—finite routing paths, primitive router and host behavior, and a simulation model that does not support concurrency. First, by minimizing the number of routing options to having only three possible route directions, it simplifies the comparison made within the simulation between the nonfuzzy logic and fuzzy logic approaches. Second, the representation of both the routers and hosts is based on very primitive programming constructs (e.g., data
structures representing single linked list and queue). Finally, the simulation model does not support real world concurrency in order to simplify the model.

However, all of these limitations can be overcome given a significant amount of time and effort that can expand on this very simplified version of the SIM software to make it a more realistic representation found in the real world.

Definitions of Terms

• **Edge device** – A physical device that can pass packets between a legacy type of network and an ATM network, using Data Link Layer and Network Layer Information (an example of an edge device is an edge router).

• **Multi-protocol label switching** – a short fixed-length label that represents an IP packet’s header with subsequent routing decisions, based on the MPLS label and not the original IP address.

• **Quality of service** – refers to a broad collection of networking technologies and techniques where the goal is to provide guarantees on the ability of the network to deliver predictable results.

• **Fuzzy logic** – a type of logic that recognizes more than simple true and false values with propositions that can be represented with degrees of truthfulness and falsehood.

  • **Ingress** – the act of entering.

  • **Egress** – the act of leaving.

• **Centralized network** – a network topology where a central set of servers control both services and information.
- **Distributed network** – a network topology where a distributed set of servers control both services and information.

- **Router** – a device that forwards data packets along networks and that must be connected to at least two networks.

- **Network switch** – a device that joins multiple computers together at a low-level network protocol level and operates at either layer 2 or layer 3.

- **Defuzzification** – a method to return a single crisp value from a set of fuzzy values.

- **Packet** – a piece of a message transmitted over a packet-switching network where it contains a destination address in addition to the data.

- **ATM** – a network technology based on transferring data in cells or packets of a fixed size.

- **Cell** – used with ATM, it is also referred to as packets of a fixed size.
A network’s edge is arguably one of its most important areas because this is where it delivers services to network users—or, from the Internet service provider perspective, to subscribers. Current TCP/IP congestion control algorithms cannot efficiently support new and emerging services needed by the Internet community [1]. In fact, one could argue that network congestion control still remains a critical and high priority issue, especially given the growing size, demand, and speed (bandwidth) of the increasingly integrated services network [2]. These devices at the network’s edge classify, prioritize, and mark packets for the rest of the network to understand, and ultimately allow them into the network. A proposal to effectively address this issue is in implementing a fuzzy logic-controlled queue that is not only novel, more effective, robust, and scalable, but it is also a more flexible approach. So given modern networks’ complex internetworking, it is not surprising that identifying, developing, and incorporating fuzzy logic into edge devices in order to minimize routing congestion are tasks that head the agendas of most of today’s network equipment manufacturers, carriers, service providers, and network administrators. Many network equipment manufacturers are now providing, as part of their product line, two different classes of network routers—core router and edge router (Table 1) [3].
Table 1

World Market Breakup for the Public Telecommunications Infrastructure in 2001

<table>
<thead>
<tr>
<th>Segment</th>
<th>Market size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core routers</td>
<td>$1.7B</td>
</tr>
<tr>
<td>Edge routers</td>
<td>$2.4B</td>
</tr>
<tr>
<td>SONET/SDH/WDM</td>
<td>$28.0B</td>
</tr>
<tr>
<td>Telecom MSS</td>
<td>$4.5B</td>
</tr>
</tbody>
</table>

This project begins with a brief background on fuzzy logic and then proceeds to define an edge device in the context of a network topology. Next, it is necessary to show how the quality of service principle supports the edge device’s capability to provide differentiated services on the network’s edge. Finally, an enhanced version of MPLS routing, combined with fuzzy logic, purports the notion of blurring the network’s edge with the implementation of fuzzy-enabled edge devices.

Fuzzy Logic Background

Fuzzy logic is a technology for developing intelligent control and information systems. There are three areas of fuzzy logic from the viewpoint of machine intelligence, control systems, and information technology. These three areas include (1) a way to represent and structure human knowledge that is imprecise by its very nature, (2) a very useful design technique that can be applied to nonlinear control systems where the basic building block consist of “if-then” rules (also known as fuzzy rule-based models) and used to approximate traditional functional mapping, and (3) an increasingly important
aspect of storing and retrieving imprecise information from databases and information
sources within advanced information systems [4]. Fuzzy logic can be viewed as a super-
set of conventional (Boolean) logic that has been extended to multi-valued logics. Fuzzy
logic is an approach to computing based on “degrees of truth” rather than the usual “true
or false.” The idea of fuzzy logic was first proposed by Dr. Lotfi Zadeh of the University
of California in Berkeley (in the 1960s) [18]. At that time, Dr. Zadeh was working on the
problem of computers’ understanding of natural language [19]. Natural language (like
most other activities in life) is not easily translated into absolute terms such as “true or
false.” In fact, fuzzy logic seems closer to the way our brains work; that is, we aggregate
data and form sets of partial truths that we in turn aggregate further into higher truths
once certain thresholds are exceeded [5].

The Edge Device

Cisco [1] defines an edge device as

. . . a physical device that can pass packets between a legacy type of network
such as an Ethernet network and an asynchronous transfer mode (ATM) net-
work, using data link layer (OSI Layer 2) and network layer (OSI Layer 3) in-
formation. An edge device does not have responsibility for gathering network
routing information, but simply uses the routing information it finds in the
network layer using the route distribution protocol. (p. 292)

For example, an edge router (sometimes called a boundary router) routes data
between one or more local area networks (LANs) and an ATM backbone network—
whether it is a campus network or wide-area network (WAN). In this case, the edge
device has the difficult task of aggregating, forwarding, and routing traffic.
We can contrast the edge router (which forwards packets to computer hosts within a network) with a core router (which forwards packets between networks). A core device or core router is part of the backbone and it serves as the single pipe through which all traffic from peripheral networks must pass on its way to other peripheral networks. Unlike core routers, which primarily switch packets, edge devices not only handle basic IP (Internet Protocol) packet processing, but also consolidate multiple wire-speed traffic streams into flows with appropriate QoS parameters, perform traffic conditioning, and provide a multiplicity of services (see Table 2 for a comparison of some of the characteristics and functions of core and edge devices) [6].

Table 2
Core and Edge Device Characteristics

<table>
<thead>
<tr>
<th>Device type</th>
<th>Throughput (Gbps)</th>
<th>Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core</td>
<td>80 to 320</td>
<td>Packet forwarding</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Route processing</td>
</tr>
<tr>
<td>Edge</td>
<td>6 to 24</td>
<td>Packet forwarding</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Route processing</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Traffic shaping</td>
</tr>
<tr>
<td></td>
<td></td>
<td>QoS scheduling</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Service features</td>
</tr>
</tbody>
</table>

Another example of an edge device is an edge switch (see Figure 3, an example of an edge switch configuration within a network) [6]. These operate in various edge networks that interconnect through a high-speed, high-capacity core network which, in turn, connects to the outside world.
In addition to edge routers and switches, other types of edge devices include aggregation routers, service delivery, next-generation voice, and multi-service provisioning platforms. Many edge devices, designed to bridge the gap between consumer demand and core capacity, play more than one of these roles. A single device might incorporate the functions of an aggregation router, IP service switch, and carrier-class router; serving all these functions would require high degrees of reliability, flexibility, scalability, and performance.

For example, to terminate and aggregate diverse traffic from the network, edge devices must support various interfaces and speeds, and they must also manage hundreds of thousands of individual concurrent sessions, each with a unique profile —1/E1 channel groups, frame relay sessions, ATM private virtual circuits (PVC), Ethernet virtual local area networks (VLAN), and so on.
The Quality of Service Principle

QoS is a key network principle for the transmission and distribution of digitized audio/video across next-generation high-speed networks. It has two objectives: finding routes that satisfy the QoS constraints and making efficient use of network resources. In this fashion, QoS mechanisms provide the necessary level of service (bandwidth and delay) to an application in order to maintain an expected quality level. To a mission-critical application, QoS means guaranteed bandwidth with zero frame loss. Fine-grain control provided by QoS places a significant burden on the network infrastructure. Each device must keep an entry in its forwarding table for each flow. In a large corporate network, devices can become overwhelmed with the millions of flows, especially at the boundaries. For example, MPLS is a protocol that delivers a level of service by application flow that could combine fuzzy logic with QoS to allow multiple constraints to be considered in a simple and intuitive way [7].

Differentiated Services on Networks

Increasing demand for inter-network access, technology advancements at both the desktop level and the network level, and the increasing use of multi-media, as well as other interactive technologies, are burdening the core network devices and LAN/WAN boundary routers in busy corporate networks. Service quality will continue to suffer without QoS mechanisms or traffic-based classification schemes such as differentiated services.

In response to demand for a robust, common system for service classification, an IETF Working Group drafted a framework and definitions for a differentiated services
mechanism [8]. Differentiated services are not based on priority, application, or flow, but on the possible forwarding behaviors of packets, called Per Hop Behaviors (PHBs).

Unfortunately, in today’s corporate network, QoS is a costly and difficult implementation. Differentiated services can alleviate bottlenecks by more efficient management of current corporate network resources. A differentiated service is a policy-based management tool for networks. By mapping multiple flows to a few service levels, differentiated services reduce the burden on network devices and easily scale with network growth.

Since a differentiated service is rule based, it is a unique mechanism for policy-based network management (making it a natural candidate for fuzzy logic). Instead of applying faster, more expensive, advanced technology, networks can be managed by appropriate network policies, applying current network technology and resources while considering in-house traffic and upstream and downstream networks, whether they are the corporate LAN backbone or external WANs.

For example, a differentiated service is analogous to travel services. A person can travel by bus, train, or airplane, first class, business class, coach, or standby. Each class of service can be characterized by how fast you reach your destination, how many stops you make along the way, and what kind of service amenities received en route, if any. Some services may have limitations, such as when you can travel, and others, such as standby, include risk of not reaching your destination in the time frame expected. In all cases, you pay more for higher quality services. The differentiated services framework offers the same kind of classification system. Based on network policies, different kinds
of traffic can be marked for different kinds of forwarding whereby resources can then be allocated according to the marking and the policies.

Network Topologies

In considering how best to deploy edge devices, we should examine two network architecture types—centralized and distributed. From these two architecture types, network administrators derive three major topological approaches—centralized, a mixture of centralized and distributed, and distributed.

Centralized Network

The centralized network topology assumes a collapsed backbone—a backbone network in which an intelligent edge device forwards a series of frames while the central device analyzes each frame and makes decisions (refer to Figure 4, an example of a centralized network [1]). From the network management perspective, this approach is relatively easy to implement. However, it means dealing with frames at a very high line rate—10 to 100 times faster than with a distributed approach. The quantity of policies and flows that the backbone must support is dramatically higher, making performance unpredictable. Lack of redundancy is another shortcoming of this approach; the central device becomes a single point of failure. A centralized network management device distributes the set of rules by which these devices operate. The central devices are pipe only, which means that they have high-performance switching and queuing capabilities but do limited analysis. For example, in a Multi-protocol Label Switching approach, edge
devices do all the flow treatment, and add or extract labels. The core devices simply perform label swapping and forward frames.

![Diagram of a centralized network](image)

**Fig. 4. Example of a centralized network.**

**Distributed Network**

A distributed network (refer to Figure 5, an example of a distributed network) that uses intelligent devices at the network edge is the preferable approach [6]. These devices perform frame analysis, classification, remarking, tagging, traffic shaping, and routing. They also perform analysis and decision-making, though at relatively slower speeds than those of centralized network devices.

Although the centralized approach is easier to implement, the distributed network is more scalable, less sensitive to network failures, and has no single point of
failure. Distributed networks are more cost-effective and yield more robust networks, with significantly better overall performance and efficiency [6].

![Distributed network example](image)

**Fig. 5. Distributed network example.**

**Combined Centralized and Distributed Network**

The combined centralized and distributed topology assumes that the edge device is not simple and can do some classification and policy treatment between the backbone node and the edge device. However, the combined approach suffers from multiple areas of weakness [9]:

- it adds significant complexity and cost to the network without any appreciable gains—there are more nodes to install, manage, and support;
• the topology requires a new type of centralized device in the network architecture—a device has the same limitations as the central device in the centralized-network approach with no increase in benefits;

• sharing QoS responsibilities between edge and middle devices that do not conform to the well defined MPLS approach (described in the next section) leads to configuration problems and inefficiency—the network effectiveness deteriorates because some fields of the packets must be inspected twice (devices in the network must implement both QoS and traffic standards burdening the network with additional traffic management overhead); and

• the middle device handles intensive queue management and flow-related activities—as a result, it adds latency and sometimes unpredictable jitter that adversely affects multimedia applications.

MPLS Routing

Currently there are two major applications of MPLS in IP networks: traffic engineering (TE) and virtual private networks (VPNs) [10]. Traffic engineering is best viewed as the combination of functions that allows a network provider to control and monitor bandwidth and paths for traffic flows. It encompasses capacity planning, routing control, traffic management, and path management. MPLS enables network-based IP-VPNs, a service offered by the network service provider that appears to the customer as a private network. VPNs benefit from MPLS because of the notion of tunnels.

The Internet Engineering Task Force [8] established the MPLS standards to achieve several capabilities, a few of which are
• Fast forwarding speed.

• Traffic engineering, including constraint-based routing, explicit routing, and the abilities to compute a path at the source, reserve network resources, and modify link attributes.

• Support for voice and video on IP, including the ability to handle delay variation plus QoS constraints.

• Support for virtual private networks, including the use of a controllable tunneling mechanism, equivalent to a frame relay or ATM virtual circuit (VC).

As MPLS uses only the label to forward packets, it is protocol-independent, hence the term “Multi-protocol” in MPLS. MPLS uses IP addresses (either IPv4 or IPv6) to identify end points and intermediate switches and routers. The IP-compatibility and easy integration with traditional IP networks make MPLS a universally accepted de facto standard as the flow-management model of choice [11]. However, unlike traditional IP, an MPLS flow is connection-oriented and its packets are routed along preconfigured Label Switched Paths (LSPs). In an MPLS network, first an edge–label-switching router (edge LSR) or ingress router assigns each incoming packet a label where the packet’s destination address determines which LSP to use. Second, the router forwards the packet along a label-switched path, where each LSR makes forwarding decisions based solely on the label contents. Third, at each hop, the LSR strips the existing label and applies a new one that tells the next hop how to forward the packet. Finally, as the labeled packet leaves the network, another edge LSR (egress router) removes the label (refer to Figure 6, an example of an MPLS network) [12]. In addition, as the priority algorithms have to act on
the forwarding path of the LSR, they cannot be excessively complex, given the requirement to execute as rapidly as possible. It is here, within the LSR priority algorithm mechanism, that fuzzy logic is a natural application candidate.

Fig. 6. An MPLS network example.

For example, one can use MPLS to forward packets over the Internet backbone through the use of a VPN—through either border gateway protocol (BGP) or MPLS VPNs in either OSI layer 2 or layer 3. Used together, the MPLS signaling and VPN protocols establish a VPN tunnel, also known as label-switched paths, through which data travels. In addition, you can set up the MPLS label edge router to maintain multiple tunnels simultaneously.
Enhanced MPLS Routing

A newly published RFC (available online), titled “Requirements for Support of Differentiated Services-Aware MPLS Traffic Engineering” (RFC3564), describes various application scenarios identified by Service Providers where existing MPLS traffic engineering (TE) mechanisms fall short and where differentiated services aware TE can address the needs. In fact, MPLS with guaranteed bandwidth services are extending MPLS traffic engineering—advertise available bandwidth for best-effort traffic and advertise available bandwidth for high priority traffic or differentiated services aware TE by utilizing QoS features to guarantee delivery of the high priority traffic (that includes classification with policing) [12].

For example, the corporate network can be divided into a core and edge topology configuration where the core network provides guaranteed bandwidth services (assuring value-added services with traffic engineering) and the edge of the network consists of existing MPLS TE classification with policing to ensure that there is no theft of the edge service provided to the customer. So, by defining a priority control scheme based on fuzzy logic (that respects the QoS of high priority traffic) combined with enhanced MPLS routing (that includes differentiated service awareness) applied to a traditional edge device results in a fuzzy-enabled edge device.

The Fuzzy-enabled Edge Device

The objective of a fuzzy-enabled edge device (attached to the Internet) is to increase network capacity and offer a high quality of differentiated services for traffic with real-time and non-real-time requirements. An assumption made for Ma, Yannis, and
Zhang’s [13] proposed solution is that some switch resources (e.g., buffer size, Q, and bandwidth portion, C) are allocated to high-priority traffic to guarantee a given QoS. To help clarify this point, a reference model is given in Figure 7 where low-priority cells are allowed to enter the allocated buffer to improve the exploitation of reserved resources [13]. This solution incorporates a fuzzy control algorithm to select packets in a fair and efficient manner and is ultimately implemented in a hardware based solution (for in-
stance, implementation would come in the form of exploiting the advantages of VLSI technology) where the processor could achieve optimal performance in both processing capacity and latency [14].

In order to meet the demand for QoS threshold (for a particular real-time or non-real-time requirement), the differentiated service relies on a priority control function to maximize network utilization without degrading the QoS for high-priority traffic. The existence of low-priority packets could be due [14]: (1) to the tagging action of traffic enforcement mechanisms; or (2) to the traffic of an agency for which a certain QoS is guaranteed. The agency could exploit the resources allocated to real-time traffic (high-priority flow) to transmit best-effort traffic (low-priority).

The fuzzy logic based-priority control function proposed by Catania, Ficili, Palazzo, and Panno [14] consists of three input variables, one output variable, membership functions, 80 fuzzy rules, and an inference method. The three input parameters, the Cell Loss Ratio experienced in the buffer by High-priority traffic (CLR), the number of cells in the buffer (N), and the rate at which the length of the queue in the buffer varies or delta N (DN) are considered to be linguistic variables. There is one output variable called Action (a discrete value) and it is composed of six fuzzy sets whose names are Strong Discard (SD), Discard (D), Marginal Discard (MD), Marginal Admit (MA) Admit (A), and Strong Admit (SA). There are a total of four membership functions where there is one membership function per parameter—CLR, N, DN, and Action (refer to Figures 8-11 for examples of four the membership functions) [14].
A requirement when defining fuzzy rules is to make them as simple as possible. In this case, even though there are over 400 combinations possible, the priority control function requires only a total of 80 fuzzy rules that are highly representative of the decision-making process (refer to Appendix A for a list of fuzzy rules that represent the fuzzy priority control scheme) [14]. The fuzzy rules are based on intuitive considerations and then tuned by simulation test using the Genesis simulator with a Genetic Algorithm approach [14].

The inference method is Max-Min and for defuzzification the Center of Gravity (COG) approach is recommended [14]. Inference supplies a membership function as a result. A representative value cannot directly process this fuzzy information, therefore the result of the inference process has to be converted into crisp numerical values. In this context, the crisp number to be determined should provide a good representation of the information contained in the membership function. The Center of Gravity approach
addresses the problem of comparison of fuzzy numbers and it is to associate with a fuzzy number F some representative value, $\text{Val}(F)$, and to compare the fuzzy subsets using these single representative values [15]. In addition, the equations can be found in Appendix D.

![An example of a membership function for the DN input variable.](image)

**Fig. 9.** An example of a membership function for the DN input variable.

**Blurring the Network’s Edge**

The network edge is no longer a simple demarcation point within a particular network; rather, it must contain some of the intelligence to meet the requirements of new technologies and of the business and regulatory environments. If customers are to receive reliable, affordable service—and if providers are to do business in a way that generates profit—network administrators need new tools to efficiently manage devices at the network edge [20].
Today, a new kind of IP network is evolving to meet the converging needs and technologies of several market segments—customers, ISPs, corporations, traditional telephony carriers, and other new types of customer-facing service providers. And the place where these market segments come together is the network edge. Understanding the important aspects of network topologies, fuzzy-enabled edge devices, and enhanced MPLS routing can help guide the network’s evolution to participate effectively and profitably in this convergence and even blur it to the point of a ubiquitous network. For example, enhanced MPLS routing can provide edge-assisted QoS to networks where, in the past, only the edge routers can participate in QoS tagging. This protocol combines differentiated services and call-admission control function that sets up a gatekeeper for call admission and establishes priority classes for IP traffic and, in a sense, blurs the network’s edge as a conventional demarcation point [16].
Fig. 11. An example of a membership function for the action output variable.

Current Fuzzy-enabled Edge Device Uses

The evolving edge device is extremely versatile, especially with the application of fuzzy logic. It must support several transport technologies and interface speeds to provide aggregation and connectivity among different network topologies and elements. Such fuzzy-enabled edge devices also support an increasing number of complex protocols and packet encapsulations to manage diverse network topologies and deliver new services and applications [17]. With the growing popularity of broadband services and third-generation, mobile wireless devices, tomorrow’s edge devices utilizing differentiated services traffic engineering and enhanced MPLS will have to scale to even larger table loads.

These forces are driving the modern network architecture to a more hierarchical-based structure with a fast, converged core combined with an intelligent edge [17]. As network devices become more specialized (to support a more hierarchical-based struc-
ture) and begin to incorporate intelligent systems techniques such as fuzzy logic into existing network devices (e.g., routers, switches, and aggregation multiplexers), a new breed of network device will emerge. Tomorrow’s fuzzy logic-enabled edge device will function much more effectively than its predecessor (e.g., Ethernet, frame relay, and ATM edge switches; IP service platforms; and carrier-class routers). We can anticipate that by exchanging information between the network core and broadband access networks, these sophisticated edge devices will not only simultaneously support widely diverse protocols, interfaces, and line rates, handle heavy control- and data-plane loads, but will also provide security, QoS, and many other IP services—all at wire speed [17].
CHAPTER III

METHODOLOGY

The work presented here aims to provide a contrast between two scenario results. A simulation model is used to demonstrate incorporating fuzzy logic principles in a conventional network edge routing device; the result is a marginal increase in performance than one with no fuzzy logic implementation. Even though the simulation model just emulates two very simplified and conceptual scenarios, the software application enables the user to experiment by making changes in the control parameters and displays how these changes affect the overall simulation model (Figure 12).

The fundamental design is to optimize the amount of high priority traffic that is routed through each device along a path and increase the overall throughput of each of the fuzzy logic-enabled devices. When the inbound queue of the device is not full every inbound packet, regardless of its priority, is accepted but when the queue is full and a low-priority packet arrives, it will be discarded. Conversely, when a high-priority packet arrives, the low-priority packet is removed from the inbound queue (of course if a low-priority packet is in the inbound queue). If there are no low-priority packets waiting, the arriving packet is discarded (Figures 13 and 14).

The software application was written using the Microsoft Visual C# .NET development platform. It runs on Microsoft Windows 2000 and XP desktop operating system as a client software application. In order to not recreate a brand new fuzzy model
Fig. 12. Simulator process flow without fuzzy logic applied.
with the SIM, an available open source fuzzy logic class library and API was used. The Free Fuzzy Logic Library (FFLL) (http://ffll.sourceforge.net) is an open source fuzzy logic class library and API that adheres to the IEC 61131-7 standard [35]. The FFLL was compiled using a different version of compiler than the project adding to the completing of this project. The FFLL was compiled on the Microsoft Visual C++ development platform and then linked with this project.

The FFLL class library and API provided all the necessary components for a working fuzzy logic model. The fuzzy variables contain sets where each set has a membership function associated with it. The membership function determines the set’s shape in this case where its use is to “fuzzify” the x values of the variable with the associated Degree of Membership (DOM). One of the most common membership function used in fuzzy systems is a triangle however, FFLL provides support for Triangles, Trapezoids, S-Curves, and Singletons. Each set’s membership function contains a lookup table used to speed the “fuzzification” process (however, even though lookup tables are fast the tradeoff is increased use of memory). The variable’s x-axis values must be mapped into to the values array. FFLL stores rules in a one-dimensional array. For example, in a 3x3x3 system (three input variables, each with three sets) there would be 27 total rules. To eliminate extra calculations, FFLL does not calculate the defuzzified output value until it is requested. There are two types of defuzzification approaches available in FFL. The Center of Gravity (COG) defuzzification approach makes heavy use of lookup tables while the Mean of Maximum (MOM) defuzzification approach simply stores a single value per output set. The defuzzification approach used for this project was COG.
The API is straightforward in that it provides seven functions—create a new fuzzy logic model, close the model, load a fuzzy model from a file (based on the IEC 61131-7 Fuzzy Control Language or FCL format, create a child helper process for the model, set input variable in a child, get defuzzified output value for a child, and get message text when an API function returns an error.

The Fuzzy Control Language or FCL syntax (based on the standard for Fuzzy Control Programming published by the International Electrotechnical Commission or
Fig. 14. Simulator process flow with fuzzy logic applied.
IEC) can be found within the IEC specification document 61131-7 [29]. FFLL maintains only the minimum compliance necessary with IEC specification document 61131-7 (refer to a table of minimum FCL compliance and a listing of production rules in Appendix E).
CHAPTER IV

RESULTS

By combining MPLS protocol, fuzzy logic, and an edge network device, I purport that network traffic can be prioritized and routed more effectively, whereby demonstrating a marginal increase in the total throughput at the edge of the network while meeting a QoS goal for high traffic demand.

Expected result was that by using fuzzy logic, the overall throughput would increase with respect to low-priority traffic when high priority traffic is admitted and low-priority traffic is not admitted. In addition, if overall throughput increased, then this implies that the network efficiency also increases, as long as it is not at the expense of noncompliance with the device policy. The throughput was measured in terms of high priority traffic as a percentage of total traffic (high priority traffic and low priority traffic) transferred along a path of connected devices over a period of time. Actual result averaged over five trials with no congestion applied and when no fuzzy logic was applied can be found in Table 3. Actual result averaged over five trials with no congestion applied and when fuzzy logic was applied can be found in Table 4.

It can be concluded from the limited results demonstrated that total throughput marginally increased when fuzzy logic was applied to the system. However, by increasing the number of trials the larger set of results would reinforce the conclusion.
Table 3
Results When no Fuzzy Logic was Applied to the System

<table>
<thead>
<tr>
<th>Trials</th>
<th>No Fuzzy Logic Applied with QoS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>50</td>
</tr>
<tr>
<td>2</td>
<td>53</td>
</tr>
<tr>
<td>3</td>
<td>56</td>
</tr>
<tr>
<td>4</td>
<td>56</td>
</tr>
<tr>
<td>5</td>
<td>56</td>
</tr>
<tr>
<td>Average</td>
<td>54</td>
</tr>
</tbody>
</table>

A couple additional scenarios were considered—increasing high priority load QoS policy threshold and increasing congestion. In the scenario where by increasing the high priority load QoS policy threshold, the number of high priority packets was greater than the number low priority packets but the total throughput remained steady. Actual result averaged over five trials with high priority QoS set at 80% and no fuzzy logic was applied can be found in Table 5.

Likewise, in the same scenario but with fuzzy logic applied, the number of high priority packets was greater than the number low priority packets, but the total throughput remained steady. Actual result averaged over five trials with high priority QoS set at 80% and with fuzzy logic was applied can be found in Table 6.
Table 4
Results When Fuzzy Logic was Applied to the System

<table>
<thead>
<tr>
<th>Trials</th>
<th>Total Throughput (Packets)</th>
<th>Total Throughput (Percent)</th>
<th>Low Priority Load (Packets)</th>
<th>High Priority Load (Packets)</th>
<th>Elapsed Time (Seconds)</th>
<th>Congestion (Percent)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>56</td>
<td>50.00%</td>
<td>28</td>
<td>28</td>
<td>0.328</td>
<td>0%</td>
</tr>
<tr>
<td>2</td>
<td>56</td>
<td>50.00%</td>
<td>28</td>
<td>28</td>
<td>0.39</td>
<td>0%</td>
</tr>
<tr>
<td>3</td>
<td>56</td>
<td>50.00%</td>
<td>28</td>
<td>28</td>
<td>0.515</td>
<td>0%</td>
</tr>
<tr>
<td>4</td>
<td>56</td>
<td>50.00%</td>
<td>28</td>
<td>28</td>
<td>0.593</td>
<td>0%</td>
</tr>
<tr>
<td>5</td>
<td>56</td>
<td>50.00%</td>
<td>28</td>
<td>28</td>
<td>0.484</td>
<td>0%</td>
</tr>
<tr>
<td>Average</td>
<td>56</td>
<td>50%</td>
<td>28</td>
<td>28</td>
<td>0.462</td>
<td>0%</td>
</tr>
</tbody>
</table>

In the scenario where by increasing the overall congestion of the system, the results were found to be the same as with the prior scenarios and the total throughput was steady like in the prior scenarios. However, the high priority packets were given priority at the expense of the low priority packets with network congestions applied. Actual results averaged over five trials for where network congestion was set at 20% and with and without fuzzy logic applied can be found in Tables 7 and 8, respectively.
Table 5

Results When no Fuzzy Logic and High Priority QoS Set at 80%

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>56</td>
<td>78.57%</td>
<td>11</td>
<td>44</td>
<td>0.859</td>
<td>0%</td>
</tr>
<tr>
<td>2</td>
<td>56</td>
<td>78.57%</td>
<td>11</td>
<td>44</td>
<td>0.796</td>
<td>0%</td>
</tr>
<tr>
<td>3</td>
<td>56</td>
<td>78.57%</td>
<td>11</td>
<td>44</td>
<td>0.796</td>
<td>0%</td>
</tr>
<tr>
<td>4</td>
<td>56</td>
<td>78.57%</td>
<td>11</td>
<td>44</td>
<td>0.828</td>
<td>0%</td>
</tr>
<tr>
<td>5</td>
<td>56</td>
<td>78.57%</td>
<td>11</td>
<td>44</td>
<td>0.828</td>
<td>0%</td>
</tr>
<tr>
<td>Average</td>
<td>56</td>
<td>79%</td>
<td>11</td>
<td>44</td>
<td>0.8214</td>
<td>0%</td>
</tr>
</tbody>
</table>
Table 6
Results When Fuzzy Logic was Applied and High Priority QoS Set at 80%

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>56</td>
<td>78.57%</td>
<td>11</td>
<td>44</td>
<td>0.843</td>
<td>0%</td>
</tr>
<tr>
<td>2</td>
<td>56</td>
<td>78.57%</td>
<td>11</td>
<td>44</td>
<td>0.765</td>
<td>0%</td>
</tr>
<tr>
<td>3</td>
<td>56</td>
<td>78.57%</td>
<td>11</td>
<td>44</td>
<td>0.718</td>
<td>0%</td>
</tr>
<tr>
<td>4</td>
<td>56</td>
<td>78.57%</td>
<td>11</td>
<td>44</td>
<td>0.625</td>
<td>0%</td>
</tr>
<tr>
<td>5</td>
<td>56</td>
<td>78.57%</td>
<td>11</td>
<td>44</td>
<td>0.625</td>
<td>0%</td>
</tr>
<tr>
<td>Average</td>
<td>56</td>
<td>79%</td>
<td>11</td>
<td>44</td>
<td>0.7152</td>
<td>0%</td>
</tr>
</tbody>
</table>
Table 7

Results When no Fuzzy Logic was Applied and Network Congestion Set at 20%

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>35</td>
<td>80.00%</td>
<td>0</td>
<td>28</td>
<td>0.828</td>
<td>20%</td>
</tr>
<tr>
<td>2</td>
<td>47</td>
<td>59.57%</td>
<td>0</td>
<td>28</td>
<td>0.828</td>
<td>20%</td>
</tr>
<tr>
<td>3</td>
<td>29</td>
<td>96.55%</td>
<td>0</td>
<td>28</td>
<td>0.812</td>
<td>20%</td>
</tr>
<tr>
<td>4</td>
<td>32</td>
<td>87.50%</td>
<td>0</td>
<td>28</td>
<td>0.812</td>
<td>20%</td>
</tr>
<tr>
<td>5</td>
<td>38</td>
<td>73.68%</td>
<td>0</td>
<td>28</td>
<td>0.812</td>
<td>20%</td>
</tr>
<tr>
<td>Average</td>
<td>36</td>
<td>79%</td>
<td>0</td>
<td>28</td>
<td>0.8184</td>
<td>20%</td>
</tr>
</tbody>
</table>
Table 8
Results When Fuzzy Logic was Applied and Network Congestion Set at 20%

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>56</td>
<td>80.36%</td>
<td>0</td>
<td>45</td>
<td>0.843</td>
<td>20%</td>
</tr>
<tr>
<td>2</td>
<td>56</td>
<td>80.36%</td>
<td>0</td>
<td>45</td>
<td>0.937</td>
<td>20%</td>
</tr>
<tr>
<td>3</td>
<td>56</td>
<td>80.36%</td>
<td>0</td>
<td>45</td>
<td>0.609</td>
<td>20%</td>
</tr>
<tr>
<td>4</td>
<td>56</td>
<td>80.36%</td>
<td>0</td>
<td>45</td>
<td>0.656</td>
<td>20%</td>
</tr>
<tr>
<td>5</td>
<td>56</td>
<td>80.36%</td>
<td>0</td>
<td>45</td>
<td>0.781</td>
<td>20%</td>
</tr>
<tr>
<td>Average</td>
<td>56</td>
<td>80%</td>
<td>0</td>
<td>45</td>
<td>0.7652</td>
<td>20%</td>
</tr>
</tbody>
</table>
CHAPTER V

SUMMARY

This master’s project covers a review of the network’s edge because this is where the network delivers services to network users—or, from the Internet service provider perspective, to subscribers. Current TCP/IP congestion control algorithms cannot efficiently support new and emerging services needed by the Internet community [17]. In fact, one could argue that network congestion control still remains a critical and high priority issue especially given the growing size, demand, and speed (bandwidth) of the increasingly integrated services network [17]. These devices at the network’s edge classify, prioritize, and mark packets for the rest of the network to understand and, ultimately, allow them into the network. In fact, the purported solution and its demonstration through a simulator software shows that an intelligent edge device incorporating fuzzy logic not only blurs the network’s edge toward ubiquitous network topology but also enables enhanced support of complex protocols (packet encapsulations to manage diverse network topologies) in order to deliver new services. The enhanced support is developed by combining fuzzy logic, the principle of Quality of Service (QoS), and an enhanced version of Multi-protocol Label Switching (MPLS) to address many of the challenges facing today’s networks. Given modern networks’ complex internetworking, it should not be a surprise to anyone that identifying, developing, and incorporating intelligent systems techniques such as fuzzy logic into the conventional edge device are at the top agenda item of most
of today’s network equipment manufacturers, carriers, service providers, and network administrators.
CHAPTER VI

CONCLUSIONS

Applying fuzzy logic to edge network device combined with other enhancements is an exciting field with many possible directions. The simulator software developed for this project provides the user the ability to explore different combinations of fuzzy logic configurations applied to simplified network topology. The simulator software demonstrates that there is an advantage to using fuzzy logic over using a traditional device without fuzzy logic.
CHAPTER VII

RECOMMENDATIONS

The limitations for the simulator software can be overcome, given a significant amount of time and effort applied to its current model. The functionality can be expanded and complexities introduced within the simulator software to make it a more realistic representation of the real world. Removing the cap on the number of routing options will facilitate more possibilities with real world network topologies for experimentation with throughput efficiencies. Second, the representation of both the routers and hosts can be made more like real devices by increasing the complexity of the programming constructs. Finally, the simulation model should support concurrency instead of a single threaded programming model.

In order to balance the number of configurations between the network and fuzzy logic features within the simulator software, a number of restrictions were implemented. However, a few of the possible configurations are restricted, such as limited to three routing options, type of membership functions, number of membership functions, number of rules, type of t-norm/r-conorm, and inference approach cannot be changed within the simulator software. The user can manipulate the configuration by selecting one of three networks paths, adjusting congestion, select all or none of the fuzzy logic membership functions, and select all or none of the fuzzy logic rules.
REFERENCES


Fuzzy PRIORITY CONTROL SCHEME

Fuzzy Priority Control Scheme (from [23]).

<table>
<thead>
<tr>
<th>Rule #</th>
<th>Input: CLR</th>
<th>Input: N</th>
<th>Input: DN</th>
<th>Output: Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Low</td>
<td>Low</td>
<td>Negative Big Negative</td>
<td>Strong Admit</td>
</tr>
<tr>
<td>2</td>
<td>Low</td>
<td>Low</td>
<td>Small</td>
<td>Strong Admit</td>
</tr>
<tr>
<td>3</td>
<td>Low</td>
<td>Low</td>
<td>Zero Positive</td>
<td>Strong Admit</td>
</tr>
<tr>
<td>4</td>
<td>Low</td>
<td>Low</td>
<td>Small</td>
<td>Strong Admit</td>
</tr>
<tr>
<td>5</td>
<td>Low</td>
<td>Low</td>
<td>Positive Big</td>
<td>Strong Admit</td>
</tr>
<tr>
<td>6</td>
<td>Low</td>
<td>Medium</td>
<td>Negative Big Negative</td>
<td>Strong Admit</td>
</tr>
<tr>
<td>7</td>
<td>Low</td>
<td>Medium</td>
<td>Small</td>
<td>Admit Marginal</td>
</tr>
<tr>
<td>8</td>
<td>Low</td>
<td>Medium</td>
<td>Zero Positive</td>
<td>Admit Marginal</td>
</tr>
<tr>
<td>9</td>
<td>Low</td>
<td>Medium</td>
<td>Small</td>
<td>Admit Marginal</td>
</tr>
<tr>
<td>10</td>
<td>Low</td>
<td>Medium</td>
<td>Positive Big</td>
<td>Admit</td>
</tr>
<tr>
<td>11</td>
<td>Low</td>
<td>High</td>
<td>Negative Big Negative</td>
<td>Admit Marginal</td>
</tr>
<tr>
<td>12</td>
<td>Low</td>
<td>High</td>
<td>Small</td>
<td>Admit Marginal</td>
</tr>
<tr>
<td>13</td>
<td>Low</td>
<td>High</td>
<td>Zero Positive</td>
<td>Admit Marginal</td>
</tr>
<tr>
<td>14</td>
<td>Low</td>
<td>High</td>
<td>Small</td>
<td>Discard Marginal</td>
</tr>
<tr>
<td>15</td>
<td>Low</td>
<td>High</td>
<td>Positive Big</td>
<td>Discard Marginal</td>
</tr>
<tr>
<td>16</td>
<td>Low</td>
<td>High</td>
<td>Negative Big Negative</td>
<td>Discard Marginal</td>
</tr>
<tr>
<td>17</td>
<td>Low</td>
<td>High</td>
<td>Small</td>
<td>Discard Marginal</td>
</tr>
<tr>
<td>18</td>
<td>Low</td>
<td>High</td>
<td>Zero Positive</td>
<td>Discard</td>
</tr>
<tr>
<td>19</td>
<td>Low</td>
<td>High</td>
<td>Small</td>
<td>Discard</td>
</tr>
<tr>
<td>20</td>
<td>Low</td>
<td>High</td>
<td>Positive Big</td>
<td>Discard</td>
</tr>
<tr>
<td>21</td>
<td>Medium</td>
<td>Low</td>
<td>Negative Big</td>
<td>Strong Admit</td>
</tr>
<tr>
<td>22</td>
<td>Medium</td>
<td>Low</td>
<td>Negative</td>
<td>Admit</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
</tr>
<tr>
<td>23</td>
<td>Medium</td>
<td>Low</td>
<td>Small Zero</td>
<td>Admit</td>
</tr>
<tr>
<td>24</td>
<td>Medium</td>
<td>Low</td>
<td>Small Positive</td>
<td>Admit Marginal</td>
</tr>
<tr>
<td>25</td>
<td>Medium</td>
<td>Low</td>
<td>Positive Big Negative Big</td>
<td>Admit</td>
</tr>
<tr>
<td>26</td>
<td>Medium</td>
<td>Medium</td>
<td>Negative Big</td>
<td>Admit</td>
</tr>
<tr>
<td>27</td>
<td>Medium</td>
<td>Medium</td>
<td>Small</td>
<td>Admit Marginal</td>
</tr>
<tr>
<td>28</td>
<td>Medium</td>
<td>Medium</td>
<td>Zero Positive</td>
<td>Admit Marginal</td>
</tr>
<tr>
<td>29</td>
<td>Medium</td>
<td>Medium</td>
<td>Small</td>
<td>Admit Marginal</td>
</tr>
<tr>
<td>30</td>
<td>Medium</td>
<td>Medium</td>
<td>Positive Big</td>
<td>Admit</td>
</tr>
<tr>
<td>31</td>
<td>Medium</td>
<td>High</td>
<td>Negative Big Negative Big</td>
<td>Admit Marginal</td>
</tr>
<tr>
<td>32</td>
<td>Medium</td>
<td>High</td>
<td>Negative Big Negative</td>
<td>Admit Marginal</td>
</tr>
<tr>
<td>33</td>
<td>Medium</td>
<td>High</td>
<td>Zero Positive</td>
<td>Discard Marginal</td>
</tr>
<tr>
<td>34</td>
<td>Medium</td>
<td>High</td>
<td>Zero Small</td>
<td>Discard Marginal</td>
</tr>
<tr>
<td>35</td>
<td>Medium</td>
<td>High</td>
<td>Zero Positive Small</td>
<td>Discard Marginal</td>
</tr>
<tr>
<td>36</td>
<td>Medium</td>
<td>High</td>
<td>Very High Negative Big</td>
<td>Discard Marginal</td>
</tr>
<tr>
<td>37</td>
<td>Medium</td>
<td>High</td>
<td>Very High Negative Big Negative</td>
<td>Discard</td>
</tr>
<tr>
<td>38</td>
<td>Medium</td>
<td>High</td>
<td>Very High Zero Positive</td>
<td>Discard</td>
</tr>
<tr>
<td>39</td>
<td>Medium</td>
<td>High</td>
<td>Very High Zero Positive Small</td>
<td>Discard Strong</td>
</tr>
<tr>
<td>40</td>
<td>Medium</td>
<td>High</td>
<td>Low Positive Big Negative Big</td>
<td>Discard</td>
</tr>
<tr>
<td>41</td>
<td>High</td>
<td>Low</td>
<td>Small</td>
<td>Admit Marginal</td>
</tr>
<tr>
<td>42</td>
<td>High</td>
<td>Low</td>
<td>Small Zero</td>
<td>Admit Marginal</td>
</tr>
<tr>
<td>43</td>
<td>High</td>
<td>Low</td>
<td>Zero Positive</td>
<td>Admit Marginal</td>
</tr>
<tr>
<td>44</td>
<td>High</td>
<td>Low</td>
<td>Small</td>
<td>Admit Marginal</td>
</tr>
<tr>
<td>45</td>
<td>High</td>
<td>Low</td>
<td>Positive Big</td>
<td>Admit</td>
</tr>
<tr>
<td></td>
<td>46</td>
<td>High</td>
<td>Medium</td>
<td>Negative Big</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td>47</td>
<td>High</td>
<td>Medium</td>
<td>Negative Small</td>
</tr>
<tr>
<td></td>
<td>48</td>
<td>High</td>
<td>Medium</td>
<td>Zero Positive Small</td>
</tr>
<tr>
<td></td>
<td>49</td>
<td>High</td>
<td>Medium</td>
<td>Small Positive Small</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>High</td>
<td>Medium</td>
<td>Positive Big</td>
</tr>
<tr>
<td></td>
<td>51</td>
<td>High</td>
<td>High</td>
<td>Negative Big Negative</td>
</tr>
<tr>
<td></td>
<td>52</td>
<td>High</td>
<td>High</td>
<td>Small Small</td>
</tr>
<tr>
<td></td>
<td>53</td>
<td>High</td>
<td>High</td>
<td>Zero Positive</td>
</tr>
<tr>
<td></td>
<td>54</td>
<td>High</td>
<td>High</td>
<td>Small Small</td>
</tr>
<tr>
<td></td>
<td>55</td>
<td>High</td>
<td>High</td>
<td>Positive Big Small</td>
</tr>
<tr>
<td></td>
<td>56</td>
<td>High</td>
<td>High</td>
<td>Negative Big Negative</td>
</tr>
<tr>
<td></td>
<td>57</td>
<td>High</td>
<td>High</td>
<td>Negative Small Small</td>
</tr>
<tr>
<td></td>
<td>58</td>
<td>High</td>
<td>High</td>
<td>Zero Positive</td>
</tr>
<tr>
<td></td>
<td>59</td>
<td>High</td>
<td>High</td>
<td>Small Small</td>
</tr>
<tr>
<td></td>
<td>60</td>
<td>High</td>
<td>Very</td>
<td>High Positive Big</td>
</tr>
<tr>
<td></td>
<td>61</td>
<td>High</td>
<td>Very</td>
<td>Low Negative Big Negative</td>
</tr>
<tr>
<td></td>
<td>62</td>
<td>High</td>
<td>Very</td>
<td>Low Small</td>
</tr>
<tr>
<td></td>
<td>63</td>
<td>High</td>
<td>Very</td>
<td>Low Zero Positive</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>High</td>
<td>Very</td>
<td>Low Small</td>
</tr>
<tr>
<td></td>
<td>65</td>
<td>High</td>
<td>Very</td>
<td>Low Positive Big</td>
</tr>
<tr>
<td></td>
<td>66</td>
<td>High</td>
<td>Very</td>
<td>Medium Negative Big Negative</td>
</tr>
<tr>
<td></td>
<td>67</td>
<td>High</td>
<td>Very</td>
<td>Medium Small</td>
</tr>
<tr>
<td></td>
<td>68</td>
<td>High</td>
<td>Medium</td>
<td>Zero</td>
</tr>
<tr>
<td></td>
<td>69</td>
<td>Very</td>
<td>Medium</td>
<td>Positive</td>
</tr>
<tr>
<td></td>
<td>High</td>
<td>Very</td>
<td></td>
<td>Small</td>
</tr>
<tr>
<td>----</td>
<td>------</td>
<td>------</td>
<td>-----</td>
<td>-------</td>
</tr>
<tr>
<td>70</td>
<td>High</td>
<td>Very</td>
<td>Medium</td>
<td>Positive Big</td>
</tr>
<tr>
<td>71</td>
<td>High</td>
<td>Very</td>
<td>High</td>
<td>Negative Big</td>
</tr>
<tr>
<td>72</td>
<td>High</td>
<td>Very</td>
<td>High</td>
<td>Small</td>
</tr>
<tr>
<td>73</td>
<td>High</td>
<td>Very</td>
<td>High</td>
<td>Zero</td>
</tr>
<tr>
<td>74</td>
<td>High</td>
<td>Very</td>
<td>High</td>
<td>Small</td>
</tr>
<tr>
<td>75</td>
<td>High</td>
<td>Very</td>
<td>High</td>
<td>Positive Big</td>
</tr>
<tr>
<td>76</td>
<td>High</td>
<td>Very</td>
<td>High</td>
<td>Negative Big</td>
</tr>
<tr>
<td>77</td>
<td>High</td>
<td>Very</td>
<td>High</td>
<td>Small</td>
</tr>
<tr>
<td>78</td>
<td>High</td>
<td>Very</td>
<td>High</td>
<td>Zero</td>
</tr>
<tr>
<td>79</td>
<td>High</td>
<td>Very</td>
<td>High</td>
<td>Small</td>
</tr>
<tr>
<td>80</td>
<td>High</td>
<td>High</td>
<td>Positive Big</td>
<td>Discard</td>
</tr>
</tbody>
</table>
DATA STRUCTURES USED

Data Structures Used

// Devices with two interfaces
Node structure {
    Node ID   // Device ID
    In Buffer Pointer to QUEUE // Inbound buffer
    In Buffer Size (N) // Inbound buffer queue size
    Out Buffer Pointer to DEQUEUE // Outbound buffer
    Out Buffer Size (N) // Outbound buffer queue size
    Node Policy In Buffer Keep // QoS high priority and low priority keep packets
    Node Policy In Buffer Drop // QoS high priority and low priority drop packets
    Node Interface Path 1 // interface to left device within path
    Node Interface Path 2 // interface to right device within path
}

// Device Inbound buffer
In Buffer QUEUE structure {
    Pointer to QUEUE // Inbound buffer queue
}

// Device Outbound buffer
Out Buffer DEQUEUE structure {
    Pointer to DEQUEUE // Outbound buffer queue
}

// Network path made up of devices
Path structure {
    Path ID (Path 1 or Path 2) // Path X -> Y or Path X -> Z only
    Pointer to Node Linked List // Path of devices that packet is routed
}

// Network path made up of devices
Node Linked List structure {
}

// node = device
Node structure {
}

// packet definition
Packet structure {

Packet ID // packet identification
Priority Flag (high or low) // high priority or low priority packet
Packet Size (fixed or variable) // a fixed number packets = cell variable number packets = frame
   Pointer to Data Linked List // cell or frame made up of packets

// Data Stream/Flow definition
Data Linked List structure {
}

// cell or frame
Data structure {
}
UI SPECIFICATION

Windows Section

Simulation with no fuzzy logic controls exposed (Default Window)

Simulation Control Settings

Device Tab
- High Priority Traffic Flow Quality of Service Guaranteed Policy with range between 0 and 100
- Low Priority Traffic Flow Quality of Service Best Efforts Policy with range between 0 and 100
- Buffer Capacity in number of packets with range between 0 and 256
- Bandwidth in number of packets per second with range between 1 and 10000
- Fixed Sized Data in number of packets with range between 0 and 56
- Variable Sized Data in number of packets with range between 0 and 72
- Const Buffer Emptying is set
- First-in-first-out Buffering is set

Network Tab
- Low Traffic Intensity to Congestion as a percentage with range 0 to 100
Status

<table>
<thead>
<tr>
<th>Status=SIM Ready</th>
<th>Control=No Fuzzy</th>
<th>LSR A=No Fuzzy</th>
<th>LSR B=No Fuzzy</th>
<th>LSR C=No Fuzzy</th>
<th>LSR D=No Fuzzy</th>
<th>Path=Host X to Host Y</th>
</tr>
</thead>
</table>

Status = SIM Ready means that the simulator is ready to run.
Control = No Fuzzy means that the SIM is not in Fuzzy Logic Mode and no Fuzzy parameters have been applied.
LSR A = Fuzzy means that device LSR A is disabled with Fuzzy Logic control scheme.
LSR B = Fuzzy means that device LSR A is disabled with Fuzzy Logic control scheme.
LSR C = Fuzzy means that device LSR A is disabled with Fuzzy Logic control scheme.
LSR D = Fuzzy means that device LSR A is disabled with Fuzzy Logic control scheme.
Path = Host X to Host Y means that the path is set between Host X to LSR A to LSR B to LSR D to Host Y.

Simulation with fuzzy logic controls exposed

Simulation Control Settings

Action Tab – Output variable made up of size fuzzy sets
  • Membership functions for the Action output variable
- X value with range between 0 and 1 for each of the fuzzy sets based on non-fuzzy output from membership function
- Y value with range between 0 and 1 for each of the fuzzy sets based on non-fuzzy output from membership function

N Tab – Input variable
- Number of packets in buffer at any point in time (N input variable) with range between 0 and 256

Delta N Tab – Input variable
- Number of packets in buffer over time period (delta N) with range between 0 and 256

CLR Tab – Input variable
- High priority loss ratio in percentage with range between 0 and 100

Status

<table>
<thead>
<tr>
<th>Status</th>
<th>Control</th>
<th>LSR A</th>
<th>LSR B</th>
<th>LSR C</th>
<th>LSR D</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIM Ready</td>
<td>Fuzzy</td>
<td>Fuzzy</td>
<td>Fuzzy</td>
<td>Fuzzy</td>
<td>Fuzzy</td>
<td>Host X to Host Y</td>
</tr>
</tbody>
</table>

Status = SIM Ready means that the simulator is ready to run
Control = Fuzzy means that the SIM is in Fuzzy Logic Mode with all Fuzzy parameters applied
LSR A = Fuzzy means that device LSR A is enabled with Fuzzy Logic control scheme
LSR B = Fuzzy means that device LSR A is enabled with Fuzzy Logic control scheme
LSR C = Fuzzy means that device LSR A is enabled with Fuzzy Logic control scheme
LSR D = Fuzzy means that device LSR A is enabled with Fuzzy Logic control scheme
Path = Host X to Host Y means that the path is set between Host X to LSR A to LSR B to LSR D to Host Y

Devices

- LSR A – router device with two interfaces and is the ingress router to the network
- LSR B – router device with three interfaces and is the intermediate router within the network
LSR C – router device with two interfaces and is the egress router to the network
LSR D – router device with two interfaces and is the egress router to the network

Device Settings for LSR A, LSR B, LSR C and LSR D

Device Name: LSR A
Device Policy: Admit More Packets to avoid waste
Device Buffer Capacity: 128 packets

Device Buffer Input Low-Priority Packets
Accept Count: 0
Reject Count: 0

Main Menus Section

File Menu

Preferences…
Exit

Preferences – will display a dialog that allows the user to configure window settings such as setting the output (also called the results) window after each time a simulation run
Exit – allows the user to quit

View Menu

<table>
<thead>
<tr>
<th>Fuzzy Logic Controls</th>
</tr>
</thead>
<tbody>
<tr>
<td>FL Fuzzy Sets...</td>
</tr>
<tr>
<td>FL Priority Control Scheme...</td>
</tr>
</tbody>
</table>

Fuzzy Logic Controls – will display all of the fuzzy controls in the Control region

FL Fuzzy Sets – will display a dialog that contains the six output fuzzy sets

FL Priority Control Scheme – will display a dialog that allows the user to select one or more rules applied out of the possible eighty fuzzy logic rules available when running the simulator in fuzzy logic mode

Simulator Menu

<table>
<thead>
<tr>
<th>Run</th>
<th>F5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step</td>
<td>F11</td>
</tr>
<tr>
<td>Trials...</td>
<td></td>
</tr>
</tbody>
</table>

MPLS Route
Network Devices
Hosts

<table>
<thead>
<tr>
<th>Reset</th>
<th>Ctrl+F5</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIM Results...</td>
<td></td>
</tr>
</tbody>
</table>

Run – will activate and run the simulation

Step – will activate and step through the simulation one step at a time

Trials…

MPLS Route – will display a submenu to allow the user to select a particular path where there are three paths available
Network Devices – will display a submenu to allow the user to select a particular device in order to display device configuration information (note: clicking on the device when the device selection region of the window is visible is also an option).

Hosts – will display a submenu to allow the user to select a particular host in order to display host configuration information.

Reset – will set the simulation back to the beginning
SIM Results… -- by default is disabled but is enabled once a simulation has run at which point a user can select this menu item to display the results of the last simulation

**Window Menu**

- Simulation Screen
- Simulation Controls
- Device Selection

Simulation Screen – will show/hide the primary screen that contains the simulation model

Simulation Controls – will show/hide the secondary screen that contains the simulation control settings

Device Selection – will show/hide all four device mouse roll-overs (or hot regions) that allow the user to quickly access the device settings dialog

**Help Menu**

About MyGradProj SIM…

About – will display a dialog containing version reference

**Screen Regions**

Simulation Region
Simulation Screen Properties

Host X:
Outbound buffer has transferred 0 packets
Progress bar is clear

LSR A: (MPLS Enabled - Label Switch router A)
Inbound Buffer currently has 0 packets
Outbound Buffer currently has 0 packets
Progress bar is clear

LSR B: (MPLS Enabled - Label Switch router B)
Inbound Buffer currently has 0 packets
Outbound Buffer currently has 0 packets
Progress bar is clear
LSR C: (MPLS Enabled - Label Switch router C)
Inbound Buffer currently has 0 packets
Outbound Buffer currently has 0 packets
Progress bar is clear

LSR D: (MPLS Enabled - Label Switch router D)
Inbound Buffer currently has 0 packets
Outbound Buffer currently has 0 packets
Progress bar is clear

Host Y:
Inbound buffer currently has 56 packets
Progress bar is clear

Host Z:
Inbound buffer currently has 0 packets
Progress bar is clear

Controls Region – simulation settings for the model
Simulation Control Settings

Device Tab

- High Priority Traffic Flow Quality of Service Guaranteed Policy with range between 0 and 100
- Low Priority Traffic Flow Quality of Service Best Efforts Policy with range between 0 and 100
- Buffer Capacity in number of packets with range between 0 and 256
- Bandwidth in number of packets per second with range between 1 and 10000
- Fixed Sized Data in number of packets with range between 0 and 56
- Variable Sized Data in number of packets with range between 0 and 72
- Const Buffer Emptying is set
- First-in-first-out Buffering is set
Network Tab

- Low Traffic Intensity to Congestion as a percentage with range 0 to 100

Controls Region – fuzzy logic settings for the model

ACTION – fuzzy logic output panel

Action Tab – Output variable made up of size fuzzy sets

- Membership functions for the Action output variable
- SD = Strong Discard, D = Discard, MD = Marginal Discard, MA = Marginal Admit, A = Admit, SA = Strong Admit
• X value with range between 0 and 1 for each of the fuzzy sets based on non fuzzy output from membership function
• Y value with range between 0 and 1 for each of the fuzzy sets based on non fuzzy output from membership function

N – fuzzy logic input variable panel

N Tab – Input variable
• Number of packets in buffer at any point in time (N input variable) with range between 0 and 256

Delta N – fuzzy logic input variable panel
Delta N Tab – Input variable
- Number of packets in buffer over time period (delta N) with range between 0 and 256

CLR – fuzzy logic input variable panel

CLR Tab – Input variable
- High priority loss ratio in percentage with range between 0 and 100
Device Region – four mouse-over regions that allow the user to quickly get device information

LSR A – router device with two interfaces and is the ingress router to the network
LSR B – router device with three interfaces and is the intermediate router within the network
LSR C – router device with two interfaces and is the egress router to the network
LSR D – router device with two interfaces and is the egress router to the network

Status Region – status on devices, hosts, controls and errors

| Status: SIM Ready | Control: No Fuzzy | LSR A: No Fuzzy | LSR B: No Fuzzy | LSR C: No Fuzzy | LSR D: No Fuzzy | Path: Host X to Host Y |

Status = SIM Ready means that the simulator is ready to run
Control = No Fuzzy means that the SIM is not in Fuzzy Logic Mode and no Fuzzy parameters have been applied

LSR A = Fuzzy means that device LSR A is disabled with Fuzzy Logic control scheme
LSR B = Fuzzy means that device LSR A is disabled with Fuzzy Logic control scheme
LSR C = Fuzzy means that device LSR A is disabled with Fuzzy Logic control scheme
LSR D = Fuzzy means that device LSR A is disabled with Fuzzy Logic control scheme
Path = Host X to Host Y means that the path is set between Host X to LSR A to LSR B to LSR D to Host Y

Dialogs

Preferences Dialog – Default Settings to show output (also called results) window after each simulation run
Fuzzy Sets Dialog – six fuzzy sets used in the model

- Strong Discard (SD)
- Discard (D)
- Marginal Discard (MD)
- Marginal Admit (MA)
- Admit (A)
- Strong Admit (SA)
Control Scheme – eighty fuzzy IF-THEN rules used in the model

![Control Scheme]

Device Settings – user can change device policy and get read out of buffer input accept/reject rates

![Device Settings]
Host Settings – user can get read out of amount of data flow either sent out by the host or received by the host depending on the host selected

![Host Settings window]

About – copyright and version information

![About window]

MyGradProj SIM

Created By Eric W. Yocam

May 1, 2004

Pre Release Version 0.2a
MAX-MIN INFERENCE AND DEFUZZIFICATION

EQUATIONS BASED ON COG APPROACH

(FROM [35])

| I |               |    |              |    |               | O |
| N | Crisp-to-Fuzzy |    | Inference    |    | Fuzzy-to-Crisp | U |
| P | |---|           |---|           | T |
| U | FUZZIFY      | max-min | DEFUZZIFY    | P |
| T |_______________|___________|_______________| U |

Membership Functions

Rule Base

Center of Gravity

Center of Gravity (COG) approach is equivalent to Centroid of Area

**Defuzzification**

Conversion of the fuzzy result of inference into a crisp output variable U

**Center of Gravity (CoG) Method**

The crisp output variable is determined as an abscissa value U of center of gravity under the membership function:

- example for center of gravity method

\[
U = \operatorname{Max} \int u \cdot \mu(u) \, du - \operatorname{Min} \int \mu(u) \, du
\]

\( U : \text{Center of Gravity (COG)} \)
APPENDIX E
MINIMUM FCL COMPLIANCE AND A
LISTING OF PRODUCTION RULES
(FROM [35]).

Minimum FCL Compliance

<table>
<thead>
<tr>
<th>Language Element</th>
<th>Keyword</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>function block declaration</td>
<td>VAR_INPUT, VAR_OUTPUT</td>
<td>contains input and output variables</td>
</tr>
<tr>
<td>membership function</td>
<td>input variable: TERM</td>
<td>maximum of three points (degree of membership coordinate = 0 or 1)</td>
</tr>
<tr>
<td></td>
<td>output variable: TERM</td>
<td>only singletons</td>
</tr>
<tr>
<td>conditional aggregation</td>
<td>operator: AND</td>
<td>algorithm: MIN</td>
</tr>
<tr>
<td>activation</td>
<td></td>
<td>Not relevant because singletons are used only</td>
</tr>
<tr>
<td>accumulation (result aggregation)</td>
<td>operator: ACCU</td>
<td>algorithm: MAX</td>
</tr>
<tr>
<td>defuzzification</td>
<td>METHOD</td>
<td>algorithm: COGS</td>
</tr>
<tr>
<td>condition</td>
<td>IF ... IS ...</td>
<td>n subconditions</td>
</tr>
<tr>
<td>conclusion</td>
<td>THEN</td>
<td>only one subconclusion</td>
</tr>
<tr>
<td>weighting factor</td>
<td>WITH</td>
<td>value only</td>
</tr>
</tbody>
</table>

Production Rules

function_block_declaration ::= 'FUNCTION_BLOCK``
function_block_name

{fb_io_var_declarations}

{other_var_declarations}

function_block_body

'END_FUNCTION_BLOCK'
fb_io_var_declarations ::= input_declarations | output_declarations
other_var_declarations ::= var_declarations
function_block_body ::= {fuzzify_block}
  {defuzzify_block}
  {rule_block}
  {option_block}

fuzzify_block ::= 'FUZZIFY' variable_name
  {linguistic_term}
  'END_FUZZIFY'
defuzzify_block ::= 'DEFUZZIFY' f_variable_name
  {linguistic_term}
  defuzzification_method
  default_value
  [range]
  'END_FUZZIFY'
rule_block ::= 'RULEBLOCK' rule_block_name
  operator_definition
  [activation_method]
  accumulation_method
  {rule}
  'END_RULEBLOCK'
option_block ::= 'OPTION'
  any manufacturere specific parameter
  'END_OPTION'
linguistic_term ::= 'TERM' term_name ':= membership_function ';'membership_function ::= singleton | points
singleton ::= numeric_literal | variable_name
points ::= '{(' numeric_literal | variable_name ','
  numeric_literal ')}'
defuzzification_method ::= 'METHOD' ':' 'COG' | 'COGS' | 'COA' | 'LM' | 'RM' | 'MOM';'
default_value ::= 'DEFAULT' ':= numeric_literal | 'NC' ';'
range ::= 'RANGE' ':= '('numeric_literal '..' numeric_literal')''operator_definition ::= ('OR' ':' 'MAX' | 'ASUM' | 'BSUM') |
  ('AND' ':' 'MIN' | 'PROD' | 'BDIF') ';
activation_method ::= 'ACT' ':' 'PROD' | 'MIN' ';
accumulation_method ::= 'ACCU' ':' 'MAX' | 'BSUM' | 'NSUM' ';
rule ::= 'RULE' integer_literal ':
  'IF' condition 'THEN' conclusion [WITH
  weighting_factor '];'
condition ::= (subcondition | variable_name) (AND' | 'OR'
  (subcondition | variable_name))
subcondition ::= ('NOT' '(' variable_name 'IS' ['NOT']
  term_name ')') | ( variable_name 'IS' ['NOT'] term_name )
FLLL shorthand: subcondition ::= term_name

conclusion ::= { (variable_name | (variable_name 'IS' term_name))
  ')
weighting_factor ::= variable | numeric_literal
function_block_name ::= identifier
rule_block_name ::= identifier
term_name ::= identifier
variable_name ::= identifier
numeric_literal ::= integer_literal | real_literal
letter ::= 'A' | 'B' | '<..' | 'Z' | 'a' | 'b' | '<..' | 'z'
digit ::= '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'
identifier ::= (letter | ('_' (letter | digit))) {{'_'} (letter | digit)}
input_declarations ::= 'VAR_INPUT' ['RETAIN' | 'NON_RETAIN']
input_declaration ';'
{input_declaration ';'}
'END_VAR'
input_declaration ::= var_init_decl | edge_declaration

var_init_decl ::= var1_init_decl | array_var_init_decl | structured_var_init_decl | fb_name_decl | string_var_declaration

var1_init_decl ::= var1_list ':'
(simple_spec_init | subrange_spec_init | enumerated_spec_init)
var1_list ::= variable_name {',' variable_name}
array_var_init_decl ::= var1_list ':' array_spec_init
output_declarations ::= 'VAR_OUTPUT' ['RETAIN' | 'NON_RETAIN']
var_init_decl ::= var1_list ':'
{var_init_decl ';'}
'END_VAR'

real_type_name ::= 'REAL' | 'LREAL'
numeric_type_name ::= integer_type_name | real_type_name
elementary_type_name ::= numeric_type_name | date_type_name |
bit_string_type_name | 'STRING' | 'WSTRING' | 'TIME'
simple_type_name ::= identifier
simple_type_declaration ::= simple_type_name ':' simple_spec_init
simple_specification ::= elementary_type_name | simple_type_name
simple_spec_init ::= simple_specification [':=' constant]
FUNCTION_BLOCK

VAR_INPUT
  N_IN  REAL; (* RANGE(0 .. 100) *)
  DN_IN     REAL; (* RANGE(0 .. 100) *)
  CLR_IN  REAL; (* RANGE(0 .. 100) *)
END_VAR

VAR_OUTPUT
  ACTION  REAL; (* RANGE(0 .. 7) *)
END_VAR

FUZZIFY N_IN
  TERM Low := (0, 0) (0, 1) (50, 0) ;
  TERM Medium := (14, 0) (50, 1) (83, 0) ;
  TERM High := (50, 0) (100, 1) (100, 0) ;
  TERM Very_High := (50, 0) (100, 1) (100, 0) ;
END FUZZIFY

FUZZIFY DN_IN
  TERM Negative_Big := (0, 0) (0, 1) (50, 0) ;
  TERM Negative_Small := (14, 0) (50, 1) (83, 0) ;
  TERM Zero := (50, 0) (100, 1) (100, 0) ;
  TERM Positive_Small := (50, 0) (100, 1) (100, 0) ;
  TERM Positive_Big := (50, 0) (100, 1) (100, 0) ;
END FUZZIFY

FUZZIFY CLR_IN
  TERM Low := (0, 0) (0, 1) (50, 0) ;
  TERM Medium := (14, 0) (50, 1) (80, 0) ;
  TERM High := (50, 0) (100, 1) (100, 0) ;
  TERM Very_High := (50, 0) (100, 1) (100, 0) ;
END FUZZIFY

FUZZIFY ACTION
  TERM Strong_Discard :=  1;
  TERM Discard :=  2;
  TERM Marginal_Discard :=  3;
  TERM Marginal_Admit :=  4;
  TERM Admit :=  5;
  TERM Strong_Admit :=  6;
END FUZZIFY

DEFUZZIFY ACTION
  METHOD: MoM;
END DEFUZZIFY
RULEBLOCK first
   AND:MIN;
   ACCUM:MAX;
   RULE 0: IF (CLR IS Low) AND (N IS Low) AND (DN IS Negative_Big)
   THEN (ACTION IS Strong_Admit);
   RULE 1: IF (CLR IS Low) AND (N IS Low) AND (DN IS Negative_Small)
   THEN (ACTION IS Strong_Admit);
   RULE 2: IF (CLR IS Low) AND (N IS Low) AND (DN IS Zero) THEN
   (ACTION IS Strong_Admit);
   RULE 3: IF (CLR IS Low) AND (N IS Low) AND (DN IS Positive_Small)
   THEN (ACTION IS Strong_Admit);
   RULE 4: IF (CLR IS Low) AND (N IS Low) AND (DN IS Positive_Big)
   THEN (ACTION IS Strong_Admit);
   RULE 5: IF (CLR IS Low) AND (N IS Medium) AND (DN IS
   Negative_Big) THEN (ACTION IS Strong_Admit);
   RULE 6: IF (CLR IS Low) AND (N IS Medium) AND (DN IS Neutral)
   THEN (ACTION IS Admit);
   RULE 7: IF (CLR IS Low) AND (N IS Medium) AND (DN IS Neutral)
   THEN (ACTION IS Marginal_Admit);
   RULE 8: IF (CLR IS Low) AND (N IS Medium) AND (DN IS Neutral)
   THEN (ACTION IS Marginal_Admit);
   RULE 9: IF (CLR IS Low) AND (N IS Medium) AND (DN IS Neutral)
   THEN (ACTION IS Marginal_Admit);
   RULE 10: IF (CLR IS Low) AND (N IS Medium) AND (DN IS Neutral)
   THEN (ACTION IS Marginal_Admit);
   RULE 11: IF (CLR IS Low) AND (N IS Medium) AND (DN IS Neutral)
   THEN (ACTION IS Marginal_Admit);
   RULE 12: IF (CLR IS Low) AND (N IS Medium) AND (DN IS Neutral)
   THEN (ACTION IS Marginal_Admit);
   RULE 13: IF (CLR IS Low) AND (N IS Medium) AND (DN IS Neutral)
   THEN (ACTION IS Marginal_Admit);
   RULE 14: IF (CLR IS Low) AND (N IS Medium) AND (DN IS Neutral)
   THEN (ACTION IS Marginal_Admit);
   RULE 15: IF (CLR IS Low) AND (N IS Very_High) AND (DN IS
   Negative_Big) THEN (ACTION IS Strong_Admit);
   RULE 16: IF (CLR IS Low) AND (N IS Very_High) AND (DN IS
   Negative_Small) THEN (ACTION IS Strong_Admit);
   RULE 17: IF (CLR IS Low) AND (N IS Very_High) AND (DN IS
   Negative_Small) THEN (ACTION IS Strong_Admit);
   RULE 18: IF (CLR IS Low) AND (N IS Very_High) AND (DN IS
   Positive_Small) THEN (ACTION IS Discard);
   RULE 19: IF (CLR IS Low) AND (N IS Very_High) AND (DN IS
   Positive_Big) THEN (ACTION IS Discard);
   RULE 20: IF (CLR IS Medium) AND (N IS Low) AND (DN IS
   Negative_Big) THEN (ACTION IS Strong_Admit);
   RULE 21: IF (CLR IS Medium) AND (N IS Low) AND (DN IS
   Negative_Big) THEN (ACTION IS Strong_Admit);
   RULE 22: IF (CLR IS Medium) AND (N IS Low) AND (DN IS
   Zero) THEN (ACTION IS Strong_Admit);
   RULE 23: IF (CLR IS Medium) AND (N IS Low) AND (DN IS
   Positive_Small) THEN (ACTION IS Admit);
   RULE 24: IF (CLR IS Medium) AND (N IS Low) AND (DN IS
   Positive_Big) THEN (ACTION IS Marginal_Admit);
RULE 25: IF (CLR IS Medium) AND (N IS Medium) AND (DN IS Negative_Big) THEN (ACTION IS Admit);
RULE 26: IF (CLR IS Medium) AND (N IS Medium) AND (DN IS Negative_Small) THEN (ACTION IS Admit);
RULE 27: IF (CLR IS Medium) AND (N IS Medium) AND (DN IS Zero) THEN (ACTION IS Marginal_Admit);
RULE 28: IF (CLR IS Medium) AND (N IS Medium) AND (DN IS Positive_Small) THEN (ACTION IS Marginal_Admit);
RULE 29: IF (CLR IS Medium) AND (N IS Medium) AND (DN IS Positive_Big) THEN (ACTION IS Marginal_Admit);
RULE 30: IF (CLR IS Medium) AND (N IS High) AND (DN IS Negative_Big) THEN (ACTION IS Marginal_Admit);
RULE 31: IF (CLR IS Medium) AND (N IS High) AND (DN IS Negative_Small) THEN (ACTION IS Marginal_Admit);
RULE 32: IF (CLR IS Medium) AND (N IS High) AND (DN IS Zero) THEN (ACTION IS Marginal_Admit);
RULE 33: IF (CLR IS Medium) AND (N IS High) AND (DN IS Positive_Small) THEN (ACTION IS Marginal_Admit);
RULE 34: IF (CLR IS Medium) AND (N IS High) AND (DN IS Positive_Big) THEN (ACTION IS Marginal_Admit);
RULE 35: IF (CLR IS Medium) AND (N IS Very_High) AND (DN IS Negative_Big) THEN (ACTION IS Marginal_Admit);
RULE 36: IF (CLR IS Medium) AND (N IS Very_High) AND (DN IS Negative_Small) THEN (ACTION IS Marginal_Admit);
RULE 37: IF (CLR IS Medium) AND (N IS Very_High) AND (DN IS Zero) THEN (ACTION IS Discard);
RULE 38: IF (CLR IS Medium) AND (N IS Very_High) AND (DN IS Positive_Small) THEN (ACTION IS Discard);
RULE 39: IF (CLR IS High) AND (N IS Low) AND (DN IS Negative_Big) THEN (ACTION IS Admit);
RULE 40: IF (CLR IS High) AND (N IS Low) AND (DN IS Negative_Small) THEN (ACTION IS Admit);
RULE 41: IF (CLR IS High) AND (N IS Low) AND (DN IS Zero) THEN (ACTION IS Marginal_Admit);
RULE 42: IF (CLR IS High) AND (N IS Low) AND (DN IS Positive_Small) THEN (ACTION IS Marginal_Admit);
RULE 43: IF (CLR IS High) AND (N IS Low) AND (DN IS Positive_Big) THEN (ACTION IS Discard);
RULE 44: IF (CLR IS High) AND (N IS Medium) AND (DN IS Negative_Big) THEN (ACTION IS Marginal_Admit);
RULE 45: IF (CLR IS High) AND (N IS Medium) AND (DN IS Negative_Small) THEN (ACTION IS Marginal_Admit);
RULE 46: IF (CLR IS High) AND (N IS Medium) AND (DN IS Zero) THEN (ACTION IS Marginal_Admit);
RULE 47: IF (CLR IS High) AND (N IS Medium) AND (DN IS Positive_Big) THEN (ACTION IS Discard);
RULE 48: IF (CLR IS High) AND (N IS Medium) AND (DN IS Positive_Small) THEN (ACTION IS Discard);
RULE 49: IF (CLR IS High) AND (N IS Medium) AND (DN IS Negative_Small) THEN (ACTION IS Discard);
RULE 50: IF (CLR IS High) AND (N IS High) AND (DN IS Negative_Big) THEN (ACTION IS Marginal_Admit);
RULE 51: IF (CLR IS High) AND (N IS High) AND (DN IS Negative_Small) THEN (ACTION IS Discard);
RULE 52: IF (CLR IS High) AND (N IS High) AND (DN IS Zero) THEN (ACTION IS Discard);
RULE 53: IF (CLR IS High) AND (N IS High) AND (DN IS Positive_Small) THEN (ACTION IS Discard);
RULE 54: IF (CLR IS High) AND (N IS High) AND (DN IS Positive_Big) THEN (ACTION IS Discard);
RULE 55: IF (CLR IS High) AND (N IS Very_High) AND (DN IS Negative_Big) THEN (ACTION IS Marginal_Discard);
RULE 56: IF (CLR IS High) AND (N IS Very_High) AND (DN IS Negative_Small) THEN (ACTION IS Discard);
RULE 57: IF (CLR IS High) AND (N IS Very_High) AND (DN IS Zero) THEN (ACTION IS Discard);
RULE 58: IF (CLR IS High) AND (N IS Very_High) AND (DN IS Positive_Small) THEN (ACTION IS Discard);
RULE 59: IF (CLR IS High) AND (N IS Very_High) AND (DN IS Positive_Big) THEN (ACTION IS Discard);
RULE 60: IF (CLR IS Very_High) AND (N IS Low) AND (DN IS Negative_Big) THEN (ACTION IS Marginal_Admit);
RULE 61: IF (CLR IS Very_High) AND (N IS Low) AND (DN IS Negative_Small) THEN (ACTION IS Marginal_Admit);
RULE 62: IF (CLR IS Very_High) AND (N IS Low) AND (DN IS Zero) THEN (ACTION IS Marginal_Admit);
RULE 63: IF (CLR IS Very_High) AND (N IS Low) AND (DN IS Positive_Small) THEN (ACTION IS Marginal_Admit);
RULE 64: IF (CLR IS Very_High) AND (N IS Low) AND (DN IS Positive_Big) THEN (ACTION IS Marginal_Admit);
RULE 65: IF (CLR IS Very_High) AND (N IS Medium) AND (DN IS Negative_Big) THEN (ACTION IS Discard);
RULE 66: IF (CLR IS Very_High) AND (N IS Medium) AND (DN IS Negative_Small) THEN (ACTION IS Discard);
RULE 67: IF (CLR IS Very_High) AND (N IS Medium) AND (DN IS Zero) THEN (ACTION IS Discard);
RULE 68: IF (CLR IS Very_High) AND (N IS Medium) AND (DN IS Positive_Small) THEN (ACTION IS Discard);
RULE 69: IF (CLR IS Very_High) AND (N IS Medium) AND (DN IS Positive_Big) THEN (ACTION IS Discard);
RULE 70: IF (CLR IS Very_High) AND (N IS High) AND (DN IS Negative_Big) THEN (ACTION IS Discard);
RULE 71: IF (CLR IS Very_High) AND (N IS High) AND (DN IS Negative_Small) THEN (ACTION IS Discard);
RULE 72: IF (CLR IS Very_High) AND (N IS High) AND (DN IS Zero) THEN (ACTION IS Discard);
RULE 73: IF (CLR IS Very_High) AND (N IS High) AND (DN IS Positive_Small) THEN (ACTION IS Discard);
RULE 74: IF (CLR IS Very_High) AND (N IS High) AND (DN IS Positive_Big) THEN (ACTION IS Discard);
RULE 75: IF (CLR IS Very_High) AND (N IS Very_High) AND (DN IS Negative_Big) THEN (ACTION IS Discard);
RULE 76: IF (CLR IS Very_High) AND (N IS Very_High) AND (DN IS Negative_Small) THEN (ACTION IS Discard);
RULE 77: IF (CLR IS Very_High) AND (N IS Very_High) AND (DN IS Zero) THEN (ACTION IS Discard);
RULE 78: IF (CLR IS Very_High) AND (N IS Very_High) AND (DN IS Positive_Small) THEN (ACTION IS Discard);
RULE 79: IF (CLR IS Very_High) AND (N IS Very_High) AND (DN IS Positive_Big) THEN (ACTION IS Strong_Discard);
END_RULEBLOCK

END_FUNCTION_BLOCK