1-1-2004

Study Of Resilient Packet Ring Based On OPNET

Xin Zhang
Ryerson University

TK
5105.88
.Z53
2004

STUDY OF RESILIENT PACKET RING BASED ON OPNET

by

Xin Zhang

BSc, Beijing United University, Beijing, P.R.China, July 1993

A thesis

presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Master of Applied Science

in the Program of

Electrical and Computer Engineering

Toronto, Ontario, Canada, 2004

AUTHOR'S DECLARATION

I hereby declare that I am the sole author of this thesis.

I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorized Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

BORROWER'S PAGE

ABSTRACT

Study of Resilient Packet Ring Based on OPNET
Master of Applied Science, 2004
Xin Zhang
Department of Electrical and Computer Engineering
Ryerson University

Resilient Packet Ring (RPR) is the next generation layer-2 protocol optimized for transporting data traffic rather than circuit-based traffic. In this thesis, we design and evaluate our own RPR simulation model that is fully compliant with the latest proposal promoted by IEEE 802.17 Work Group.

By using this model, we investigate the limitations of the fairness control algorithms proposed by IEEE 802.17 WG. An alternative design, namely, Fuzzy Logic Control, is considered to overcome the shortcomings. Real world scenarios are simulated using this new approach. The simulation results justify the application of this new RPR model, and support its validity.

Furthermore, by using this model we also derived an equation to calculate Fairness Round Trip Time (FRTT), which is a key parameter in designing an appropriated size for Secondary Transit Queue (STQ) in RPR. This equation overcomes the limitations proposed by IEEE 802.17 Work Group.

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Prof. N. W. Ma, for his continuous intellectual support. His teaching on different protocols has immensely helped me in converting IEEE 802.17 proposal into the computer code.

I would like to thank my co-supervisor, Prof. K. Raahemifar, for his encouragement and moral support. His teaching on OPNET has greatly facilitated the new approach presented in this thesis.

I thank their efforts on helping me turning this project from utopian scheme into reality.

I also thank Hossein Ghandehari, whose endless encouragement on the project made the model finally complete. Furthermore, Hossein first proposed to combine Fuzzy Logic Control (FLC) theory with congestion detection and use it as an alternative to the IEEE 802.17 proposed algorithms. Without his initiation, there won't be FLC algorithm in this model.

I would like to thank Gary Yip for his participation in this project. His help was an integral part of this project.

Arseny Taranenko, System Administrator in Network Research Lab, at Ryerson University, was a great facilitator in providing technical helps on OPNET.

DEDICATION

This thesis is dedicated to my wife, and my parents. Their support in this thesis was without doubt instrumental.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF ILLUSTRATIONS

Chapter 1: Introduction

The advent of the Internet ushered in an enormous increase in data traffic across networks. Since 1995, the Internet traffic has doubled every six months or one year, due to an increase in the number of users and the growth of per capita usage. This ratio of growth is expected to continue over the next few years, with the Internet traffic growing to 280 Tbps by the year 2005. The optical fiber seems to play a major role as transmission medium in Metropolitan Area Network (MAN) and Wide Area Network (WAN) to facilitate the additional bandwidth and to adapt the growth.

The advances in optical transmission technology have occurred at a rapid rate. Synchronous Optical Network (SONET) in North America and Synchronous Digital Hierarchy (SDH) in Europe have dominated the backbone of the telephone networks as the only standards for fiber optic digital transmission systems. However, with the blooming of Internet, transporting data traffic takes a significant role in communication networks. Because SONET was originally designed for point-to-point, circuit-switched applications (e.g. voice traffic), and there are many well-known disadvantages of using SONET for transmitting packet-based data traffic, for example, inefficiently managed shared resources, waste of protection bandwidth, excess copies of multicast packets, etc., these made SONET not an ideal solution for scaling metro network to meet the demand.
Since SONET/SDH is not optimized to transport packet-based data traffic, Resilient Packet Ring (RPR) offers a solution to integrate multiple services, instead of having data, voice and video delivered over separate parallel networks, and has become a hot topic among the industry and research and educational institutions.

Resilient Packet Ring is an emerging network architecture and technology designed to meet the requirements of packet-based metropolitan area networks. By creating a Media Access Control (MAC) layer protocol for ring networks, RPR attempts to find a fundamental solution to solve the metro bottleneck problems. This new technology is developed to take advantage of existing SONET ring architecture. It allows full capacity utilization of bandwidth by statistically multiplexing data. Also, it provides features such as spatial reuse, fairness access in packet transport and intelligent ring protection.

1.1 An Overview of Resilient Packet Ring

1.1.1 Topology of RPR

Resilient Packet Ring uses a bi-directional dual counter-rotating ring topology. The rings are referred to as "Inner Ring" and "Outer Ring". Both rings are concurrently utilized for transporting data and RPR control packets, thus fully utilizing the total available bandwidth of the rings. Data and control packets flow in the opposite direction, for instance, outer ring traffic-control information is transported in the inner ring to the upstream node and inner ring traffic-control information is transported in the outer ring to the upstream node. RPR control packets handle tasks such as topology discovery, protection switching and bandwidth control, etc. Figure 1.1 shows a RPR ring with four nodes. As one can see the data packet and control packet are flowing in opposite direction.

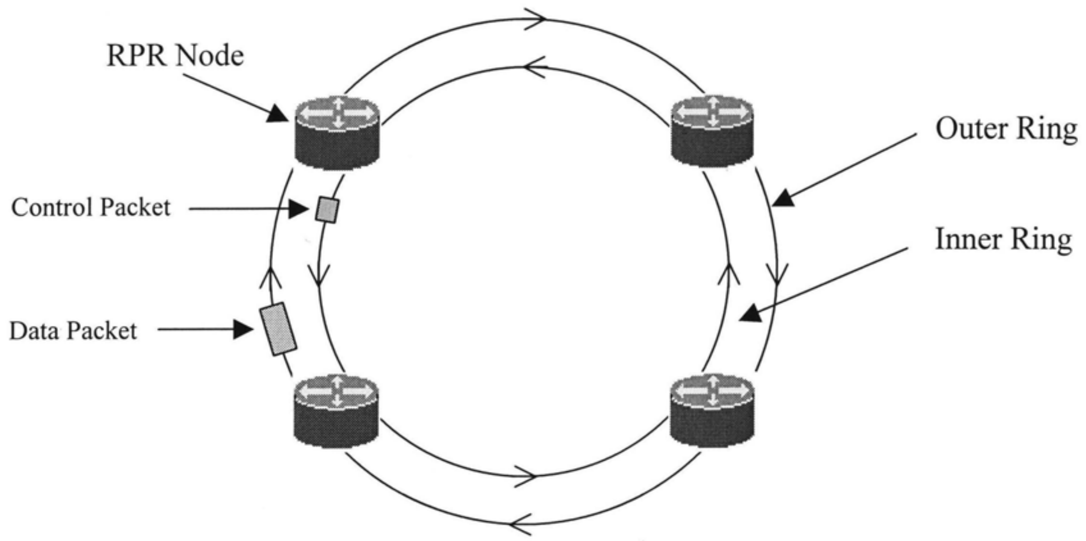


Figure 1.1: Resilient Packet Ring

1.1.2 Resiliency in RPR

Packet Rings have a natural resiliency advantage [8]. Ring fail-over is often described as "self-healing" or "automatic recovery." In practice, ring-based transport systems have reliably achieved less than 50ms fail-over periods. A Packet Ring protocol can initiate a "ring wrap" at the nodes surrounding the cut (see Figure 1.2) or packet "steering" by causing the sending node to redirect packets. In either case traffic can reach the original destination by going around the ring in the opposite direction in the event of a fiber cut.

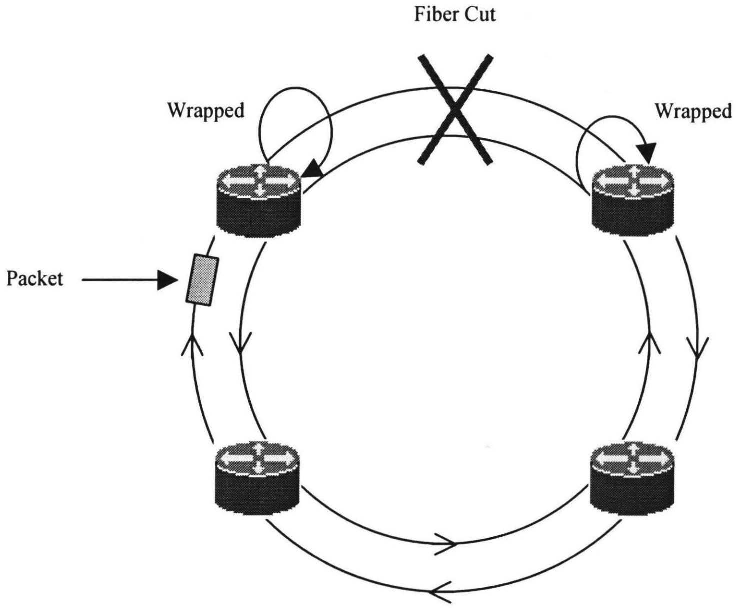


Figure 1.2: Recovery From a Fiber Cut

1.1.3 Bandwidth Fairness in RPR

Packet Rings have an inherent advantage for implementing fairness algorithms to regulate bandwidth usage. Ring bandwidth is a shared resource, and is vulnerable to exploitation by individual users or nodes. A fairness algorithm is a mechanism that gives every customer on the ring a "fair" share of the ring bandwidth, ideally without the straitjacket of a provisioned circuit. A ring-level fairness algorithm can and should allocate ring bandwidth as one global resource.

Bandwidth policies that can allow maximum ring bandwidth to be utilized between any two nodes when there is no congestion can be implemented without the inflexibility of a fixed circuit-based system like SONET, but with greater effectiveness than point-to-point Ethernet. SONET also implements point-to-point circuits that allocate and reserve a fixed amount of bandwidth for each connection, but lack of flexibility is the problem. Adding or subtracting bandwidth requires manual configuration of new circuits, and the reservation of such circuit wastes bandwidth.

In a network with dynamically changing traffic patterns (which is typical in any packet network), the only way to optimize network utilization without discarding traffic is to have a feedback mechanism built into the network. The feedback mechanism informs the traffic sources of the capacity available on the network so that the sources can adjust the rate at which they inject traffic into the network.

The MAC entity on each node monitors the utilization on its immediate links and makes that information available to all the nodes on the ring. Each node can then either send in more data or throttle back. This efficient use of bandwidth enables RPR rings to scale beyond 95% of their total capacity. Ethernet switches or SONET ADMs have no bandwidth management capabilities and, hence, cannot maximize network utilization.

1.1.4 Quality of Service (QoS) in RPR

Quality of Service (QoS) refers to the capability of a network to provide better service to selected network traffic over various technologies, including Frame Relay, Asynchronous Transfer Mode (ATM), Ethernet and 802.1 networks, SONET, and IP-routed networks that may use any or all of these underlying technologies. The primary goal of QoS is to provide priority including dedicated bandwidth, controlled jitter and latency (required by some real-time and interactive traffic), and improved loss characteristics. Also important is making sure that providing priority for one or more flows does not make other flows fail. QoS technologies provide the fundamental building blocks that are used for future business applications in campus, WAN, and service provider networks [7].

RPR implemented some of the QoS technologies. Traffic is categorized into three classes: ClassA, ClassB and ClassC, as proposed by IEEE 802.17 Work Group, which can be differentiated as different priority traffic, the higher the priority the lower the end-to-end delay. RPR has the ability to differentiate between low- and high-priority packets. Just like other quality of service (QoS)-aware system, nodes have the ability to transmit high-priority packets before those of low priority. In addition, RPR nodes also have a transit path, through which packets destined to downstream nodes on the ring flow. With a transit buffer capable of holding multiple packets, RPR nodes have the ability to transmit higher-priority packets while temporarily holding other low-priority packets in the transit buffer. Nodes with smaller transit buffers can use bandwidth-control message to ensure that bandwidth reserved for high-priority services stays available.

As RPR needs to maintain QoS guarantee for service classes, each class of traffic is rate controlled by a shaper with which token bucket algorithm is running, so the traffic won't be sent beyond its predefined rate. . (However the transit traffic is not subject to rate control.) This also has the effect of limiting the strict precedence of transmit decisions so that each service class gets its fair share of transmissions.

1.1.5 Classes of Service in RPR

To provide priority to certain traffic, the traffic must first be identified and (if desired) marked. These two tasks are commonly referred to as classification. As shown in Figure 2.1, mac_intf module in LLC layer accepts upper layer traffic and denotes them as three classes of service, ClassA, ClassB and ClassC. The service classes are explained next.

1.1.5.1 Service Class ClassA

ClassA service provides an allocated, guaranteed data rate and a low end-to-end delay and jitter bound. It is marked as high priority traffic. Time sensitive traffic falls into this category, for example, the voice traffic. The low delay is the key index to reflect the quality of the service. Therefore, the lesser the delay the better the service.

MAC further divides this class into two sub classes, subclassA0 for reserved bandwidth and subclassA1 for reclaimable bandwidth, as described in Table 1.1. ClassA traffic is not subject to the fairness algorithm at ingress to the ring or when transiting through the ring. ClassA traffic moves through the primary transit path in each node as it propagates around the ring.

1.1.5.2 Service Class ClassB

ClassB service provides an allocated, guaranteed data rate, and bounded end-to-end delay and jitter for the traffic within the allocated rate, and access to additional best effort data transmission that is not allocated, guaranteed, or bounded, and is subject to the fairness algorithm. ClassB service has similarities to ClassA service in that frame transmission rates within the allocated rate profile (known as ClassB committed information rate, or ClassB-CIR) are guaranteed a bounded delay and jitter, although with higher bounds than for ClassA frames, and are not marked as fairness eligible. Traffic within the allocated rate profile is not subject to the fairness algorithm at ingress to the ring or when transiting through nodes on the ring.

ClassB traffic also has similarities to ClassC service in that traffic beyond the allocated rate profile (known as ClassB excess information rate, or ClassB-EIR) is subjected to the fairness algorithm. Fairness eligible frames are counted as part of the RPR fairness algorithm both at ingress to the ring, and while transiting nodes on the ring. ClassB traffic moves through the secondary transit path, regardless of whether the frame marked as fairness eligible or not.

ClassB traffic is marked as medium priority traffic. Traffic not too much sensitive to time falls into this category, for example, the video traffic.

1.1.5.3 Service Class ClassC

ClassC service provides a best-effort traffic service with no allocated or guaranteed data rate and no bounds on end-to-end delay or jitter. ClassC traffic is always subject to the fairness algorithm, and is marked by the MAC as such with the *fe* bit in the RPR header prior to transmission on the ring. ClassC frames are counted as part of the RPR fairness algorithm both at ingress to the ring, and while transiting nodes on the ring through the secondary transit path. ClassC traffic is marked as low priority traffic. Traffic not sensitive to time falls into this category, for example, the data traffic.

Table 1.1 summarized the properties of each service class:

Class of Serive			*Quality of Service*			
Name	Example Use	Subclass	Guaranteed Bandwidth	Delay/Jitter	Bandwidth Type	Bandwidth Subtype
ClassA	real time	SubclassA0	Yes	low	Allocated	Reserved
		SubclassA1				Unreserved
ClassB	near real time	ClassB_CIR	Yes	Bounded	Allocated	
		ClassB_EIR	No	Unbounded	Opportunistic	
ClassC	best effort	-				

Table 1.1: Service Classes

On ingress, ClassA traffic has higher precedence, than, ClassB traffic which has higher precedence than ClassC traffic. On transit, ClassA traffic has higher precedence than ClassB or ClassC traffic. No distinction is made between ClassB and ClassC traffic during transit. Strict precedence on ingress and transit is used to ensure that frames are not reordered within a service class, and to provide the bandwidth guarantee with delay and jitter bounds specified for each service class.

1.1.6 Broadcast or Multicast Traffic in RPR

Packet Rings are a natural fit for broadcast and multicast traffic. As detailed above, for unicast traffic, nodes on a Packet Ring generally have the choice of stripping packets from the ring or forwarding them. However, for a multicast, the nodes can simply receive the packet and forward it, until the source node strips the packet. This makes it possible to multicast or broadcast a packet by sending only one copy around the ring.

1.1.7 Advantages of RPR over SONET/SDH and Ethernet

Data, rather than voice, dominates today's bandwidth requirements. New services such as IP VPN, voice over IP (VoIP) and digital video are no longer confined within the corporate local-area network (LAN). These applications are placing new requirements on metropolitan-area network (MAN) and wide-area network (WAN) transport. RPR is uniquely positioned to fulfill this bandwidth and feature requirements as networks transition from circuit-dominated, like SONET/SDH, to packet-optimized infrastructures.

1.1.7.1 Disadvantages of SONET/SDH and Ethernet

Historically, separate parallel networks have been used to deploy data, voice and video services. However, it has become more efficient to converge the multiple services over a single network. These converged services typically include committed information rate (CIR) services such as virtual private networks and business Internet access; best-effort, consumer-oriented Internet access; voice-over-Internet-Protocol (VoIP); packet video; and time-division multiplexing (TDM) services.

Those services have their own network-performance requirements, which put designers under extreme pressure to accommodate widely varying degrees of latency, bandwidth efficiency, quality-of-service (QoS) and overall reliability and manageability. For metro data networks, SONET and Ethernet have garnered much of the attention, with many fiber rings currently deployed as SONET/SDH networks. Neither technology, however, is optimized for MAN data applications. In the case of SONET/SDH, customer data is transported over a TDM infrastructure back to the provider. The legacy SONET/SDH add-drop multiplexer network offers customer access in the form of DS-0, T1 and T3 interfaces, and sometimes Ethernet. The advantages of SONET/SDH are that it offers good jitter and latency performance, and it is a lossless transport in that packets are delivered from source to destination without any QoS decisions in between. In addition, SONET/SDH uses spare fibers or capacity to provide protection in case of fiber cuts or equipment failures.

To better suit data applications, SONET/SDH has implemented improvements such as virtual concatenation and Link Capacity Adjustment Scheme (LCAS). Nonetheless, with roots as a phone network, it's not the most efficient transport as it creates point-to-point circuits. Also, bandwidth is reserved for every source on the ring, meaning other nodes cannot claim unused bandwidth. SONET/SDH networks can also be expensive, due to multiple layers of equipment such as routers and switches.

For its part, Ethernet is both cheap and ubiquitous, but it's not a carrier-class solution as it provides point-to-point connections: At every hop on the ring, a router or switch processes each packet, which can be time-consuming for large rings. As a result, Ethernet would have trouble meeting the jitter and latency requirements for voice and video.

1.1.7.2 Advantages of RPR over SONET/SDH and Ethernet

RPR technology is a new Layer 2 protocol for the MAN and WAN that appeals to carriers and service providers seeking to quickly deploy the newer, profitable services. Combining the advantages of SONET and Ethernet, RPR allows the support of the newer services while simultaneously supporting traditional carrier-class features such as resiliency, restoration and QoS.

Standout features of the RPR MAC include a transit path to avoid packet switching, as well as express treatment throughout the network of all marked packets to support jitter- and latency-sensitive traffic. Resilience on a par with SONET is achieved with 50-ms protection switching, while bandwidth efficiency is achieved by delivering packet services instead of circuits. Bandwidth management gives the flexibility to oversubscribe the total ring bandwidth with a greater number of users for certain (non-guaranteed) services and temporarily reclaim reserved bandwidth from idle nodes.

RPR utilizes bandwidth traditionally set aside for SONET protection and strips data from the ring when it reaches its destination, leaving spare bandwidth to be reused (spatial reuse). This stripping yields a dramatic increase of the effective bandwidth over the equivalent SONET network. For multicast packets, one packet is circulated to multiple nodes. This is more efficient than the flooding with multiple packets by Ethernet.
For reduced latency, RPR uses dual counter-rotating rings (Inner Ring and Outer Ring), each carrying working traffic. To transmit, each node has a topology map to help it choose a ring based on the least-hop count to a destination (or some other cost metric such as distance). Unlike Ethernet rings, each RPR node processes only the packets addressed to it. Other packets quickly transit through the RPR MAC.

RPR networks have the ability to carry multiple services, including jitter- and latency-sensitive traffic such as voice and video in addition to Ethernet and Internet Protocol (IP) services. RPR combines the best features of legacy SONET/SDH and Ethernet into one layer to maximize functionality while delivering high quality service.

1.2 Contributions

Contributions made in this thesis are: developing an in-house version of RPR model based on OPNET Modeler, developing an alternative traffic control algorithm – Fuzzy

Logic Control (FLC), and deriving equations for Fairness Round Trip Time (FRTT) and the upper bound of minimal Second Transit Queue (STQ) size.

1.2.1 Developing RPR Model in OPNET

RPR is an emerging layer-2 protocol that hasn't been standardized yet, it is still in the research and developing phases among communication network industry and institutions, so there are not too many vendors which intend to manufacture RPR equipments. Those who intend to do have the task of compliance with IEEE 802.17's future standard, as this standard is evolving rather fast. At present time, Cisco has made Spatial Reuse Protocol (SRP) as its RPR counterpart. SRP was implemented at least two years ago. Therefore, the protocol may not comply with the latest IEEE 802.17's development. On the other hand, there is no publicly available, tried-and-tested software code that is fully configurable and workable. For the purpose of the research at Ryerson University, one had to start simulating RPR behavior and to study its performance empty handed. Other requirements such as different features needed for research, and the flexibility of the code made it paramount to create an in-house software code, customizable enough so one can use it as a main platform for the further research on PRR. The first major contribution of this thesis is developing a set of RPR simulation model that are fully compliance with IEEE 802.17 Work Group's most-up-to-date proposal. The model set includes node module, packet modules and link module. They all come with many configurable parameters which allow simulating most of the real world scenarios.

The simulation program we developed is based on OPNET Modeler. OPNET Modeler is a powerful network simulation tool that is used to model communication networks and distributed systems [5]. It provides a comprehensive development environment so the behavior and performance of modeled systems can be easily analyzed and studied. Furthermore, it provides a C-styled language, called proto-C, to allow developers write their own programming codes for media access control (MAC) layer, logical link control (LLC) layer, network layer, and so on to suit to their research and analysis purposes.

OPNET was originated from a research group in MIT in the early 1980's and was finally commercialized and incorporated as a company in 1986. Modeler is one of the software developed by OPNET, it is now widely used by most major network equipment manufactures, research and educational institutes, service providers, military and defense department and so on, some of the names are Cisco, AT&T, NASA, University of California - Berkeley, etc.

1.2.2 Developing An Alternative Traffic Control Algorithm - Fuzzy Logic Control

One can realize after extensive study of IEEE 802.17 Work proposal that there are some limitations of fairness algorithms: when the node congested, the throughput of the node often oscillates and it causes the throughput of all the upstream nodes to oscillate too. To overcome it, Fuzzy Logic Control is developed and applied as an alternative. The second contribution in this thesis is to apply Fuzzy Logic to RPR technology and to prove that Fuzzy Logic can be used in controlling traffic in RPR.

1.2.3 Devised Equations For The Upper Bound Of Minimal STQ Size

Fairness Round Trip Time (FRTT) is a key parameter to design the size of Second Transit Queue (STQ), which is used to temporarily hold the low priority traffic while the node is processing high priority traffic. The size of STQ can not be set too small, otherwise the STQ can easily overflows; it can not be set too large, neither, otherwise traffic held in STQ would spend longer time to be processed, thus, delaying the transmission and degrading the quality of services. Optimizing the size of STQ is an interesting and challenge work.

IEEE 802.17 Work Ground proposed an equation to estimate the upper bound of FRTT, however the equation doesn't provide a true upper bound. The third contribution of this thesis is to devise a much more effective upper bound equation for FRTT and to estimate the required STQ size.

1.3 Thesis Overview

This thesis is organized as follows: Chapter 2 describes the model designed in the lab, its architecture, properties and functionalities. The model consists of a group of modules. They are the link module, node module and packet modules. Attributes of each of them is discussed in details. The Node module has several layers of components. Packet generators locate at the top layer, they are used to generate different classes of traffic, i.e. ClassA, ClassB and ClassC traffic. The traffic can also be generated in different patterns, for example constant traffic, bursty traffic. MAC module locates at the middle layer and RPR protocol suite is at this layer. It is used to control the bandwidth, detect the congestion, direct traffic, and so on. The majority of the code is written inside this module. Receivers and Transmitters are located at the bottom layer, they are in charge of receiving and transmitting packets from and to the node.

The protocols and algorithms implemented in each model, for example, rate control protocol, fairness algorithm, or traffic shaper, are discussed in details

Chapter 3 presents Fuzzy Logic Control as an alternative algorithm to the current IEEE 802.17 proposed algorithm. We discuss Fuzzy Logic Control is discussed in theory. The Chapter demonstrates why and how FLC can overcome the shortcomings of the existing algorithms. Finally, we compare the results of simulation from FLC algorithm with IEEE 802.17 algorithm are compared.

In Chapter 4, FRTT and minimum STQ size are investigated. To avoid STQ overflow caused by ClassA traffic, an upper bound equation for FRTT is derived and used to determine the minimum STQ size required to prevent overflow.

Chapter 5 concludes the thesis and addresses some of the future plan.

Chapter 2: The In-House Version Of RPR Model

A major contribution of this thesis is to design and develop an in-house version of the RPR module in OPNET for research purposes in the Department of Electrical and Computer Networking, at Ryerson. This version adopts IEEE P802.17/D1.1 as the blue print. This chapter discusses this model, what properties it comes with, and what functions it can provide.

2.1 The Design of the In-House RPR Model

The challenge is to design a RPR model with different components in it. Such design should comply with OSI layers to better simulate the real world situations, in other words the different components are located inside different layers, not across. Each component of the model would illustrate a clear picture of what functions they can provide. Another advantage of adopting such design is that each component of the model can be independent from each other: the communication between them is only through the flow of packets, which is the fundamental principle of OSI layers.

Based on the above principle, the in-house RPR OPNET model is designed in four layers. Figure 2.1 illustrates these four layers and their corresponding OSI layers. The upper layer[1] consists of four components: three traffic generators and one traffic sink. Traffic generators would generate ClassA, ClassB and ClassC traffics. These traffics are sent to the lower layer, which is Logical Link Control (LLC) layer. The traffic sinks from LLC layer by the Traffic Sink. Module mac_intf is located in Data Link Layer (DLL) of LLC layer. It marks the received upper layer packets as Data frames and puts source address, destination address and service class to the header of the frames, then, sends it to the lower layer, which is Media Access Control (MAC) layer. Module mac is located in MAC Layer. RPR protocol suite is implemented in this module. Main functions of this

[1] The traffic generators and sink are taken directly from OPNET, because it is unnecessary to spend time writing a different version of program that achieves the same functionality. Same holds true for Receiver and Transmitter modules.

module include Ring Protection, Topology Discovery, Ringlet Selection, Bandwidth Control, Fairness Algorithm, Spatial Reuse, QoS control, OAM, etc. The bottom layer is the Physical layer, which consists of two sets of receivers and transmitters, one per each ring. Module IRx and ITx are the inner ring receiver and transmitter, respectively; module ORx and OTx are those of the outer ring. Figure 2.1 shows that the two sets of receivers and transmitters operate in opposite direction, which is a characteristic of RPR.

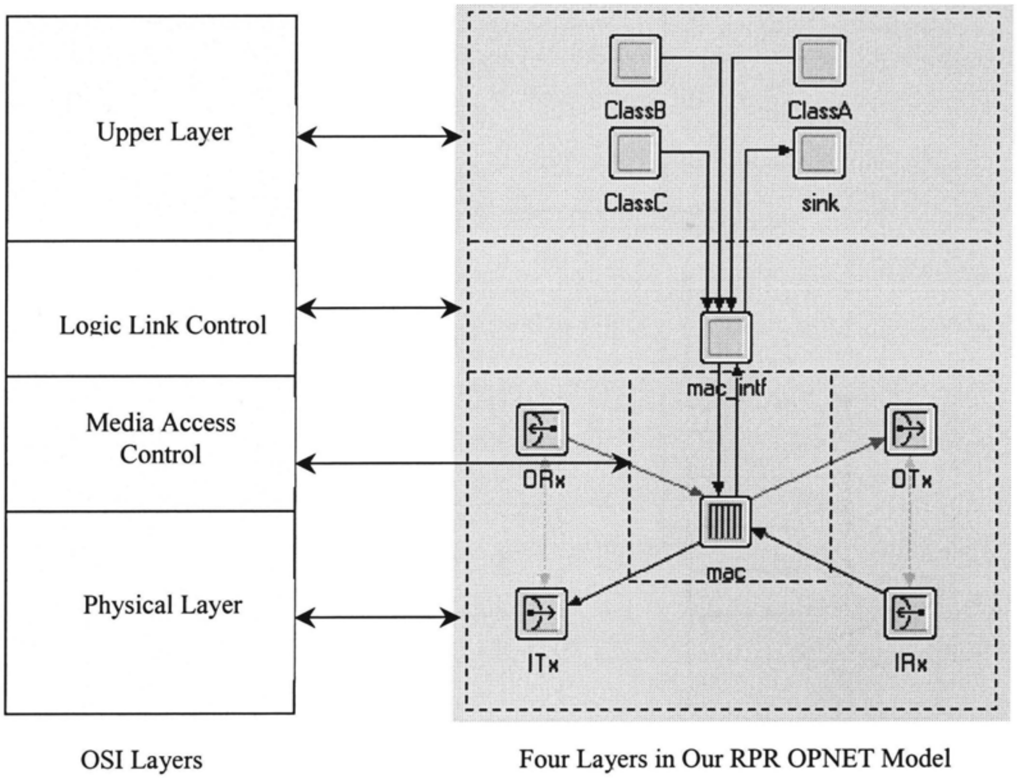


Figure 2.1: Our RPR OPNET Model

2.1.1 The Model: Internal Components and Their Functionality

Figure 2.1 shows a general layout of the RPR model. How each component works and what functions they provide are discussed in this Section. Figure 2.2 is the same as Figure 2.1, except that it is magnified for the purpose of discussion. This Figure shows in more details how each component cooperatively works together to fulfill the commitment that

RPR claims. It marks the functionality of each component and the interconnection between them.

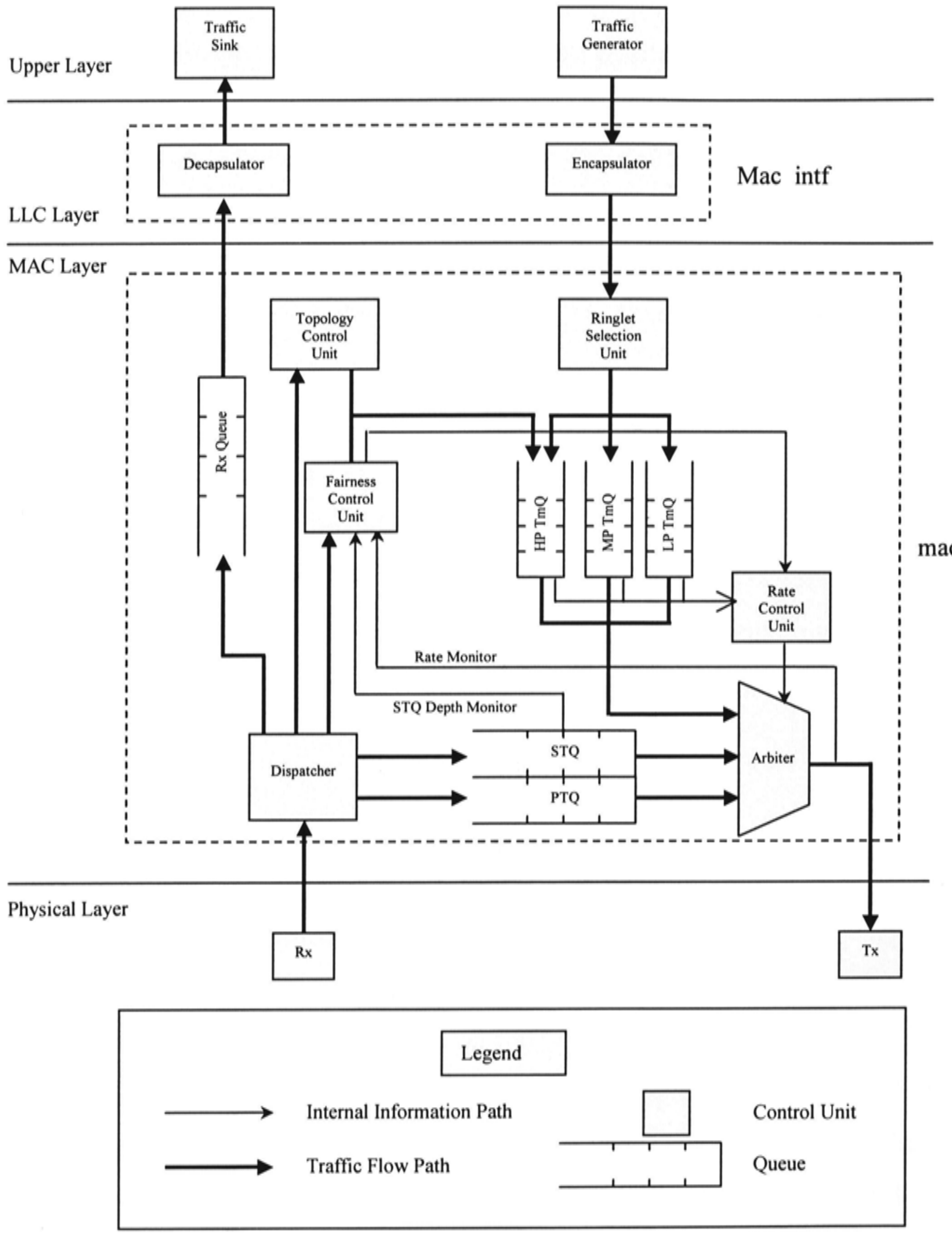


Figure 2.2: Magnified Model

2.1.1.1 Upper Layer Components

Traffic Generator and Traffic Sink are located in the top layer. Traffic Generator can generate different traffic patterns, including constant traffic and bursty traffic, in order to better simulate the real world scenarios. It can generate different traffic classes as well, for example, ClassA, ClassB and ClassC traffics. When the traffic is sent to the lower layer, they are encapsulated by RPR header and trailer and are marked as RPR data frames, therefore the format of the traffic is transparent to RPR, in other words, RPR doesn't care what format the traffic in this layer is.

Traffic rate is controlled by generator's on/off time, packet interarrival time and packet size. For example, if generator's ON time follows exponential(10 ms) rule and OFF time follows exponential (20 ms) rule, packet interarrival time follows exponential (0.115 ms) rule and packet size is 12000 bit, then we have the traffic rate of

$$R = \frac{12000}{0.115 \times 10^{-3}} \times \frac{10}{10+20} \approx 34.8 Mbps$$

in average.

Traffic Sinker sinks the traffic.

2.1.1.2 LLC Layer Components

Encapsulator and decapsulator are located in the LLC layer. Encapsulator can distinguish different classes of traffic received from the upper layer, encapsulates the traffic with RPR header and trailer and marks the Service Class field according to its class. It also fills the Source Address, Destination Address field and other fields then send the traffic to the lower layer. If the traffic is sent to multiple destinations, encapsulator uses Weighted-Round-Ribbon algorithm to decide what destination address to fill in the destination field in the header. For example, suppose a node sends traffic to node A with Weight value of 1 and to node B with Weight value of 2. Then, in any three time units of filling in destination field and sending traffic to the lower layer, node A gets one time unit and node B gets two time units.

Decapsulator receives the traffic from the lower layer, removes the header and trailer, then, sends the traffic to the upper layer.

2.1.1.3 MAC Layer Components

2.1.1.3.1 Ringlet Selection Unit

After packet is encapsulated, it is sent to Ringlet Selection Unit (RSU) in MAC layer. RSU marks the packet with the proper ringlet id, which depends on how the node is configured. As per Table 2.1, ringlet selection is one of the node configurable attributes, we can set Inner Ring, Outer Ring or Auto Selected as the value to this attribute. If we set it to Inner Ring, then the ringlet id field in the header of all the packets received from the upper layer is changed to 0. From then on until it reaches its destination it flows in Inner Ring and is processed only by the Inner Ring Receiver and Transmitter. This way an Inner Ring packet does not mess up with Outer Ring packets or vice versa. However if we set to Auto Select, RSU runs Ringlet Selection Protocol to decide which ringlet id to change to. Ringlet Selection Protocol is explained in details in the next section.

After the ringlet id field in the header is changed, RSU sends the packet to High-Priority Transmit Queue (HP-TmQ), Medium-Priority Transmit Queue (MP-TmQ) or Low-Priority Transmit Queue (LP-TmQ), depending on which class of traffic the packet belongs to, and the packet waits there to be served by arbiter.

2.1.1.3.2 Rate Control Unit

A node is not permitted to use more than its fair-shared bandwidth for the insertion of fairness eligible traffic when congestion has been detected on a ringlet. This restriction prevents a node from using a disproportionate share of available capacity by virtue of its relative position on the ring (The rate restriction is enforced by a shaper within the MAC datapath sublayer.). Without the regulated access, upstream node would take advantage of all available capacity during periods of congestion, preventing further downstream nodes from inserting opportunistic traffic. Therefore, in order to implement fair access, it is necessary to use shapers to restrict the occupancy of available capacity during periods of congestion, eliminating the access advantage of upstream nodes.

To allow traffic being transmitted out, the shaper of such traffic must be set to true. The value of shaper is determined by the amount of tokens for such traffic. If the amount of tokens greater than a threshold, it is set to true; otherwise it is set to false. Tokens are increased as the time elapses and decreased as traffic is sent out. When a flow of traffic reaching or exceeding its predefined rate, the amount of tokens for such traffic decreases to below a threshold, and the shaper of it is set to false, thus the traffic is held in the queue and not allowed to be sent out until enough tokens available. RCU uses this way to keep the traffic within the predefined rate in average.

Rate Control Unit (RCU) adapts token bucket traffic shaper algorithm [6] to control different classes of traffics within their predefined rate. Each class of traffic comes with a shaper to regulate its traffic: shaperA0, shaperA1, shaperB, shaperC, shaperD and shaperM. These shapers are explained in detail in the next section.

2.1.1.3.3 Dispatcher

Upon accepting a packet from Receiver (Rx), Dispatcher checks the header of the packet to decide where to forward the packet. If the packet is a topology discovery packet, Dispatcher forwards it to TCU. If the packet is a fairness packet, Dispatcher forwards it to FCU. If the packet is a data packet and the destination is this node, Dispatcher forwards it to Receiving Queue (Rx Queue). If the destination of the packet is not this node, Dispatcher forwards it to Primary Transit Queue (PTQ) or Secondary Transit Queue (STQ) depending on the traffic class the packet belongs to, then the packet stays there and wait to be served by arbiter.

2.1.1.3.4 Topology Control Unit

Topology discovery packet is sent to Topology Control Unit by Dispatcher. Source address, hop count and other information of the packet are retrieved by TCU and TCU uses that information to construct Topology Database. With the Topology Database, the node finds out how far it is to another node on the ring. This information is critical in fairness algorithm. Periodically, TCU broadcasts topology discovery packet to inform other nodes in the ring of its existence.

2.1.1.3.5 Fairness Control Unit

The Dispatcher sends fairness packet to the Fairness Control Unit. Upon receiving the fairness packet, FCU retrieves the value from fairness control field in the packet. It then compares this value with the one it got from fairness algorithm and send the lower value of the two to Rate Control Unit. Fairness algorithm is explained in details in the next section of this chapter.

Fairness Control Unit is used to regulate low-priority traffic. It needs to make sure all the nodes participating in the ring could get their fair shared bandwidth while not affecting higher class of traffic. Because the nodes dynamically participate in the ring, we can't assign a fixed rate to each node; on the contrary, the node should dynamically determine the fair rate and limit itself to this rate. Fairness algorithm serves this purpose.

However, fairness algorithm is only activated when FCU detects congestion in the node. Congestion is detected when either packets accumulated in the STQ over a predefined threshold, normally one eighth of the total STQ size, or transmit rate over unreserved rate. When one of these two events happens, FCU runs fairness algorithm and get a fairness value. This value is compared with the one retrieved from fairness packet and the smallest of the two is fed into Rate Control Unit. RCU, then, uses it to limit fairness eligible traffic in the next aging interval. Meanwhile, FCU generates a fairness packet and insert this fairness value as fairness control value and sends it to its upstream neighbor. By repeating the same procedure in each node, the output rate of each node can be limited to the fairness value.

2.1.1.3.6 Queues

For whatever reason, if traffic that cannot be processed by the transmitter after it is received by the node it must be either discarded or temporarily held in a queue waiting to be processed. MAC layer contains three sets of queues: receive queue, transmit queues and transit queues. Each queue serves different purposes. Receive Queue (RxQueue) is used to hold traffic destined to the node itself. Transmit queues are used to hold the traffic generated by the node itself. They are further divided into three sub queues based on the class of the traffic it would hold: High Priority Transmit Queue (HP_TmQ) hold ClassA traffic, Medium Priority Transmit Queue (MP_TmQ) hold ClassB traffic and Low Priority Transmit Queue (LP_TmQ) hold ClassC traffic. Transit queues are used to hold the traffic by-passing the node. They are further divided into Primary Transit Queue (PTQ) and Secondary Transit Queue (STQ). PTQ only holds ClassA traffic, STQ holds ClassB and ClassC traffic.

By dividing queues into different categories and sub queues, higher priority class of traffic gets the chance to be transmitted before lowering priority class of traffic. Therefore, head of line problem is avoided and the delay and jitter bound imposed on higher class of traffic is guaranteed.

2.1.1.3.7 Arbiter

Traffics held in the queues wait to be served by Arbiter and to be sent out. Arbiter uses a priority-based traffic management method to decide which traffic should be served, which is shown in the following sequence:

- If there is a packet in PTQ, send the packet out from PTQ; otherwise,

- If packets accumulated in the STQ is more than the size of STQ minus MTU, send the packet out from STQ; otherwise,
- If there is a packet in HP-TmQ and HP-TmQ is online, send the packet out from HP-TmQ; otherwise,
- If there is a packet in MP-TmQ and MP-TmQ is online, send the packet out from MP-TmQ; otherwise,
- If there is a packet in LP-TmQ and LP-TmQ is online, send the packet out from LP-TmQ; otherwise,
- If there is a packet in STQ, send the packet out from STQ; otherwise,
- If there is no packet in any queue, arbiter goes to idle.

This method guarantees the minimal jitter and latency on high-priority traffic.

2.1.1.4 Physical Layer

Receiver (Rx) and Transmitter (Tx) are located in physical layer. IEEE 802.17 work group proposes SONET and/or Ethernet (Gigabit Ethernet and 10Gigabit Ethernet) as the transmission media for RPR in this layer.

2.1.2 OPNET Node Attributes

Each OPNET node module has its own attributes. The attributes of traffic generator in the upper layer mainly consist of Packet Format, Packet Interval Time, Packet Size, Start and Stop Time. By setting up different values for Packet Interval Time and Packet Size, the system can start and stop generating different rates for different traffic types at different time, which provides great flexibilities for the simulations.

Mac_intf node module has only one attribute associated with it, Weighted Destination Parameters, a compound value. It includes Destination MAC Address and Weight. Destination MAC Address is inserted into the header of the frame before being transmitted to the lower layer. If Weight is taken into account (default is 1), mac_inft node is send the traffic to its destination in the amount based on the Weight value associated with it.

Attributes in the MAC node module can be put into three categories, Node Configuration, Fairness Parameters and Station Address. Node Configuration includes the setup for Two Transit Buffer (2TB) Mode, Double Low Priority Transmit Buffer (DLPTB), Swap Stage Queues, Ringlet Selection and Rate Allocation. By changing the values of these configurable attributes the system can be easily setup to run in different scenarios and the behavior of the RPR protocols can be studied in more details. Table 2.1 shows the values of each attribute and its description.

Attribute	*Value*	*Description*
2TB Mode	Enable	The module has two transit buffers in each ring, PTQ and STQ.
	Disable	The module has one transit buffer in each ring, PTQ.
DLPTB Mode	Enable	The module has two low priority transmit buffers, one for holding frames destined before congestion point, one for beyond congestion point.
	Disable	The module has only one low priority transmit buffer.
Swap Stage Queue	Enable	Sending traffic from two low priority transmit buffers in Round-Robin fashion.
	Disable	Sending traffic from one low priority transmit buffer.
Ringlet Selection	Auto Selected	The module sends the traffic based on the close-path-first algorithm.
	Inner Ring	The module sends the traffic to Inner Ring.
	Outer Ring	The module sends the traffic to Outer Ring.
Rate Allocation	ClassA0	User defined traffic rate for ClassA0 traffic.
	ClassA1	User defined traffic rate for ClassA1 traffic.
	ClassB_CIR	User defined traffic rate for ClassB CIR traffic

Table 2.1: Node Configuration Attributes

Fairness Parameters includes Fairness Algorithm Mode, Fairness Message Type, Advanced Fairrate Propagation, age_coef, lp_coef, ramp_coef and Weight. By changing the values of these parameters the behavior and performance of fairness algorithm can be easily studied. Table 2.2 shows the values of each parameter and its description.

Parameter	*Value*	*Description*
Fairness Algorithm Mode	Aggressive	The module runs Aggressive fairness algorithm.
	Conservative	The module runs Conservative fairness algorithm.
	Fuzzy in Aggressive	The module runs Aggressive fairness algorithm with Fuzzy Logic Control.
	Fuzzy in Conservative	The module runs Conservative fairness algorithm with Fuzzy Logic Control.
Fairness Message Type	Single Chock	The module only response to the lowest fairness advertising rate it received from downstream nodes.
	Multi Chock	The module response to each fairness advertising rate it received from downstream nodes.
age_coef	(user defined)	Age coefficient
lp_coef	(user defined)	Lp coefficient
ramp_coef	(user defined)	Ramp coefficient
Weight	(user defined)	The ratio of maxim bandwidth assigned to the module. The ratio is defined as $\frac{LineSpeed}{TotalModue} \times Weight$, default is 1.

Table 2.2: Fairness Parameters

Station address is defined in Table 2.3.

Station Address	Auto Assigned	The module's MAC address is auto-assigned by the system.
	(user defined)	The module's MAC address is assigned by the user. In this case, it is user's own responsibility to make sure there is no duplicated address among the nodes.

Table 2.3: Station Address Parameters

2.2 OPNET Link Model

In order to simulate real world optical fiber, RPR OPNET link module is designed to be a simplex point-to-point link, which means it can only accepts one-way traffic. With a pair of bi-directional dual counter-rotating rings connected to its MAC node, the module is able to transmit and receive traffics with its adjacent neighbors. The data transmit rate of the link is set as 622.08 Mbps to simulate OC-12 optical signal. To simplify the calculation, the delay in each link is set to a fixed value, 0.07 ms, to simulate the distance of 15 kilometers between two adjacent neighbors. The link can only accept the Data frame, Control frame and Fairness frame. The next section discusses these formats in details.

The packets are sent from the upper layer in any format, e.g. IP or IPX. When packets arrive at the Logic Link Control layer from the upper layer, as illustrated in Figure 2.1, they are encapsulated with RPR header and converted to RPR Data format, then sent to the MAC layer. When frames arrive at the Logic Link Control layer from the MAC layer, the header of frames is deprived and the de-capsulated frames are sent to the upper layer for further processing. Except data frame, other format are generated and stripped in the MAC layer, simply because they are the control frames.

2.3 Frame Format

The in-house RPR OPNET model includes three different frame formats, data frame format, control frame format and fairness frame format. The header of all these formats is the same. The following paragraph illustrates the header and the body of the frame in details.

2.3.1 Frame Header

Each frame has a fixed size header. Table 2.4 shows the header format of non-fairness frame. Table 2.5 shows header format of fairness frame.

Header Field	*Field Length (bit)*	*Description*
TTL	8	Time-To-Live, a hop count that specifies the maximum number of hops the frame is expected to cover before reaching the destination.
BaseControl	8	Basic control field
DA	48	Destination Address
SA	48	Local Station Address
TTLBase	8	Initial value of the TTL field upon transmission of a data frame.
ExtendedControl	8	Additional control field only for Data frames.
HEC	16	Header Error Checksum

Table 2.4: RPR Non-Fairness Frame Header Format

Header Field	*Field Length (bit)*	*Description*
TTL	8	Time-To-Live, a hop count that specifies the maximum number of hops the frame is expected to cover before reaching the destination.
BaseControl	8	Basic control field
SaCompact	48	Source station that provided the values in fairnessHeader and fairRate fields.

Table 2.5: RPR Fairness Frame Header Format

2.3.2 BaseControl Field

BaseControl field is shown in Table 2.6.

Sub Field	*Length (bit)*	*Description*
RI	1	Ringlet Identifier. Indicates the ringlet onto which the frame was originally transmitted. 0 – Inner Ring, 1 – Outer Ring.
FE	1	Mark the frame whether fairness eligible subject to the RPR fairness algorithm. 0 – Not fairness eligible, 1 – fairness eligible.
FT	2	Frame Type. 00_2 - Idle Frame; 01_2 - Control Frame; 10_2 - Fairness Frame; 11_2 - Data Frame.
SC	2	Service class of the frame. 00_2 – Class C; 01_2 – Class B; 10_2 – Class A, subclass A1; 11_2 - Class A, subclass A0.
WE	1	Indicates that the frame is eligible to be wrapped during a wrap condition. 0 – Not wrap eligible, 1 – wrap eligible.
reserved	1	Reserved for future use.

Table 2.6: BaseControl Field

2.3.3 Data Frame Format

Table 2.7 shows the RPR Data frame format:

Field	*Length (bit)*	*Description*
header	144	Non-Fairness Frame Header.
ProtocolType	16	>= 1536, MAC client protocol (type interpretation) < 1536, length of frame (length interpretation)
SDU	n	Service Data Unit provided by the MAC client.
fcs	32	Frame Check Sequence, a CRC of the frame.

Table 2.7: RPR Data Frame Format

2.3.4 Control Frame Format

Table 2.8 shows the RPR Control frame format:

Field	*Length (bit)*	Description
header	144	Non-Fairness Frame Header.
ControlType	8	Type of control frame. 01_{16} – Station Configuration Discovery frame; 02_{16} – Topology and Protection frame; 03_{16} – OAM control frame; 04_{16} – Vendor specific discovery frame; other - Reserved
ControlVer	8	Version number of ControlType field.
CDU	n	Control Data Unit
fcs	32	Frame Check Sequence, a CRC of the frame.

Table 2.8: RPR Data Frame Format

2.3.5 Fairness Frame Format

Table 2.9 shows the RPR fairness frame format:

Field	*Length (bit)*	*Description*
header	48	Fairness Frame Header.
FairnessHeader	16	Type of fairness frame: 0 – Single Chock: Provides the advertisedFairRate of a station to the upstream neighbor once per advertisementInterval. 1 – Multi Chock: Provides the normLocalFairRate of a station to all other stations on the ringlet once per reporting interval. Other – Reserved.
FairnessRate	16	Normalized rate encoded as a 16-bit quantity.
fcs	32	Frame Check Sequence, a CRC of the frame.

Table 2.9: RPR Fairness Frame Format

2.4 RPR Protocol Suite

RPR is a complicated layer-2 protocol stack that comes with many features. Four major protocols are explained next.

2.4.1 Topology Discovery Protocol

The topology discovery protocol provides a reliable and accurate means for all nodes on a ring to discover each other. This includes both the initialization of the topology and any changes to that topology. The protocol provides each node on the ring with knowledge of the number and arrangement of other nodes on the ring. This collection of information is referred to as the topology database. Each node maintains its own local copy of the topology database for the entire ring. The topology database is also used by other protocols such as the RPR ringlet selection protocol and the RPR fairness algorithm.

Initially, the node's topology database contains information only about itself. The information required to create the basic topology database (including, for example, hop counts per ringlet from the local node to all other nodes on the ring) is derivable from the *ttl* value from the header of topology frames received from each node on the ring. The transmission of Topology discovery frames is initiated as needed and periodically. If the topology database is stable, the periodic Topology discovery frame transmissions do not result in any change to the topology database. A sample topology database of a three-node-ring is shown as Table 2.10.

MAC Address	*Hop count for Ringlet0*	*Hop count for Ringlet1*	*Data Reachability, Ringle0*	*Data Reachability, Ringle1*
00-10-A4-97-A8-DE	0	0	Yes	Yes
00-10-A4-97-A8-AC	1	3	Yes	Yes
00-10-A4-97-A8-BD	2	2	Yes	Yes
00-10-A4-97-A8-CE	3	1	Yes	Yes

Table 2.10: A Sample Topology Database

2.4.2 Ringlet Selection Protocol

The primary responsibility for ringlet selection protocol is to choose the appropriate ringlet for client add frames. This can be done either by manual configuration or based on short-path-first algorithm. With manual configuration, the frames have to be sent to designated ringlet. With short-path-first algorithm, the choice is made based on the lowest hop count to the destination. The algorithm searches the entry of destination address in the topology database and compares the hop counts of two ringlets. The ringlet with the smaller hop count is chosen and is set in the *ri* field in the header of the frame. This is stated in Eq. (E 2.1):

$$\text{Ringlet\_id} = \text{MIN}\ (\text{hop\_count(dest\_addr, ringlet0), hop\_count(dest\_addr, ringlet1)}) \qquad \text{(E 2.1)}$$

Consider Table 2.10 as an example. If local node wants to send frames to destination 00-10-A4-97-A8-CE, it compares the hop counts of both ringlets in the topology database. Because ringlet1 has lower hop count than ringlet0, the ringlet1 is selected.

2.4.3 RPR Fairness Algorithm (RPR-fa)

RPR-fa is a local fairness algorithm that provides a fair access for all nodes on the ring. When congestion happens fairness algorithm decides what the allowed rate for the fairness eligible traffic is in the local node and what the rate it should advertise to its upstream neighbors. RPR-fa is an intelligent algorithm that can dynamically allocate faired bandwidth to all the nodes within the congestion domain[1]. Ideally, such faired bandwidth should be the total bandwidth divided by total number of nodes within the congestion domain.

$$Fair\_rate = \frac{Total\_Bandwidth}{Total\_Number\_of\_Stations} \qquad \text{(E2.2)}$$

[1] A congestion domain is a set of contiguous stations that affected by congestion.

2.4.3.1 RPR-fa Mechanism

The RPR-fa is a mechanism that enforces fairness among the nodes on the ring. It applies only to ClassC and ClassB Excess Information Rate (EIR) traffic coming from the MAC client. Each node is assigned a weight, which allows the user to allocate more ring bandwidth to certain node in congested situation. The RPR-fa does not need to understand the ring topology, however it utilizes the implicit topology information (i.e. TTL value) passed by the MAC client to perform fairness and policing functions. In RPR-fa, if a node experiences congestion, it advertises a fair rate (e.g. add_rate) to upstream nodes via the opposite ring. The fair rate counter is run through a low pass filter function and divided by a weighting function (e.g. local node weight). The low-pass filter stabilizes the feedback, and the division by weight normalizes the transmitted value to a weight of 1.0. Upon receiving an advertised fair rate, the upstream nodes adjust their traffic speed so as its transmission rates not to exceed the received advertisement rate (adjusted by their weights) if such traffic destined beyond the congestion point. Generally, nodes also propagate the received advertisement rate to their immediate upstream neighbor. Nodes receiving advertised values which are also congested propagate the minimum of their normalized low pass filtered advertised fair rate and the received fair rate. If the traffic destines within the congestion point, the node can take advantage of all the available bandwidth on the ring.

2.4.3.2 RPR Fairness Algorithms

Three types of RPR-fa are implemented in RPR (Aggressive-fa, Conservative-fa and Fuzzy-fa) to calculate fair rate while a node is experiencing congestion. Aggressive-fa and Conservative-fa are the proposed by IEEE 802.17 draft, Fuzzy-fa is proposed here to improve the performance of the other two algorithms.

During each aging_interval, the add_rate is accumulated by the size of frame length whenever there is fairness eligible traffic sent out from the node. At the end of the aging_interval, the add_rate runs through a low pass filter function and divided by a weighting function and represented it as lp_add_rate. The Aggressive-fa takes the value of lp_add_rate as the fair rate if the congestion detection unit finds the node is congested; otherwise it takes unreserved_rate as fair rate.

Conservative-fa gets its fair rate in two phases: first, when congestion detection unit finds the node is congested, it enters "just enter congestion" state and takes the value of unreserved_rate/active_stations as its fair rate for the period of this state. After this time period elapses, if congestion detection unit finds the node is still congested, it enters "ramping" state. In "ramping" state, congestion state is monitored. If the traffic accumulated in the STQ is less than low threshold, the congestion is looked as "light congest," in this case the node should "ramp up" and send more traffic than previous interval. The fair rate passes a "ramping up" function and increases its value. If the traffic accumulated in the STQ is higher than medium threshold, the congestion is looked as "heavy congest," in this case the node should "ramp down" and send less traffic than previous interval. The fair rate passes a "ramping down" function and decreases its value. If the traffic accumulated in the STQ is between low threshold and medium threshold, the congestion is looked as "normal congest", in this case the node should not "ramp up" or "ramp down" and it should send the traffic the same as previous interval. The fair rate does not pass the ramping function, it keeps the value of it as is.

Both Aggressive-fa and Conservative-fa have pros and cons. Aggressive-fa is fast in convergent to the fair rate. However, when too many nodes are in the ring, it introduces tremendous oscillation and sometime brings the node into starving state. Conservative-fa, on the other hand, is not likely to introduce oscillation, however, it is slow in convergence to the fair rate and has less utilization of the bandwidth than Aggressive-fa. In order to overcome the limitation mentioned above, a new algorithm, Fuzzy-fa, is introduced that has less oscillation with fast convergency time. See [1] for detailed explanation on Aggressive-fa and Conservative-fa. Fuzzy-fa is explained in details in Chapter 3.

2.4.3.3 Single Chock Fairness Message vs. Multi-Chock Fairness Message

Shaper parameters for fairness eligible traffic are computed using a distributed fairness algorithm. The fairness algorithm relies on fairness messages that are circulated periodically on the ringlet opposing that of the associated data traffic. There are two types of fairness message frame, single chock fairness message frame (SCFF) and multi-chock fairness message frame (MCFF). The SCFF is processed at each node and carries the identity of the most congested node encountered and the time averaged ingress rate, or fair rate, reported by that node. Upon receiving the frame, the node compares its own calculated normalized low pass filtered advertised fair rate with the received fair rate, and insert the lower value of the two into its SCFF and send to the upstream node.

The second type of fairness message frame, MCFF, is broadcasted periodically by all nodes. The MCFF is passed to the fairness control unit for processing and the relevant information is passed to the client to support multi choke fairness. Therefore, in a multi-choke implementation of the RPR-fa, each client tracks advertised fair rates for congested nodes. A node is allowed to send unlimited traffic to any node between itself and the first congested node (choke point). It can send traffic to nodes between the first and second choke point based on the first choke point's advertised fair rate. In general, a node can send traffic to a particular destination if it has satisfied the fair rate conditions for all choke points between itself and the destination. See [1] for detailed explanation on single chock fairness message and multi chock fairness message.

2.4.4 Traffic Rate Control Protocol

A variation of token bucket traffic shaper is implemented in this thesis to control the traffic so as to confine service classes within their allocated rate.

2.4.4.1 Token Bucket Traffic Shaper

Figure 2.3 illustrated the theory of Token Bucket Traffic Shaper (TBTS) [6]. Tokens are generated periodically at a constant rate and are stored in a token bucket. If the token bucket is full, arriving tokens are discarded. A packet from the buffer can be taken out only if a token in the token bucket can be drawn. If the token bucket is empty, arriving packets have to wait in the packet buffer. Thus, one can think of a token as a permit to send a packet.

If the buffer has a backlog of packets when the token bucket is empty, these backlogged packets have to wait for new tokens to be generated before they can be transmitted out. Since tokens arrive periodically, these packets are transmitted periodically at the rate the tokens arrive. If the token bucket is not empty, packets are transmitted out as soon as they arrive without having to wait in the buffer, since there is a token to draw for an arriving packet. Thus, the burstiness of the traffic is preserved in this case. However, if packets continue to arrive, eventually the token bucket becomes empty and packets starts to leave periodically. The size of the token bucket essentially limits the traffic burstiness at the output.

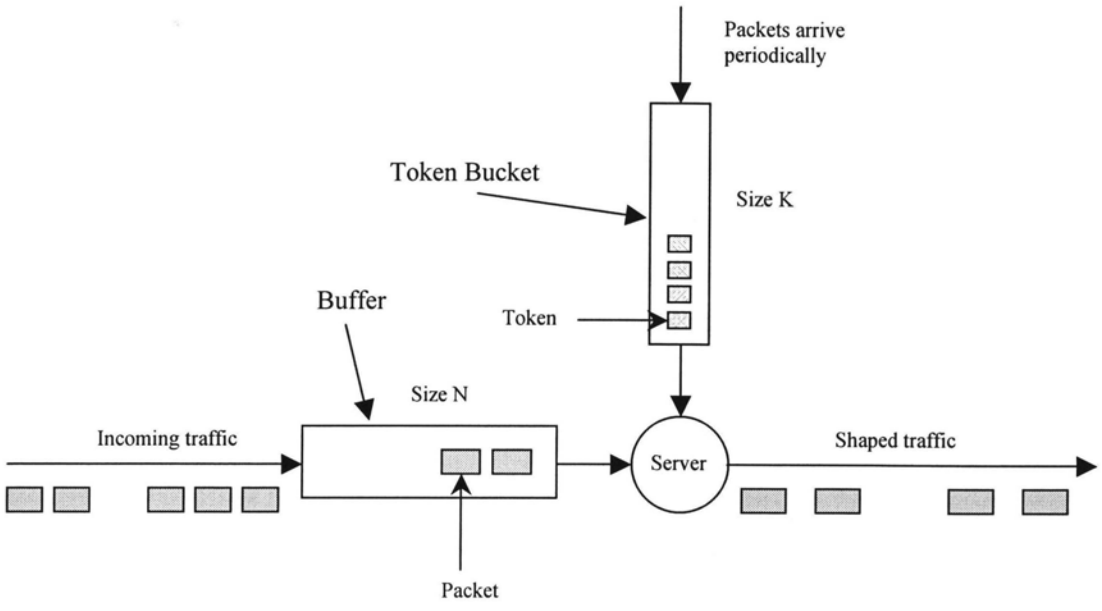


Figure 2.3: Token Bucket Traffic Shaper

2.4.4.2 Token Bucket Traffic Shaper Implemented in RPR

In RPR, a variation of Token Bucket Traffic Shaper[1] is implemented to control different service classes. In this variation, each shaper consists of a token bucket as illustrated in Figure 2.4. The token bucket has a default maximum depth of at least MTU_SIZE bytes. The credits[2] in the token bucket are incremented by *incSize* at every interval time *t* and decreased by *decSize* whenever a packet transmitted out. The value of *incSize* is the product of interval time *t* multiplied by allocation rate of the offered traffic. The value of *decSize* is the length of packet in byte.

When a packet is waiting for access to the ring at the head of a queue, it is granted ring access only if there are at least *lowLimit* (MTU_SIZE) bytes of credits in the token bucket. The number of credits in the token bucket is then reduced by the length of the packet transmitted, *decSize*. The credits are reduced as each byte is sent, not all at once. It

[1] Although multiple shapers are used within this thesis, the behavior of all shapers can be characterized by a common algorithm with instance-specific parameters.
[2] The term "credit" is a substitute of token in this variation.

is never possible for the number of credits to become negative after subtracting the number of bytes in the transmitted frame. The maximum number of credits that can accumulate in the token bucket depends on the shaper.

Crossing below the *lowLimit* threshold shall generate a rate-limiting indication, so that offered traffic can stop before reaching zero credits, where excessive transmissions are rejected. To bound the burst traffic after inactivity intervals, when no packets are ready for transmission, credits are reduced to *lowLimit* (if currently higher than *lowLimit*) and can accumulate to no more than *lowLimit*. For all shapers described in this variation, *lowLimit* is set to MTU_SIZE in order to allow the transmission of a full sized frame without reducing the credits below zero.

The *hiLimit* threshold limits the positive credits, to avoid overflow. When packets are ready for transmission (and are being blocked by transit traffic), credits can only be accumulated up to *hiLimit*. The *hiLimit* value shall be at least MTU_SIZE. If it is set to exactly MTU_SIZE, then no bursts are allowed.

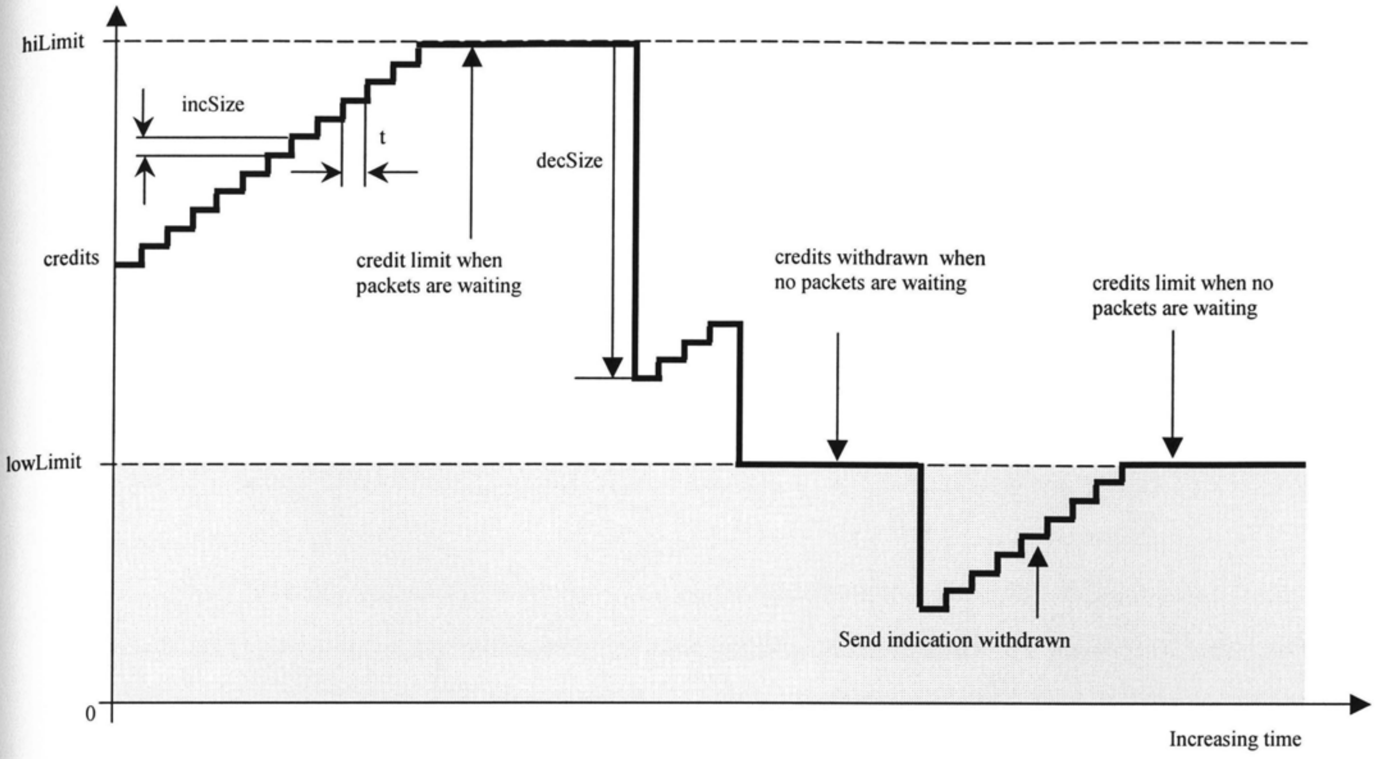

hiLimit
incSize
t
decSize
credits
credit limit when packets are waiting
credits withdrawn when no packets are waiting
credits limit when no packets are waiting
lowLimit
Send indication withdrawn
0
Increasing time

Figure 2.4: RPR Token Bucket Traffic Shaper

RPR variation of TBTS differs from standard TBTS in the following aspects:

- Terminology of counter: Standard TBTS uses "token" as the terminology to describe the counter. RPR variation TBTS uses "credit."
- Prerequisite for sending traffic: Standard TBTS allows packet to be transmitted out as long as there is a token in the token bucket. However, in RPR variation, a packet can be transmitted out only when credits accumulated in the bucket between *lowLimit* threshold and *hiLimit* threshold.
- RPR variation regulates traffic more smoothly than standard TBTS. When there is no packets waiting in the buffer after a few inactivate intervals, RPR variation either withdraws credits to *lowLimit* threshold if higher than it or accumulates no more than *lowLimit* if below it. In standard TBTS, the token continues to increase until bucket is full. In the worst case scenario, when two packet with the length of

MTU_SIZE arrive in the buffer sequentially, RPR variation maximizes transmitting one packet at a time. Standard TBTS, on the other hand, sends two packets if there is enough tokens in the bucket.

2.4.4.3 Inside Rate Control Unit

Recall Figure 2.2. Having explained the overall structure of Rate Control Unit, we are now able to show how this unit works and how it controls the different types of traffic. Figure 2.5 illustrate the architecture of this unit:

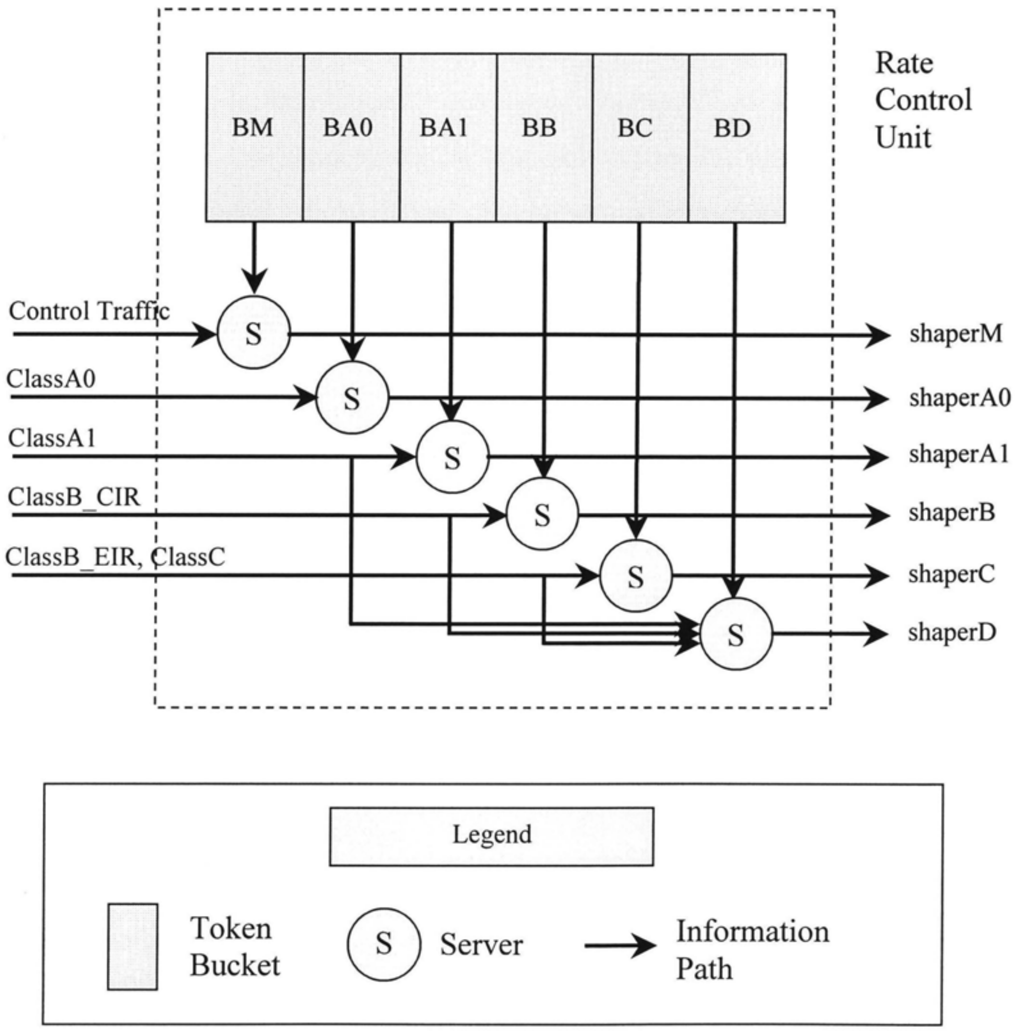


Figure 2.5: Enlarged Rate Control Unit

In this figure, there are six token buckets corresponding to six shapers to regulate five types of traffic. Tokens in bucket BM regulates RPR MAC supplied control traffic and gets the result *shaperM*, tokens in bucket BA0 regulates ClassA0 traffic and gets the result shaperA0, and so on. Each type of traffic passes through its own token packet traffic shaper and gets a result for its own type of traffic, meanwhile ClassA1, ClassB and ClassC traffic need to pass through a additional shaper to constrain the client to sustain the downstream allocated subclassA0 traffic. These results are fed into Arbiter for the decision-making process.

2.4.4.4 Shapers

There are five shapers implemented in this thesis: *shaperM, shaperA, shaperB, shaperC, shaperD*. *ShaperM* is the shaper of RPR control traffic, for example, fairness control packet. *ShaperA, shaperB, shaperC* are the shapers of the service classes accordingly. *ShaperD* is an additional shaper to constrain the client to sustain the downstream allocated subclassA0 traffic.

2.4.4.4.1 Control Shaper – shaperM

The control shaper limits the MAC-supplied control traffic to its allocated limits, it runs in addition to the other shapers through which control packets pass (i.e. shaperA, shaperB, shaperC), therefore, implemention should account for this by setting the rates of shaperA, shaperB and shaperC such that they include enough bandwidth for the anticipated control traffic. The parameters of shaperM are illustrated in Table 2.11.

Parameters	*Value*	*Explanation*
decSize	MacA	Length of MAC supplied control packet.
incSize	rateM * time	Allowed rate for control packet multiply by the unit of time since the last increment.
hiLimitM	2 * sizeMTU	Default value for hiLimitM is two MTU size.
lowLimitM	sizeMTU	Default value for lowLimitM is one MTU size.

Table 2.11: shaperM parameters

The MAC-supplied control traffic is allowed to be transmitted out only when the value of shaperM is TRUE, which is defined in Equation 2.3 and Equation 2.4.

creditM = MIN(hiLimitM, creditM + rateM * time) (E2.3)

shaperM = (creditM >= lowLimitM) (E2.4)

Here creditM is the minimum of hiLimitM and last creditM plus incSize, (rateM * time). ShaperM is TRUE when creditM greater than or equal to lowLimitM; otherwise FALSE.

2.4.4.4.2 ClassA Shaper – shaperA0 and shaperA1

The ClassA shaper limits the client-supplied ClassA transmissions. ShaperA is further divided into shaperA0 and shaperA1. The parameters of shaperA0 and shaperA1 are illustrated in Table 2.12 and Table 2.13 respectively.

Parameters	*Value*	*Explanation*
decSize	ClientA0	Length of client-supplied subclassA0 packet.
incSize	rateA0 * time	Allocated rate for subclassA0 packet multiply by the unit of time since the last increment.
hiLimitA0	SizeMTU+ rateA0*MAX_JITTER/2	Amount of credits that would be needed to buffer the amount of traffic that could be generated by MAX_JITTER, plus one MTU to account for *lowLimitA0*.
lowLimitA0	sizeMTU	Default value for lowLimitA0 is one MTU size.

Table 2.12: shaperA0 parameters

The default value for hiLimitA0 is set to preserve the delay and jitter guarantees of subclassA0 traffic. The default value is given by Equations 2.5 and 2.6.

hiLimitA0 = sizeMTU + rateA0 * MAX_JITTER / 2 (E2.5)

MAX_JITTER = (numStations*(sizeMTU/LR)) / (1 – rateA0/LR) (E2.6)

Here numStations is the number of nodes in the ring, LR is the link rate. MAX_JITTER is determined by the possible log jam due to coincidental submission of subclassA0 packets by all the nodes in the ring, (numStations*(sizeMTU/LR)), and the rate at which the node can recover, (1-rateA0/LR).

Table 2.13 shows the parameters of shaperA1:

Parameters	*Value*	*Explanation*
decSize	clientA1	Length of client-supplied subclassA1 packet.
incSize	rateA1 * time	Allocated rate for subclassA1 packet multiply by the unit of time since the last increment.
hiLimitA1	sizeMTU+ rateA1*MAX_JITTER/2	Amount of credits that would be needed to buffer the amount of traffic that could be generated by MAX_JITTER, plus one MTU to account for *lowLimitA1*.
lowLimitA1	sizeMTU	Default value for *lowLimitA1* is one MTU size.

Table 2.13: shaperA1 parameters

Parameters of shaperA1 are similar to those of shaperA0, except transmission rate is replacing with "rateA1".

The Class A traffic is allowed to transmit out only when the value of shaperA is TRUE, which is defined in Equation 2.7, 2.8 and 2.9.

CreditA0 = MIN(hiLimitA0, creditA0 + rateA0 * time) (E2.7)

CreditA1 = MIN(hiLimitA1, creditA1 + rateA1 * time) (E2.8)

ShaperA = (creditA0>=lowLimitA0 || (creditA1>=lowLimitA1 && shaperD) (E2.9)

Here creditA0 is the minimum of hiLimitA0 and last creditA0 plus incSize, (rateA0 * time), creditA1 is the minimum of hiLimitA1 and last creditA1 plus incSize, (rateA1 * time), ShaperA is TRUE when creditA0 is greater than or equal to lowLimitA0, or creditA1 greater than or equal to lowLimitA1 and shaperD is TRUE; otherwise shaperA is FALSE.

2.4.4.4.3 ClassB Shaper – shaperB

The ClassB shaper limits the client-supplied ClassB transmissions. The parameters of shaperB are illustrated in Table 2.14.

Parameters	*Value*	*Explanation*
decSize	ClientB	Length of client-supplied ClassB CIR packet.
incSize	rateB * time	Allocated rate for ClassB CIR packet multiply by the unit of time since the last increment.
hiLimitB	sizeMTU + (rateA1 + rateB) * MAX_JITTER/2	Amount of credits that would be needed to buffer the amount of traffic that could be generated by MAX_JITTER, plus one MTU to account for *lowLimitB*.
lowLimitB	sizeMTU	Default value for lowLimitB is one MTU size.

Table 2.14: shaperB parameters

The Class B traffic is allowed to transmit out only when the value of shaperB is TRUE, which is defined in Equation 2.10 and 2.11.

CreditB = MIN(hiLimitB, creditB + rateB * time) (E2.10)

ShaperB = ((creditB >= lowLimitB) && shaperD) (E2.11)

Here creditB is the minimum of hiLimitB and last creditB plus incSize, (rateB * time), shaperB is TRUE when creditB is greater than or equal to lowLimitB and shaperD is TRUE; otherwise shaperB is FALSE.

2.4.4.4.4 Fairness Eligible Shaper – shaperC

The fairness eligible shaper limits the client-supplied ClassB-EIR and ClassC transmissions. The parameters of shaperC are illustrated in Table 2.15.

Parameters	*Value*	*Explanation*
decSize	clientC	Length of client-supplied ClassB-EIR and ClassC packet.
incSize	MAX_ALLOWED_RATE * time	Maximum allowed rate for fairness eligible packet multiply by the unit of time since the last increment.
hiLimitC	2 * sizeMTU	Default value for hiLimitC is two MTU size.
lowLimitC	sizeMTU	Default value for lowLimitC is one MTU size.

Table 2.15: shaperC parameters

The ClassB-EIR and ClassC traffic is allowed to transmit out only when the value of shaperC is TRUE, which is defined in Equation 2.12 and 2.13.

$$\text{creditC} = \text{MIN}(\text{hiLimitC}, \text{creditC} + \text{MAX\_ALLOWED\_RATE} * \text{time}) \quad \text{(E2.12)}$$
$$\text{shaperC} = ((\text{creditC} >= \text{lowLimitC}) \ \&\& \ \text{shaperD}) \quad \text{(E2.13)}$$

Here creditB is the minimum of hiLimitB and last creditB plus incSize, (rateB * time), shaperB is TRUE when creditB is greater than or equal to lowLimitB and shaperD is TRUE; otherwise shaperB is FALSE.

Here creditC is the minimum of hiLimitC and last creditC plus incSize, (MAX_ALLOWED_RATE * time), this allows fairness eligible traffic to be bursty over the short term, while the fairness algorithm limits it to *allowed_rate* over a longer time periold. ShaperC is TRUE when creditC is greater than or equal to lowLimitC and shaperD is TRUE; otherwise shaperC is FALSE.

2.4.4.4.5 Downstream Shaper – shaperD

The downstream shaper monitors the transit traffic to ensure sufficient levels of sustainable subclassA0 traffic for downstream nodes. The parameters of shaperD are illustrated in Table 2.16.

Parameters	*Value*	*Explanation*
decSize	!ClientA0	Length of client-supplied non-sublassA0 packet.
incSize	*unreserved_rate* * time	Unreserved rate for non-subclassA0 packet multiply by the unit of time since the last increment.
hiLimitD	2 * sizeMTU	Default value for hiLimitD is two MTU size.
lowLimitD	sizeMTU	Default value for lowLimitD is one MTU size.

Table 2.16: shaperD parameters

Non-subclassA0 add traffic is allowed to transmitted out only when the value of shaperD is TRUE, which is defined in Equation 2.14 and 2.15.

CreditD = MIN(hiLimitD, creditD + *unreserved_rate* * time) (E2.14)

ShaperD = (creditD >= lowLimitD) (E2.15)

Here creditD is the minimum of hiLimitD and last creditD plus incSize, (*unreserved_rate* * time), shaperD is TRUE when creditD is greater than or equal to lowLimitD; otherwise shaperD is FALSE.

2.5 Verification

In this section, a few testing scenarios are created in OPNET to compare the simulation result of the in-house RPR model with the SRP model that comes with OPNET. In our model, we use OC-12 as link type with the maximum bandwidth up to 622 Mbps, packets generated from each node is in fixed size, 12000 bits. The delay in each link is set at a fixed value, 70 ns, and reserved rate is 2% of total bandwidth, each node carries the same weight.

2.5.1 Scenario 1 – Verify Fairness Algorithm

Let's compare the simulation result generated from Cisco's SRP model and that of the in-house OPNET Model to verify fairness algorithm implemented. The topology of this scenario is that each node, from Node 1 through Node 3 inclusive, sends traffic at the constant rate of 600 Mbps to Node 4, however each node start generating the traffic at different time as shown in Figure 2.6.

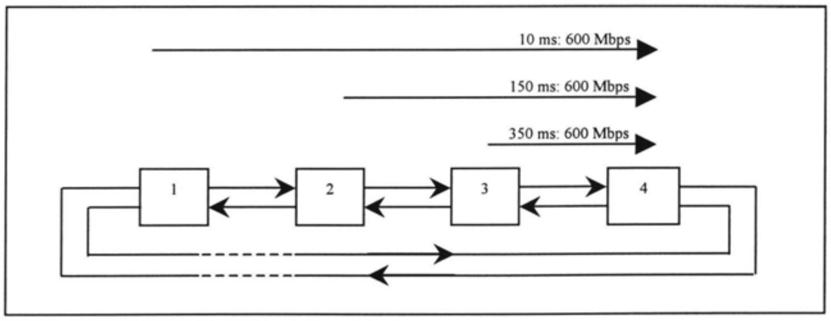


Figure 2.6: Verify Fairness Algorithm

Theoretically, Node 1 should generate 600 Mbps traffic from 10 ms to 150ms, Node 1 and Node 2 should generate 622 / 2 =311 Mbps traffic from 150 ms to 350 ms and Node 1 through Node 3 should generate 622 / 3 = 207 Mbps traffic from 350 ms and later.

2.5.1.1 Cisco's SRP Model

In this test I adopted Cisco's SRP model in OPNET and sampled 200 values for each node in 450 ms simulation time, the simulation result is shown in Figure 2.7.

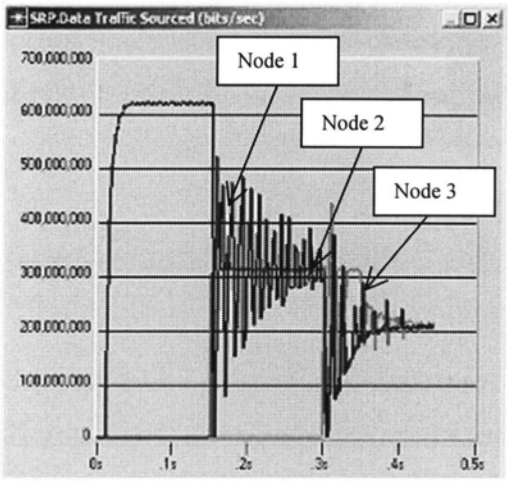


Figure 2.7: Simulation Result of SRP Model

From the result one can see that Node 1 generated 600 Mbps traffic from beginning of the simulation to 150 ms, then generated 300 Mbps in average after Node 2 joined in. Node 2 also generated a little bit more than 300 Mbps traffic until Node 3 joined in, which in turn made each node in the ring to generate 200 Mbps traffic. At the steady state, it provides the desirable fairness to each node. The result meets the theory therefore we can say it is correct.

2.5.1.2 Our Model in Conservative Mode

Let's run our model in conservative mode where we sampled 200 values for each node in 450 ms simulation time. The result is shown in Figure 2.8.

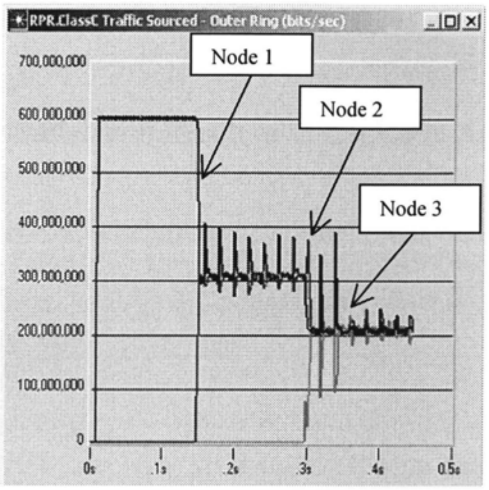


Figure 2.8: Simulation Result of Our Node in Conservative Mode

From the result one can see the bandwidth of the link is fairly shared among the node in their steady state, which meets the theory of RPR, therefore we can say the fairness algorithm implemented in conservative node in our simulation model is correct.

2.5.1.3 Our Model in Aggressive Mode

Let's run our model in aggressive mode where we sampled 200 values for each node in 450 ms simulation time. The result is shown in Figure 2.9.

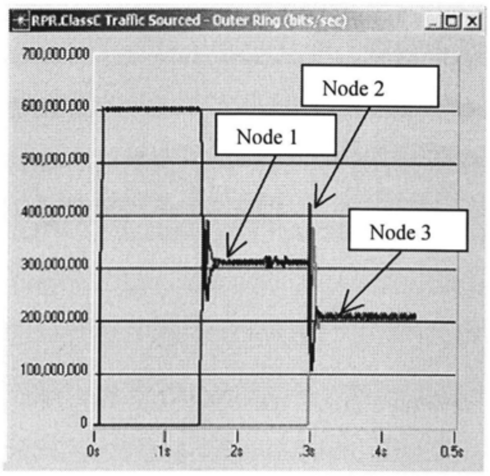


Figure 2.9: Simulation Result of Our Node in Aggressive Mode

From the result one can observe the behavior of all the nodes meets the theory.

2.5.2 Scenario 2 – Verify Spatial Reuse

Cisco's SRP Model in OPNET can only send traffic to one destination, which does not meet the prerequisites of this scenario. Therefore, let's compare the simulation result generated from our model in OPNET with theoretical result to verify spatial reuse properties implemented in our model. The topology of this scenario is the same as the previous one. Moreover, Node 1 generates 200 Mbps extra amount of traffic to Node 2 starting at 10 ms to take the advantage of the unused bandwidth. The topology is shown in Figure 2.10.

Theoretically Node 2 and Node 3 should behave the same way as in previous scenario, however, Node 1 utilizes the spatial reuse properties in RPR to use the leftover bandwidth: it generates the faired-share rate allocated for it plus up to whatever leftover in each link, therefore, theoretically it generates full line rate from 10 ms to 150 ms and 500 Mbps from 150 ms to 350 ms and 400 Mbps from 350 ms to the end of the simulation.

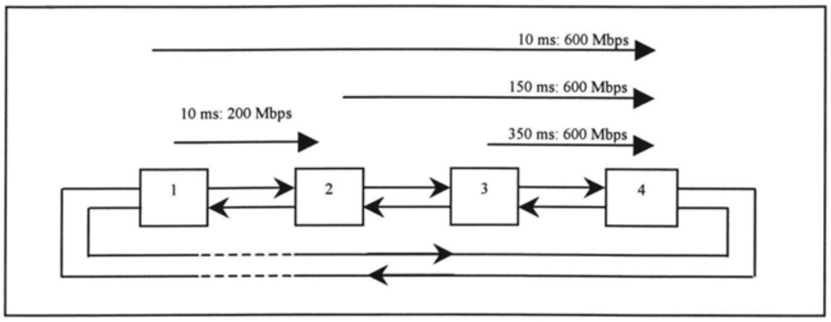


Figure 2.10: Verify Spatial Reuse

2.5.2.1 Our Model in Conservative Mode

Let's run our model in conservative mode where the total simulation time is 450 ms, the simulation result is shown in Figure 2.11.

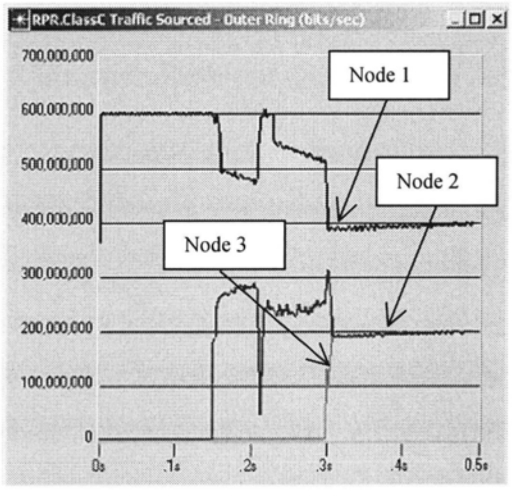


Figure 2.11: Simulation Result of Our Model in Conservative Mode

From the result one can see that Node 1 generated 600 Mbps traffic from beginning of the simulation to 150 ms, then generated 500 Mbps in average after Node 2 joined in, from 350 ms to the end of simulation it generated 400 Mbps traffic. Node 2 generated around 300 Mbps traffic in average until Node 3 joined in, which in turn made Node 2 and Node 3 generate 200 Mbps traffic. This result meets the theory.

2.5.2.2 Our Model in Aggressive Mode

Let's run our model in aggressive mode where the total simulation time is 450 ms, the simulation result is shown in Figure 2.12.

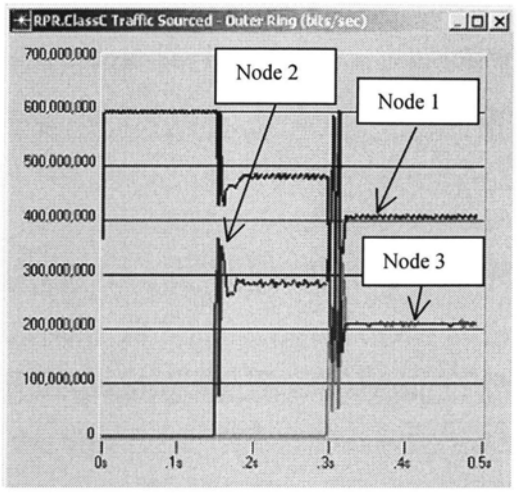


Figure 2.12: Simulation Result of Our Model in Aggressive Mode

From the result one can observe that the simulation result closely matches the theoretical result.

Chapter 3: Fuzzy Logic Control

3.1 Introduction

IEEE 802.17 RPR Darwin Proposal [1] introduced a solution to control the traffic sent from each node participating in the ring. There are two algorithms provided in this solution, Aggressive algorithm and Conservative algorithm. Aggressive algorithm is faster in convergency but introduces higher oscillation. Conservative algorithm, on the other hand, is a little bit slower but has less oscillation. Figure 3.1 illustrates a scenario that node 1 through node 16 send 38 Mbps constant traffic to the destination node, node 0. All the nodes in the ring are running Aggressive algorithm and the bandwidth of each ring is 622Mbps. Because the total traffic sent by the nodes is 38Mbps × 16 = 608Mbps, which is less than the maximum bandwidth of the link, no node is congested. Under this situation, RPR performs well. Essentially, the bandwidth is equally shared among the nodes. That is, each node has a throughput of 38 Mbps. Figure 3.2 shows the throughput of node 1. As can be seen, the throughput is very stable, practically no oscillation occurs.

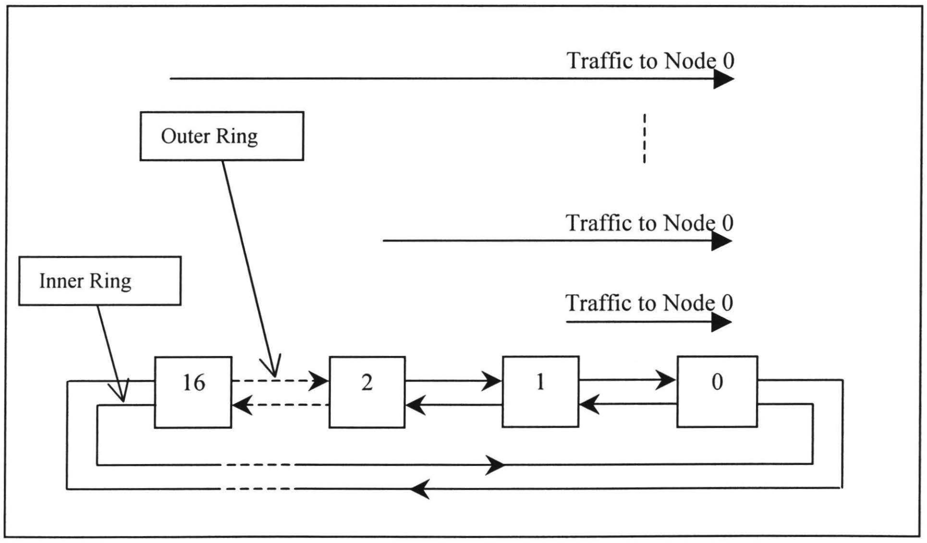


Figure 3.1: Multi Sourced Traffic

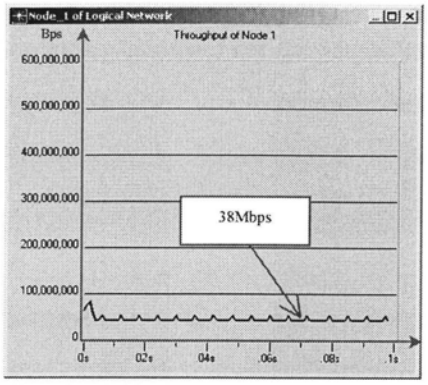


Figure 3.2: Throughput of Node 1 at a Constant Rate

Under the congestion situation, however, RPR with aggressive or conservative algorithm does not perform well. For example, if traffic generated from each node in the example above is increased from 38 Mbps to 42 Mbps, congestion occurs and it causes the oscillation in throughput. The throughput of each node should be 622 Mbps / 16 $\approx$ 39 Mbps. However, because of the nature of the RPR fairness algorithm [1], the throughput of the congestion node (Node 1), would have large oscillations as shown in Figure 3.3a, others would have less oscillation as shown in Figure 3.3b. Note that the throughputs presented here are averaged in every 10 aging_interval[1], which is equal to 1 msec. The reason for measuring the throughput over every 10 againg_interval is that it gives a more accurate indication of the oscillation problem.

In aggressive mode, the congestion node (Node 1) and the furthest upstream node (Node 16) in the congestion domain experience more oscillation than any other nodes in the same domain [3]. The large degree of oscillation at the congested node is the nature of the aggressive mode whereas the large degree of oscillation at the furthest upstream node is due to the occasional advertisements of FULL_RATE by the second furthest upstream node. We have modified the mechanism of the fair rate propagation in the original aggressive mode algorithm to eliminate the occasional advertisements of FULL_RATE sent by the second furthest upstream node. As a result, the furthest upstream node has

[1] Aging _interval for OC-3 is 400 us, for OC-12 and higher is 100 us.

similar behavior as the other nodes in the same congestion domain (except the congestion node) in all of our simulation results. See [3] for more details on the modification.

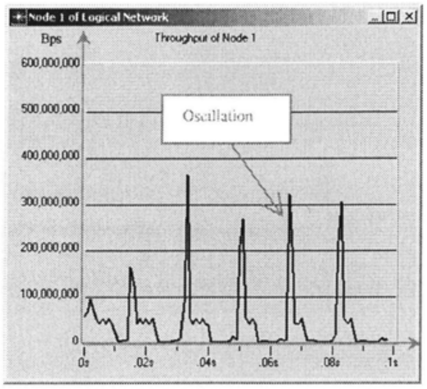


Figure 3.3a: Throughput of Node 1

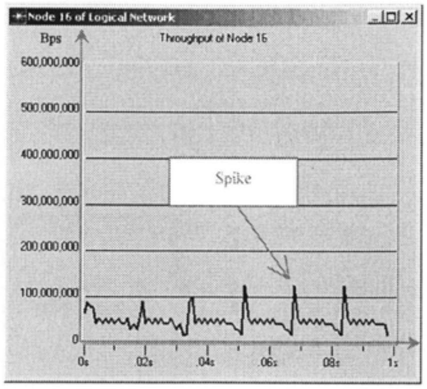


Figure 3.3b: Throughput of Node 16

Oscillation brings many drawbacks, for example, the unfair allocation of bandwidth between different nodes, unstable performance of the system, reduced quality of service, etc. It can even bring the system temporarily out of service.

3.1.1 The Architecture Of Traffic Control System

Figure 3.4 illustrates a simplified architecture of the traffic control system[1]. According to IEEE 802.17 RPR Darwin Proposal [1], there are two queues holding the traffic in the node, Transmit Queue and Transit Queue. Transmit Queue holds the traffic generated from the node itself, Transit Queue holds the traffic bypassing the node. Add_rate is the counter that counts the amount of data sending out of the Transmit Queue during each aging_interval, fw_rate counts the traffic sending out of the Transit Queue. The value of add_rate and fw_rate are zeros at the beginning of each aging_interval and are reset to zeros at the end of each aging_interval. Also at the end of each aging_interval, they are input to a low pass filter in Rate Control Unit to calculate lp_add_rate and lp_fw_rate, respectively, which, in turn, is used to calculate the loca_fair_rate and shaper_add_rate. Local_fair_rate is fed into Fairness Control Unit to become a candidate for advertise_rate, which are injected into fairness advertisement packet and are sent to the upstream node by Arbiter. Note that when advertise_rate is received by its upstream

[1] To simplify the illustration, only show the low priority queues in the diagram is shown.

neighbor, it is then called rcvd_rate and becomes another candidate for advertise_rate. Shaper_add_rate is fed into Arbiter to control the output speed of Transmit Queue and Transit Queue. Rcvd_rate is also fed into Rate Control Unit to participate in calculating shaper_add_rate.

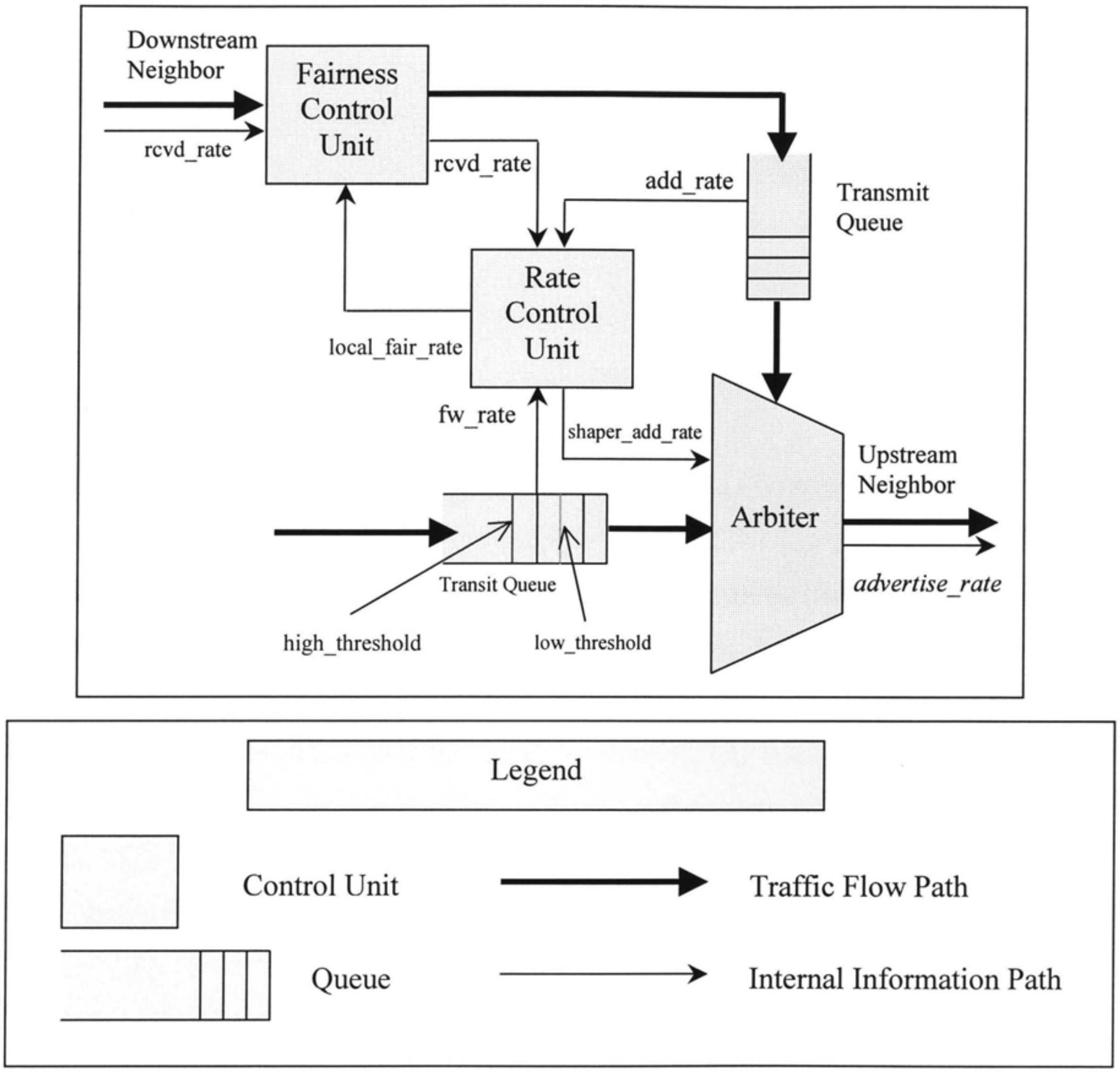


Figure 3.4: Simplified Architecture of the Traffic Control System

This control system is working well in most scenarios, however, in some congestion situations, it introduces oscillation as shown in Figure 3.3.

3.1.2 Oscillation Analysis

Arbiter uses shaper_add_rate to control the outgoing traffic in Transmit Queue and Transit Queue. Before it sends any packet, the scheduler checks shaper_add_rate and the traffic accumulated in these two queues first, and then decides which queue to send traffic based on the following decision-making steps:

1) If the traffic accumulated in the Transit Queue is higher than STQ_SIZE - MTU, Arbiter sends a packet from Transit Queue;
2) Otherwise, if there is any packet in the Transmit Queue, Arbiter sends a packet from Transmit Queue;
3) Otherwise, if there is any packet in the Transit Queue, Arbiter sends a from Transit Queue;
4) Otherwise, if there is no packet in Transit Queue, system goes to idle.

When a node is congested, it advertises the advertise_rate to its upstream nodes. This causes the upstream nodes to throttle down their data rates. By doing this, the congestion in the congested node gradually drops down. Once the congestion is temporarily eliminated, the previous congested node advertises a FULL_RATE to its upstream nodes again. This causes the upstream nodes to increase their data rate again and eventually causes another congestion. The above cycle is repeated as illustrated by the large oscillation of throughput in each node.

3.1.3 Design Task

The goal of this research is to derive a close-to-ideal mechanism that can reduce the oscillation, thus allocate bandwidth fairly among the nodes, stabilizing the traffic and eventually improving the performance of the system. Fuzzy Logic Control (FLC) is used to achieve this goal. In the next section the implementation of fuzzy logic is discussed in details.

3.2 Implementation of Fuzzy Logic Control in RPR

3.2.1 Overview

Fuzzy Logic Control (FLC) can be viewed as an alternative, non-conventional way of designing effective feedback control without relying on formal models of the controlled system and control theoretic tools. FLC has been applied successfully to the task of controlling systems for which analytical models are not easily obtainable or the model itself, if available, is too complex and highly nonlinear. Figure 3.5 is an outlined process of FLC.

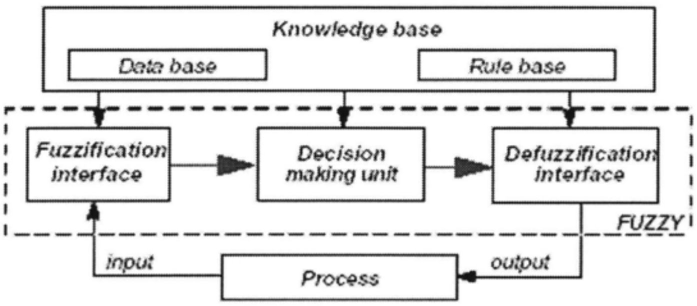
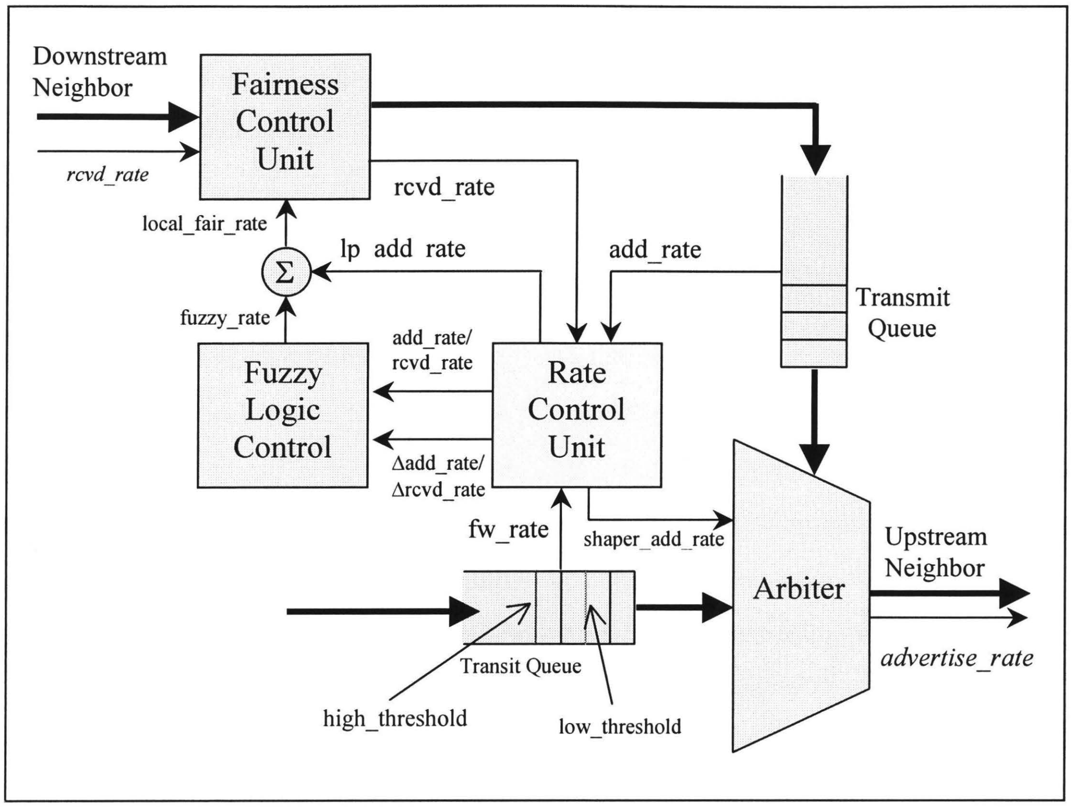


Figure 3.5: An Outlined Process of FLC

3.2.2 Architecture of FLC

In order to minimize the oscillation, one needs to fine-tune the key control value, local_fair_rate. Therefore, the proposed FLC algorithm concentrates on fine-tuning local_fair_rate. The FLC unit has two groups of inputs, add_rate and Δadd_rate is in one group, rcvd_rate and Δrcvd_rate is in another group, and one output, fuzzy_add_rate. When rcvd_rate is FULL_RATE, FLC uses add_rate and Δadd_rate as input parameters, otherwise usees rcvd_rate and Δrcvd_rate as input parameters. Depending on the situation of the traffic accumulated in Transmit Queue and Transit Queue, the system uses FLC to accelerate or decelerate in sending the traffic intelligently, thus, reducing the oscillation and improving the performance of all the nodes in the ring. Figure 3.6 shows the simplified architecture of FLC in RPR. It is similar to the one illustrated in Figure 3.4, except a FLC unit is added.



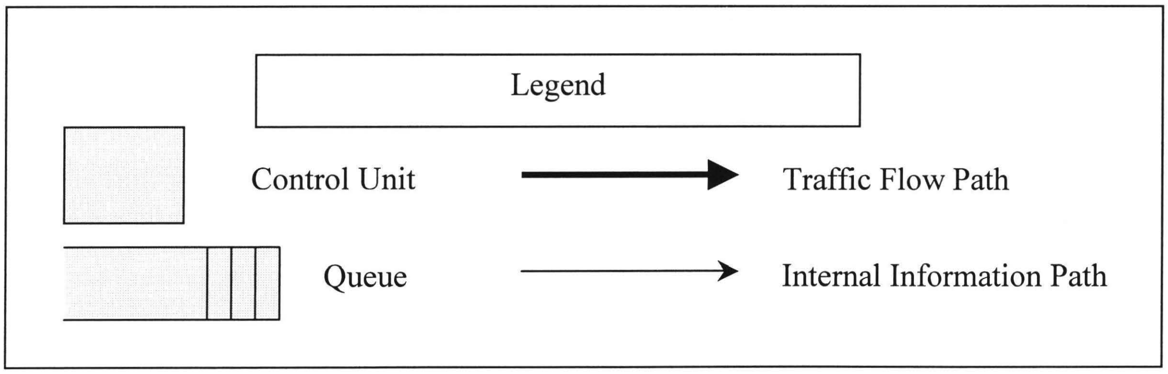


Figure 3.6: Traffic Control System With FLC

FLC algorithm is ignited when the system detects the congestion, which must meet at least one of the following two conditions:

- lp_nr_xmit_rate >= unreserved_rate (E3.1)
- STQ_depth >= low_threshold (E3.2)

Here lp_nr_xmit_rate = lp_add_rate + lp_fw_rate, STQ_depth is the traffic accumulated in STQ.

Then, FLC processes the inputs and gets the output, fuzzy_add_rate, which uses the following formula to generate a new result for local_fair_rate:

$$\text{local\_fair\_rate} = \text{lp\_add\_rate} + \text{fuzzy\_add\_rate} \tag{E3.3}$$

3.2.3 Membership Function (MF)

MF is used to determine how much degree the system is on in a given state. There are many MFs available, trapmf, gaussmf, sigmf, etc. Because RPR is a high-speed switching protocol implemented in MAC layer, timing is a critical issue here, therefore selecting a simple but effective MF becomes a major concern to the success of the implementation. In this version, a simple triangle membership function (trimf) is selected. In Figure 3.7, add_rate is used as an example to demonstrate how MF works.

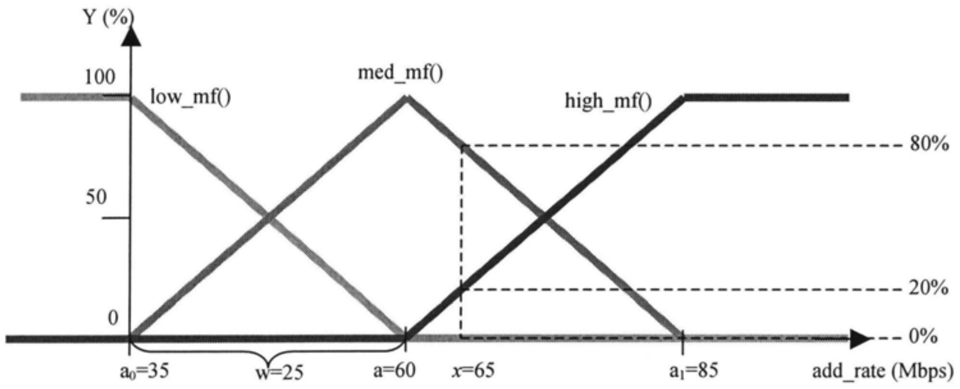


Figure 3.7: Triangle Membership Function (trimf) of add_rate

As it can be seen in Figure 3.7, horizontal axis represents add_rate in Mbps, vertical axis represents the output of membership functions in percentage, *a* is the ideal rate we are trying to achieve, $[a_0, a_1]$ is the range that the fuzzy works within, *w* is half of the range size. The input of the function, *x*, is the add_rate in a certain aging_interval.

Here *a* plays a key role in the success of FLC. Node itself doesn't know what rate is the ideal rate without getting the feedback from other nodes. Therefore, we use rcvd_fair_rate as the estimated value of *a* if rcvd_fair_rate is not FULL_RATE; otherwise last interval's local_fair_rate is used as the estimated value of *a*.

There are three membership functions, low_mf(), med_mf() and high_mf() as illustrated in E3.4, E3.5 and E3.6 that correspondes to three states: low_speed state, med_speed state and high_speed state. For a given add_rate *x*, the output of the three membership functions corresponding to *x* can be computed. They represent the ratio of *x* in these states. For example, as illustrated in Figure 3.7, add_rate *x* of 65 Mbps gives 0% in low_speed state, 20% in high_speed state and 80% in med_speed state.

$$Y_1 = \begin{cases} 1, & x < a_0 \\ \dfrac{a - x}{w}, & x \geq a_0 \cap x < a \\ 0, & x \geq a \end{cases} \qquad \text{(E3.4)}$$

$$Y_2 = \begin{cases} 0, & x < a_0 \cup x \geq a_1 \\ \dfrac{x - a_0}{w}, & x \geq a_0 \cap x < a \\ \dfrac{a_1 - x}{w}, & x \geq a \cap x < a_1 \end{cases} \qquad \text{(E3.5)}$$

$$Y_3 = \begin{cases} 0, & x < a \\ \dfrac{x - a}{w}, & x \geq a \cap x < a_1 \\ 1, & x \geq a_1 \end{cases} \qquad \text{(E3.6)}$$

Here *x* is the add_rate at any given time, a_0 is $a - w/2$, a_1 is $a + w/2$.

To achieve the best result from FLC, the range must be selected carefully. It should be neither too wide, otherwise a large degree of oscillation likely occurs; nor too narrow, otherwise the system has less chance to correct oscillation. Our results show that there are no significant differences in performance when *w* is within the range from around 20Mbps to 40Mbps for the 622Mbps ring. In this thesis, 25Mbps for *w* is used.

The MFs of Δadd_rate is illustrated in Figure 3.8, which is similar to that of add_rate.

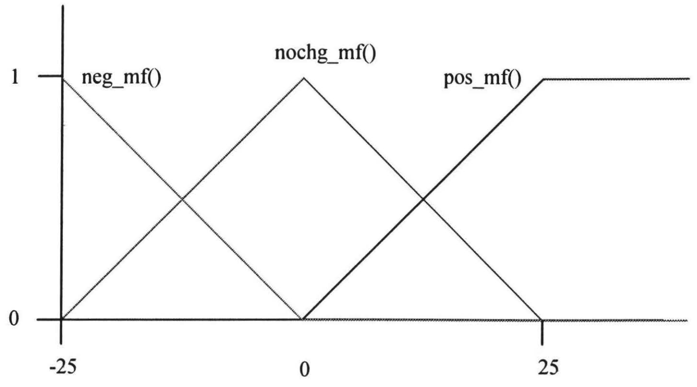


Figure 3.8 Triangle Membership Function (trimf) of Δadd_rate

3.2.4 If-Then Rules and Defuzzify

If-Then rules is used to determine how FLC is defuzzified. There are five If-Then rules applied in FLC, as shown in Table 3.1:

Rule id	*Explanation*	*WEIGHT*	*Applied MF*	*Output Y*
1	If add_rate is low, then Transmit Queue fast increase sending traffic.	1.0	low_mf()	Y_1
2	If add_rate is medium and Δadd_rate is negative, then Transmit Queue slowly increase sending traffic.	1/4	neg_mf()	Y_2
3	If add_rate is medium and Δadd_rate is no_change, then Transmit Queue keeps sending traffic as in last aging_interval.	0	nochg_mf()	Y_3
4	If add_rate is medium and Δadd_rate is positive, then Transmit Queue slowly	-1/4	pos_mf()	Y_4

	decease sending traffic.			
5	If add_rate is high, then Transmit Queue fast decrease sending traffic.	-1.0	high_mf()	Y_5

Table 3.1: If-Then Rules

The weight assigned to each rule corresponds to the degree of changes in sending rate for the transmit queue. For example, because rule 1 and rule 5 need larger changes, so more weight to Y_1 and Y_5 is assigned.

Defuzzify is a process that uses applicable rule and the results of MF as input to get the output of FLC. First, let's assume the add_rate $x = 65$ Mbps and Δadd_rate $\Delta x = -6.25$ Mbps, then, one can get the following outputs from different MFs:

- $Y_1 = \text{low\_mf}(x = 65) = 0.0$
- $Y_2 = \text{neg\_mf}$[1]$(\Delta x = -6.25) = 0.25$
- $Y_3 = \text{nochg\_mf}(\Delta x = -6.25) = 0.75$
- $Y_4 = \text{pos\_mf}(\Delta x = -6.25) = 0.0$
- $Y_5 = \text{high\_mf}(x = 65) = 0.2$

Then, the following formula is used to get the result of FLC:

$$\text{fuzzy\_add\_rate} = \frac{W \times \sum_{i=1}^{n} \{Yi \times WEIGHTi\}}{\sum_{i=1}^{n} Yi} \qquad \text{(E3.7)}$$

n is the total number of rules, Yi is the output of MF for rule i, $WEIGHTi$ is the weight for rule i and the fuzzy_add_rate is the result of FLC. Therefore,

[1] We didn't give the explanation of Δadd_rate and its MFs, so here we just use 0.25 to illustrate how defuzzify works. Same for nochg_mf() and pos_mf().

$$\text{fuzzy\_add\_rate} = \frac{25Mbps \times [0.0 \times 1 + 0.25 \times \frac{1}{4} + 0.75 \times 0 + 0.0 \times (-\frac{1}{4}) + 0.2 \times (-1)]}{0.0 + 0.25 + 0.75 + 0.0 + 0.2}$$

$$= -2.86 \text{ Mbps}$$

From the result, we can see when the add_rate x = 65 Mbps and Δadd_rate Δx = -6.25 Mbps, add_rate decreases 2.86 Mbps traffic at the next aging_interval according to the FLC.

There are many methods to defuzzify the Fuzzy Logic. The above method is chosen due to its simple method. The result of FLC is used to fine-tune the traffic in the next aging_interval.. A sample output of local_fair_rate is shown in Figure 3.9, as one can see the oscillation is alleviated.

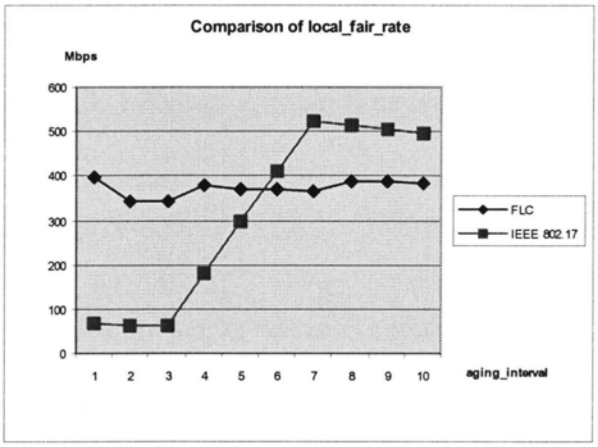


Figure 3.9: Comparison of local_fair_rate

3.3 Case Study

In this chapter, a few testing scenarios are created in OPNET to compare the performance of Fuzzy Logic Control with that of Aggressive and Conservative Algorithms. To simplify the system, OC-12 is used as link type with the maximum bandwidth up to 622 Mbps, packets generated from each node with fixed size, 12000 bit. The delay in each link is set at a fixed value, 70 ns, and reserved rate is 2% of total bandwidth, each node carries the same weight. In FLC algorithm, A is last interval's local_fair_rate if rcvd_fair_rate is FULL_RATE; otherwise it is rcvd_fair_rate.

3.3.1 Scenario 1 – 16 Nodes Simple Sourced Traffic Scenario

In this scenario, each node from Node 1 through Node 16 inclusive sends traffic at the constant rate of 42 Mbps to Node 0 as shown in Figure 3.10.

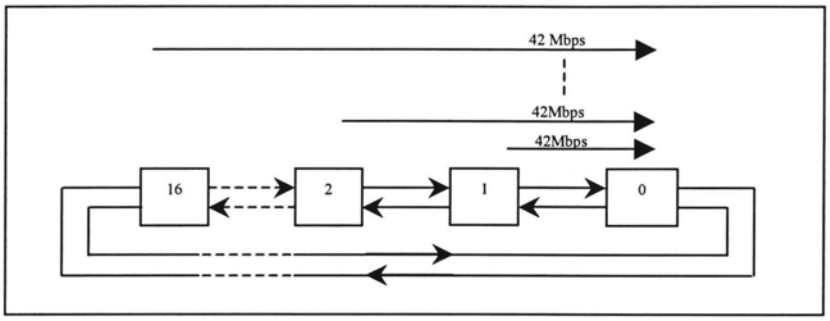
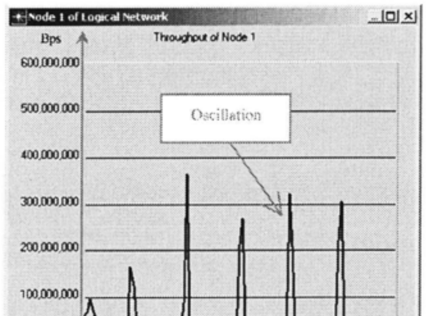


Figure 3.10: 16 Nodes Simple Sourced Traffic Scenario

3.3.1.1 Aggressive Mode

In this scenario, all the nodes are set to run in Aggressive mode. The result is shown in Figure 3.11. Only the throughput of Node 1 is shown, the closest node to the destination (Node 0), and Node 16, the farthest node to the destination. The rest of the nodes behave similar to Node 16.



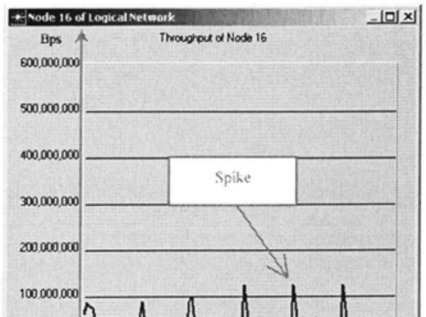


Figure 3.11: Simulation Results of Node 1 and Node 16 in Aggressive Mode

From the result one can see that Node 1 has huge oscillation, Node 16 and others have lesser degree of oscillation than Node 1, but it is still considered to be high.

3.3.1.2 Conservative Mode

Cisco's Spatial Reuse Protocol fairness algorithm (SRP-fa) implemented in OPNET DPT model is running in conservative mode. An exactly the same scenario as the one mentioned above is designed using OPNET DPT model to compare the result, which is shown in Figure 3.12.

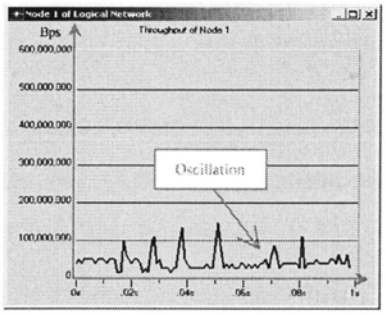
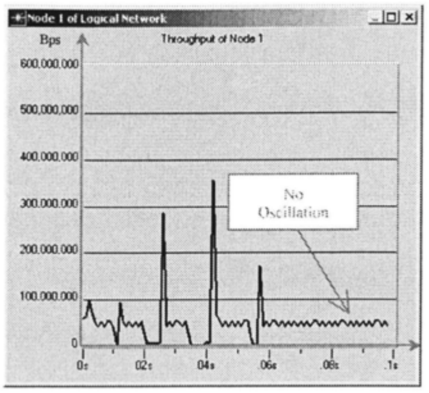


Figure 3.12: Simulation Result of Node 1 in Conservative Mode

Figure 3.12 shows that oscillation in the Node 1 is still quite large. Other nodes behave similar to Node 1.

3.3.1.3 Fuzzy Logic Control Mode

In this scenario, everything is kept the same as the previous case except the traffic control algorithm. Here, Fuzzy Logic Control is used to direct the behavior of the system. The result of simulation runs 100ms and is shown in Figure 3.13.



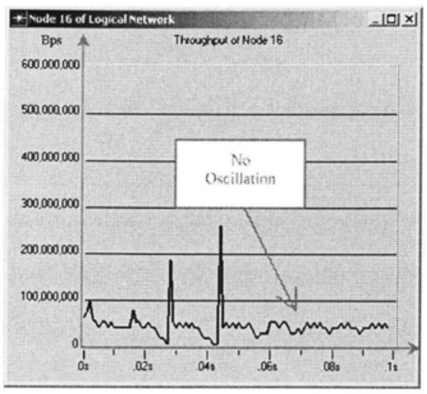


Figure 3.13: Simulation Results of Node 1 and Node 16 in FLC Mode

As can be seen in Figure 3.13, Node 1 does not create huge oscillation as in the previous scenarios, same thing for Node 16[1]. After about 50ms self-adjusting, each node in the ring has steady throughput, shares the bandwidth fairly and the oscillation is virtually eliminated.

3.3.2 Scenario 2 – Parking Lot Scenario

In this scenario, Dual Stage Queue [3] is adapted. Node 1 through Node 16 sends 42 Mbps traffic to Node 0. Furthermore, the last node, Node 16, send additional 210 Mbps traffic to its downstream neighbor, Node 15. In such case, Outer Ring between Node 1 and Node 0 forms a chock point and Node 1 advertises its local_fair_rate of 622 Mbps / 16 = 38.875 Mbps to its upstream neighbors. Upon receiving such rate all the upstream neighbors adjust their throughput to comply with Node 1, including Node 16.

[1] All the other nodes behave almost the same as Node 39 so we omit the results for them.

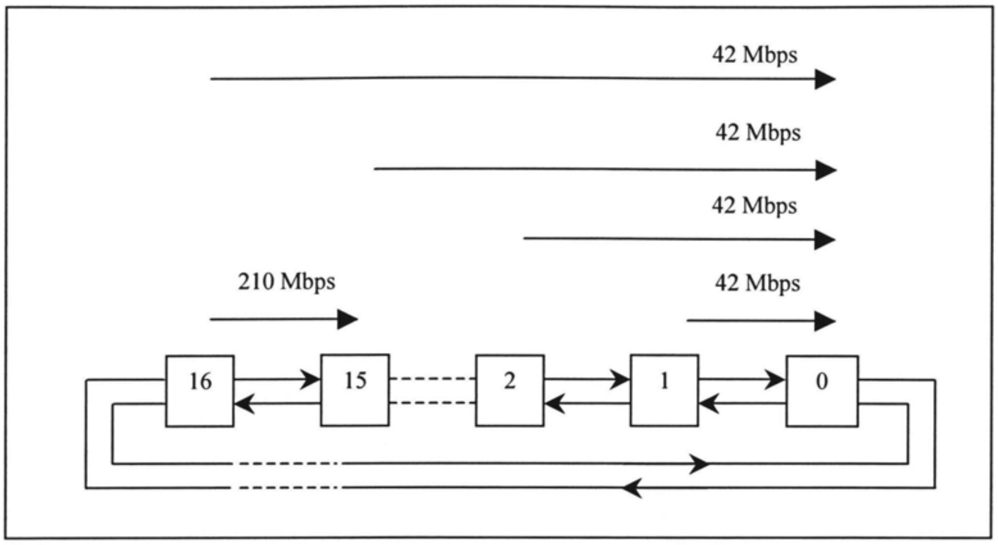


Figure 3.14: Parking Lot Scenario

However, Node 16 has extra 210 Mbps traffic sending to Node 15, and this traffic should not be affected by the advertisement from Node 1, otherwise Node 16 does not fully utilize the available bandwidth of the link. Therefore the ideal throughput of Node 16 should be 38.875 + 210 = 248.875 Mbps. Let's study the behaviors of the Aggressive Mode and FLC Mode next.

3.3.2.1 Aggressive Mode

In this scenario all the nodes run in Aggressive mode, the simulation runs 200ms and the results are shown in Figure 3.15.

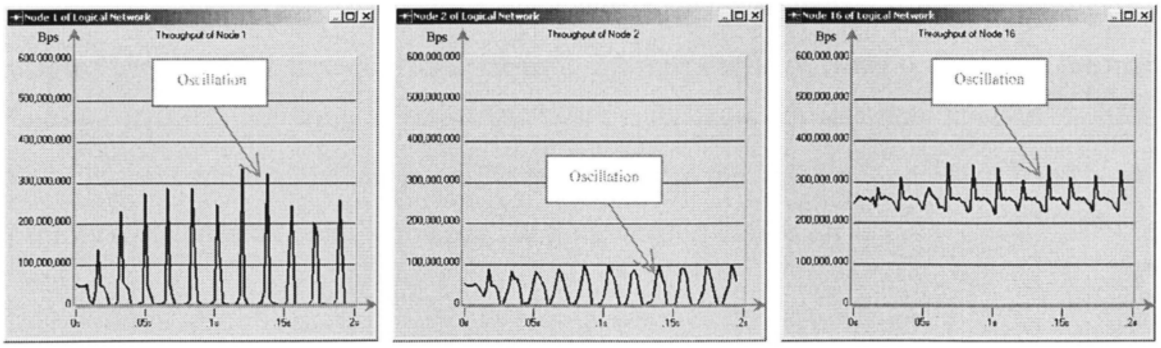


Figure 3.15: Simulation Results of Parking Lot Scenario in Aggressive Mode

As shown in Figure 3.15, Node 1 generates huge oscillation. Throughput of Node 2 through Node 15 is very similar. Only Node 2 is shown here. As can be seen in the figure, oscillation occurs and remains in steady state. Node 16 generates about 250 Mbps traffic in average, which complies with the principle of spatial reuse, but oscillation is also occurred.

3.3.2.2 Fuzzy Logic Control Mode

In this scenario all the nodes run in FLC mode. As shown in Figure 3.16, oscillations in all the nodes are alleviated after about 50ms for self-adjusting. Meanwhile, Node 16 generates around 250Mbps traffic, which complies with the principle of spatial reuse.

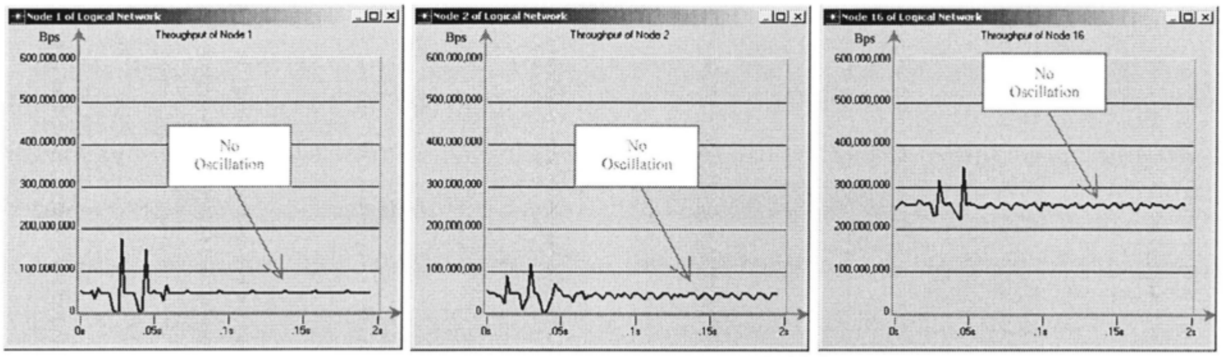


Figure 3.16: Simulation Results of Parking Lot Scenario in FLC Mode

3.3.3 Scenario 3 – Multi-Chock Parking Lot Scenario

In this scenario, two chock points are created in the ring. Again, Dual Stage Queue [3] is adapted here. Node 1 through Node 16 send 42Mbps traffic to Node 0, therefore the link between Node 1 and Node 0 forms the first chock point. In addition, Node 9 through Node 16 send additional 42Mbps traffic to Node 8, so the link between Node 8 and Node 9 forms second chock point in the ring, as illustrated in Figure 3.17.

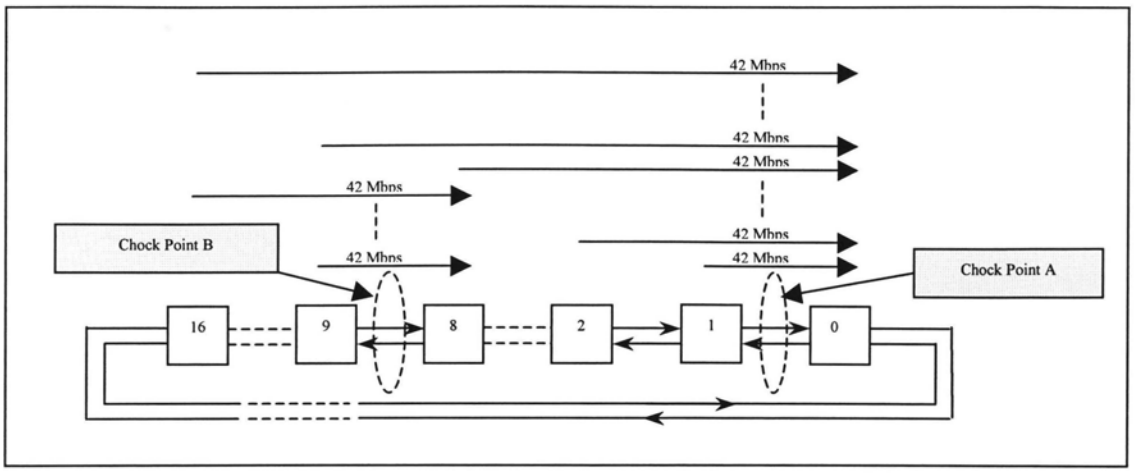


Figure 3.17: Multi-Chock Parking Lot Scenario

3.3.3.1 Aggressive Mode

In this scenario, all the nodes run in Aggressive mode, the simulation runs 200ms and the result is shown in Figure 3.18.

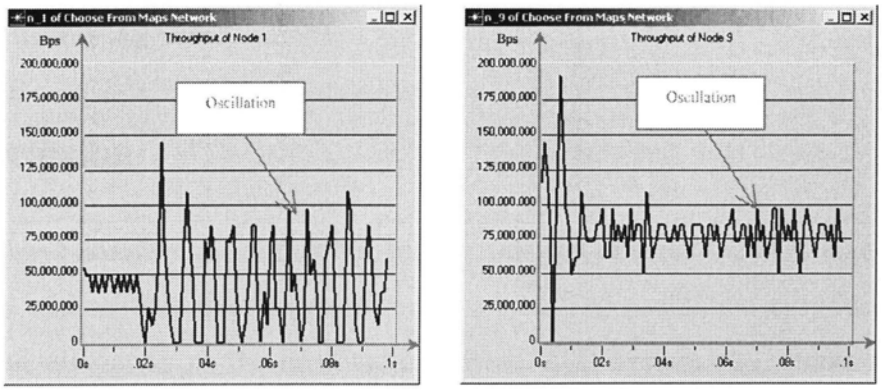
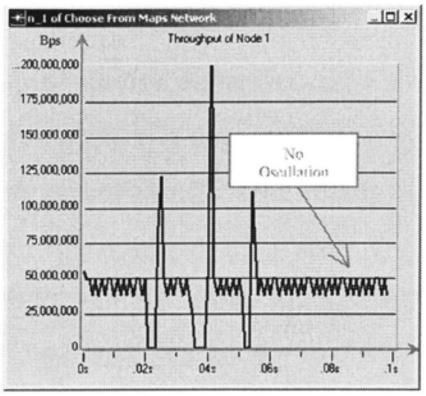


Figure 3.18: Simulation Results of Multi-Chock Parking Lot Scenario in Aggressive Mode

As can be seen from Figure 3.18, there are huge oscillation in Node 1. Node 9 has less oscillation than Node 1, but still quite large. All the other nodes don't have too much oscillation.

3.3.3.2 Fuzzy Logic Control Mode

In this scenario, all the nodes run in FLC mode, the simulation result is shown in Figure 3.19.



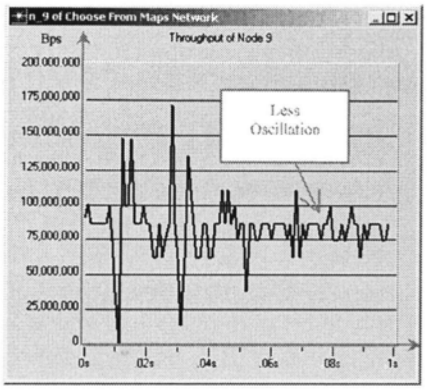


Figure 3.19: Simulation Results of Multi-Chock Parking Lot Scenario in FLC Mode

As one can see in Figure 3.19, the throughput in Node 1 settles in a very small degree of oscillation after about 50ms of self-adjusting period and Node 9 has less oscillation comparing with in Aggressive mode.

3.4 Conclusion

From the above analysis and simulation results, one can observe the FLC algorithm is far more stable and reliable than Aggressive algorithm. The oscillation is reduced together with the full utilization of the bandwidth and the RPR system can almost achieve the ideal designed goal. Therefore, one can confidently implement Fuzzy Logic Control in traffic control system in MAC layer of RPR protocol. Although it is a light-weigh algorithm, it dose solve some issues.

Chapter 4: Fairness Round Trip Time (FRTT) And The Size Of STQ

The quality of service depends on the size of STQ. If there is not enough space in STQ, the quality of service is degraded. The minimum size of STQ is studied in this Chapter to prevent low priority traffic from being discarded. IEEE 802.17 draft proposed a formula to calculate the minimal STQ size. The formula depends on the Fairness Round Trip Time (FRTT), but the upper bound of FRTT proposed in the draft is not a true upper bound, "... this calculation for *FRTT* is both overestimates and underestimates the true *FRTT*" [1]. A true upper bound of FRTT is derived in this Chapter. Using the upper bound of FRTT, one can find the minimal size of STQ to prevent STQ from overflowing even in the worst-case scenario; at the same time, not wasting STQ's space.

4.1 Bandwidth Reclaimable Characteristic Of ClassA1 Traffic

In the dual-transit-queue design of RPR, transit queues are divided into two parts: primary transit queue (PTQ), used to hold ClassA0 and ClassA1 traffic; and secondary transit queue (STQ), used to hold ClassB and ClassC traffic. Bandwidth for ClassA0 traffic is reserved and is not reclaimable by any other type of traffic. Bandwidth for ClassA1 traffic, however, is reclaimable by the lower priority traffic. In other words, if the bandwidth allocated for ClassA1 traffic is not used by ClassA1 traffic, ClassB and ClassC traffic can claim it.

This bandwidth reclaimable characteristic of ClassA1 traffic brings a fact that when ClassA1 traffic claiming its bandwidth there maybe a period of time the traffic rate of lower priority class exceeds the bandwidth left for that class, that period can last as long as Fairness Round Trip Time (FRTT). STQ is designed to hold these extra traffic during the period like this, but if designed too short, in the worst case scenario the traffic accumulated in STQ overflows; if designed too large, the fairness algorithm may take longer time to react because of increased value in its traffic control threshold (Low_Threshold) and the transit traffic may suffer longer delay. To find an appropriate size of STQ becomes an interesting subject in RPR.

4.2 FRTT

FRTT is the time between when congestion is detected to the time the fairness control takes effect and there is no more congestion in the ring. However, because of the unpredictable, bursty and stochastic nature of the traffic flow in the communication network, it is not possible to determine FRTT precisely, thus it is not easy to determine the STQ size precisely. A major contribution of this thesis is to derive a formula of the upper bound of FRTT to determine the minimal STQ size. The upper bound of FRTT is divided into two periods. The first period is starting from the time when the head of congestion domain is aware of the congestion, thus igniting fairness algorithm and sending the fairness advertisement packet to its upstream neighbor to the time when the tail of the congestion domain receives this fairness advertisement packet. This time period is defined as FRTT1. The second period, FRTT2, is measured from the time when the tail of congestion domain receives a new fairness advertisement packet sent by its downstream neighbor, denoted as t_0, to all the traffic sent by this tail node before t_0 has been drained (delivered to the destination). The upper bound of FRTT would be the sum of the upper bound of FRTT1 and the upper bound of FRTT2. Note that after FRTT the number of queued packets in STQ does not increase, the maximum queue length of STQ should occur within FRTT.

4.2.1 The Upper Bound Of FRTT1

Figure 4.1 demonstrates how the upper bound of FRTT1, denoted as MAX(FRTT1), is calculated. Here, traffic is generated from node 1 through node N inclusively and destined in outer ring to node N+1. When traffic increased to a threshold, congestion point formed between node N and node N+1 and node 1 to node N constitutes a congestion domain.

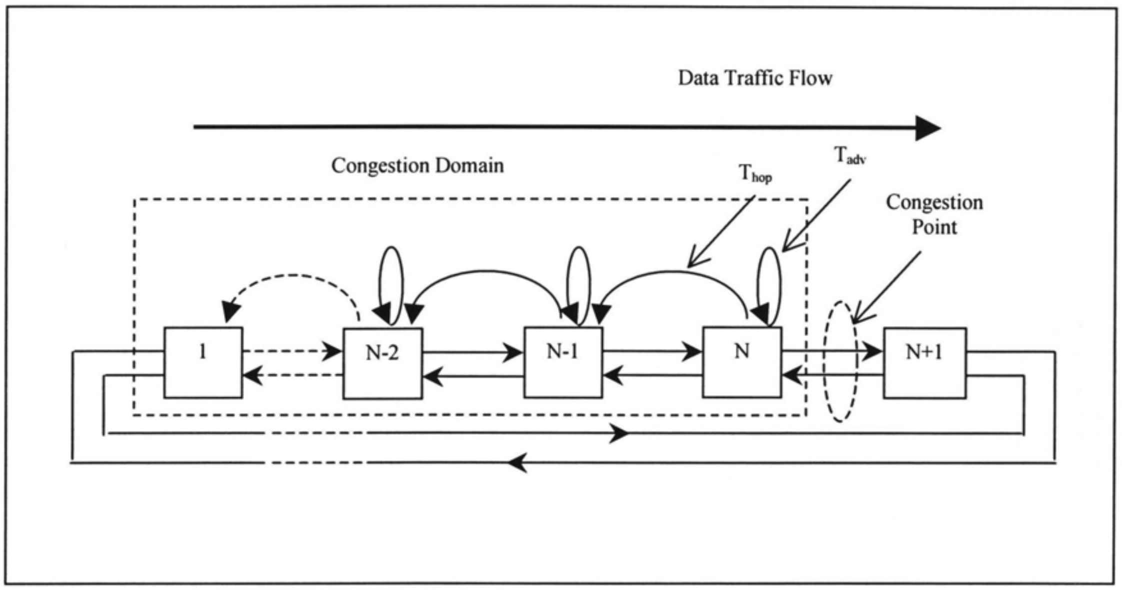


Figure 4.1: Calculate Upper Bound Of FRTT1

At this moment, node N starts running fairness algorithm and gets a fair rate. In the worst-case scenario, node N could wait up to one full advertisement interval, T_{adv}, to send the fairness advertisement packet to its upstream neighbor, node N-1, and the fairness packet takes hop time, T_{hop}, to reach it. Therefore, the maximum time from node N, which is aware of congestion, to node N-1, which receives fairness advertisement packet, is $T_{adv} + T_{hop.}$ If we define the distance between node *x* and node *y* is defined as $D_{x,y}$, then T_{hop} could be expressed as Equation 4.1:

$$T_{hop} = 5us \times D_{x,y} \qquad \text{(E4.1)}$$

Therefore the maximum of FRTT1 expressed as Equation 4.2:

$$\mathrm{MAX(FRTT1)} = (N - 1) * T_{adv} + 5us * D_{1,N} \qquad \text{(E4.2)}$$

4.2.2 The Upper Bound Of FRTT2

To calculate the upper bound of FRTT2, let's measure the time it takes for the last packet sent out by node 1 before t_0 until this packet reaches its destination, node N+1. This packet is marked as *tagged* packet. Also, to calculate the upper bound, the tagged packet always finds that the STQ depth of node 1 to node N-1 upon arrival is *STQLT*. Here *STQLT* is the low threshold of STQ, if packet accumulated in STQ beyond this threshold, node reports congestion. The tagged packet which finds the STQ depth of node N upon arrival is *S* in the worst-case scenario, where *S* is the STQ size in a node.

Two steps are taken to calculate the upper bound of FRTT2. First, the sum of STQ drain time within all the nodes in the ring except the tail node[1]is calculated; second, the total hop time from the tail node to the head node of a congestion domain is calculated. These two values are added together to get FRTT2. Here the STQ drain time of node *i*, $T_{stq,i}$, is defined as the duration of the tagged packet stays in the STQ of node *i* before being sent out.

$T_{stq,i}$ can be expressed in general as:

$$T_{stq,i} = \frac{STQ\_Depth(i)}{STQ\_Output\_Rate(i)}. \qquad \text{(E4.3)}$$

Here *STQ_Depth(i)* is the length of the packets accumulated in STQ in node *i* upon the arrival of the tagged packet, *STQ_Output_Rate(i)* is the speed of the node *i* transmitting the packet out of its STQ.

[1] Fairness eligible traffic generated from the tail node is add traffic, which does not go through transit queue.

Therefore, the total STQ drain time is the sum of each individual STQ drain time in each node within the congestion domain.

$$MAX(T_{stq}) = \sum_{i=2}^{N} T_{stq,i} = \sum_{i=2}^{N} \frac{STQ\_Depth(i)}{STQ\_Output\_Rate(i)} \qquad \text{(E4.4)}$$

Note that the tagged packet doesn't pass through its own transit queue, so the above formula doesn't take node 1 into consideration.

4.2.2.1 Maximum STQ Drain Time in Node *i* – MAX($T_{stq,i}$)

To find out the upper bound of STQ drain time, let's assume in the worst-case scenario the tagged frame encounters STQ with *STQLT* length in node 1 through node N-1 and with *S* length in node N along the way to its destination. Also, let's assume each node in the ring would generate ClassA1 traffic at the rate of R_{A1}, and ClassB_CIR traffic at the rate of R_B and such traffics are evenly distributed within the ring, and the maximum number of node sharing the bandwidth is *N*, then to node i^1, $2 \le i \le N-1$, we have:

$$\begin{aligned} MAX(T_{stq,i}) &= \frac{STQLT}{(i-1)\times\frac{LS - N(R_{A1}+R_B)}{N} + (N-1)\times R_B} \\ &= \frac{STQLT}{(i-1)\times(\frac{LS}{N} - R_{A1} - R_B + \frac{N-1}{i-1}R_B)} \\ &= \frac{STQLT}{(i-1)\times(\frac{LS}{N} - R_{A1} + \frac{N-i}{i-1}R_B)} \\ &= \frac{S}{8\times(i-1)\times(\frac{LS}{N} - R_{A1} + \frac{N-i}{i-1}R_B)} \qquad (2 \le i \le N-1) \qquad \text{(E4.5a)} \end{aligned}$$

[1] Because the traffic in tail node is the local add traffic, it doesn't go through its own transit queue, therefore we don't need to count STQ drain time in this node.

to node $i = N$, we have:

$$MAX(T_{stq,n}) = \frac{S}{(N-1) \times \frac{LS - N(R_{A1} + R_B)}{N} + (N-1) \times R_B}$$

$$= \frac{S}{(N-1) \times \frac{LS - NR_{A1}}{N}}$$

$$= \frac{NS}{(N-1) \times (LS - NR_{A1})} \qquad (i = N) \qquad \text{(E4.5b)}$$

Here *$[LS - N(R_{A1} + R_B)] / N$* is the *fair_rate* that allocated for each node in the congestion domain. R_{A1} is the output rate of local add ClassA1[1] traffic and R_B is the output rate of local add ClassB traffic per node. According to the IEEE 802.17 draft, *STQLT* is 1/8 of *S*.

Traffic sent from tail node before t_0 goes through N – 1 nodes inside the congestion domain to reach its destination, so we have:

$$MAX(T_{stq}) = \sum_{i=2}^{N} MAX(T_{stq,i})$$

$$= \frac{S}{8} \times \sum_{i=2}^{N-1} \{ \frac{1}{(i-1) \times (\frac{LS}{N} - R_{A1} + \frac{N-i}{i-1} R_B)} \} + \frac{NS}{(N-1) \times (LS - NR_{A1})} \qquad \text{(E4.6)}$$

4.2.2.2 Upper Bound Of FRTT2 – MAX(FRTT2)

Hop time in FRTT2 is the same as in FRTT1, the total hop time from node N to node 1 is:

$$T_{hop} = 5us \times D_{1,N} \qquad \text{(E4.7)}$$

The equation for the maximum of FRTT2 is:

[1] Bandwidth of ClassA0 is reserved, and it doesn't need to be considered here.

$$MAX(FRTT2) = MAX(Tstq) + Thop$$

$$= \frac{S}{8} \times \sum_{i=2}^{N-1} \{ \frac{1}{(i-1) \times (\frac{LS}{N} - R_{A1} + \frac{N-i}{i-1} R_B)} \} + \frac{NS}{(N-1) \times (LS - NR_{A1})}$$

$$+ 5us \times D_{1,N} \qquad \text{(E4.8)}$$

4.2.3 Upper Bound Of FRTT – MAX(FRTT)

The upper bound for FRTT is derived from Equations 4.2 and 4.8 as follows.

$$MAX(FRTT) = MAX(FRTT1) + MAX(FRTT2)$$

$$= (\mathrm{N} - 1) \times \mathrm{Tadv} + 5us \times D_{1,N} + \frac{S}{8} \times \sum_{i=2}^{N-1} \{ \frac{1}{(i-1) \times (\frac{LS}{N} - R_{A1} + \frac{N-i}{i-1} R_B)} \}$$

$$+ \frac{NS}{(N-1) \times (LS - NR_{A1})} + 5us \times D_{1,N}$$

$$= (\mathrm{N} - 1) \times \mathrm{Tadv} + 2 \times 5us \times D_{1,N}$$

$$+ \frac{S}{8} \times \sum_{i=2}^{N-1} \{ \frac{1}{(i-1) \times (\frac{LS}{N} - R_{A1} + \frac{N-i}{i-1} R_B)} \} + \frac{NS}{(N-1) \times (LS - NR_{A1})} \qquad \text{(E4.9)}$$

4.3 The Minimal Size Of STQ

One of the applications of using the upper bound of FRTT is to determine the minimal size of STQ to hold ClassB and ClassC traffic without overflow even in the worst-case scenarios. IEEE 802.17 draft [1] gives a formula to describe the relationship between the size of STQ, FRTT and ClassA1 traffic. The formula is shown as follows:

$$S \geq STQLT + FRTT \times (R_{A1} + R_B) + (STQHT - STQLT) \times \frac{LS - (R_{A1} + R_B)}{LS} \qquad \text{(E4.10)}$$

In this formula, STQHT is high threshold of STQ, according to the draft, it is equal to ¼ of STQ size.

The draft assumes that in the worst-case scenario all the incoming traffic is either ClassB or ClassC, and that the local add traffic is ClassA1, ClassB, and ClassC traffic.

$$\begin{aligned}
S &\geq \frac{1}{8}S + FRTT \times (R_A + R_B) + \frac{1}{8}S \times \frac{LS - (R_A + R_B)}{LS} \\
&\geq \frac{1}{8}S + \frac{1}{8}S - \frac{1}{8}S \times \frac{R_A + R_B}{LS} \\
&\quad + [(\mathrm{N}-1) \times \mathrm{Tadv} + 2 \times 5us \times D_{1,N} + \frac{S}{8} \times \sum_{i=2}^{N-1} \{ \frac{1}{(i-1) \times (\frac{LS}{N} - R_{A1} + \frac{N-i}{i-1} R_B)} \} \\
&\quad + \frac{NS}{(N-1) \times (LS - NR_{A1})}] \times (R_A + R_B) \qquad \text{(E4.11)}
\end{aligned}$$

Let's consider

$$D = (\mathrm{N}-1) \times \mathrm{Tadv} + 2 \times 5us \times D_{1,N} \ ,$$

$$E = \frac{1}{8} \times \sum_{i=2}^{N-1} \{ \frac{1}{(i-1) \times (\frac{LS}{N} - R_{A1} + \frac{N-i}{i-1} R_B)} \},$$

$$F = \frac{N}{(N-1) \times (LS - NR_{A1})},$$

then we have:

$$S \geq \frac{2}{8}S - \frac{(R_A + R_B)}{8LS}S + (D + ES + FS) \times (R_A + R_B)$$

$$S \geq \frac{2}{8}S - \frac{(R_A + R_B)}{8LS}S + (E+F) \times (R_A + R_B) \times S + D \times (R_A + R_B)$$

$$[1 - \frac{2}{8} + \frac{(R_A + R_B)}{8LS} - (E+F) \times (R_A + R_B)] \times S \geq D \times (R_A + R_B)$$

$$\frac{6LS + R_A + R_B - (E+F) \times (R_A + R_B) \times LS}{8LS} \times S \geq D \times (R_A + R_B)$$

$$\frac{6LS + (R_A + R_B) \times [1 - 8(E+F) \times LS]}{8LS} \times S \geq D \times (R_A + R_B)$$

$$S \geq \frac{8D \times LS \times (R_A + R_B)}{6LS + (R_A + R_B) \times [1 - 8(E+F) \times LS]}$$

$$S \geq \frac{8D \times (R_A + R_B)}{6 + \frac{R_A + R_B}{LS} \times [1 - 8(E+F)]} \qquad \text{(E4.12a)}$$

If $\lambda = \frac{R_A + R_B}{LS}$, then equation 4.25a becomes:

$$S \geq \frac{8D \times (R_A + R_B)}{6 + \lambda[1 - 8(E+F)]} \qquad \text{(E4.12b)}$$

4.4 Simulation

To verify the correctness of the above equations, a few scenarios are created in OPNET to compare them with the simulation results. FRTT is divided into several parts to create separate scenarios to verify them. This would allow investigating each part separately.

4.4.1 Scenario 1 – Verify FRTT1

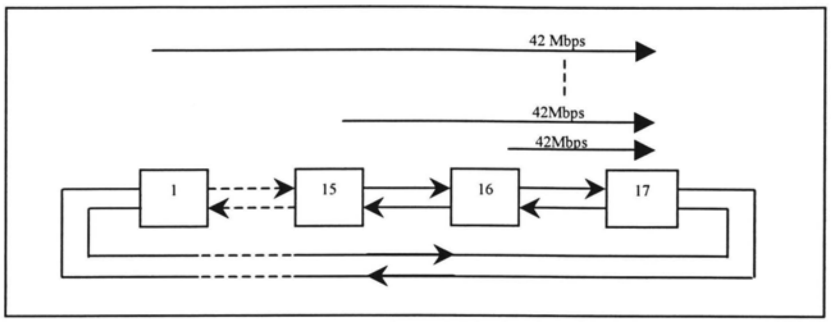


Figure 4.2: 17 Nodes Simple Sourced Traffic Scenario

In the scenario illustrated in Figure 4.2, node 1 through node 16 sends 42Mbps ClassC bursty traffic in outer ring to node 17. Each node sends 20Mbps ClassA1 bursty traffic to every other node in the outer ring as well. (See Table 4.1.) Thus we expect 70Mbps ClassA1 bursty traffic passing through PTQ in each node. The distance between two adjacent nodes is 14 kilometers and the advertisement interval T_{adv} is 0.05 ms.

Traffic Type	ClassA1	ClassC
Start Time (ms)	40	1
ON Time (ms)	Exponential (10)	Exponential (10)
OFF Time (ms)	Exponential (10)	Exponential (10)
Interarrival Time (ms)	Exponential (0.3)	Exponential (0.143)
Packet Size (bits)	12000	12000
Traffic Rate (mbps)	20	42

Table 4.1: Traffic Generation Pattern

According to Equation 4.2,

$$MAX(FRTT1) = (N - 1)*T_{adv} + 5us*D_{1,N} = (16-1) * 0.05 + 0.005 * 14 * 15 = 1.80 \text{ (ms)}$$

Table 4.2 records the time stamped when a node receiving and sending fairness packet in the OPNET simulation program. FRTT1 is counted as the time spent from node 16 send fairness packet to node 1 receiving fairness packet, which is 51.47 - 50.0 = 1.47 ms. The simulation result is within calculated parameters.

Node id	*16*	*15*	*14*	*13*	*12*	*11*	*10*	*9*	*8*	*7*	...	*3*	*2*	*1*
Send Fairness PK At (ms)	**50**	50.1	50.2	50.3	50.4	50.5	50.6	50.7	50.8	50.9	...	51.3	51.4	-
Receive Fairness PK At (ms)	-	50.07	50.17	50.27	50.37	50.47	50.57	50.67	50.77	50.87	...	51.27	51.37	**51.47**
FRTT1 (ms)	51.47 – 50.0 = 1.47													

Table 4.2: Simulation Result of FRTT1

4.4.2 Scenario 2 – Verify T_{stq} With Constant Traffic

In this scenario, a RPR ring with 9 nodes is creayed as shown in Figure 4.3, each node sends two types of constant traffic in outer ring[1]: 20 Mbps of ClassA in constant rate and 80 Mbps of ClassC in constant rate. ClassA traffic's destination is the node's upstream node, (because the traffic is sent in outer ring, this flow of traffic needs to pass through all the nodes in the ring to reach its destination, which is what is needed.) ClassC traffic's destination is node 9 (see Table 4.3). Node 1 through node 8 starts sending ClassC traffic at 1 ms and ClassA1 traffic at 30 ms.

[1] Except node 9 which only sends ClassA traffic.

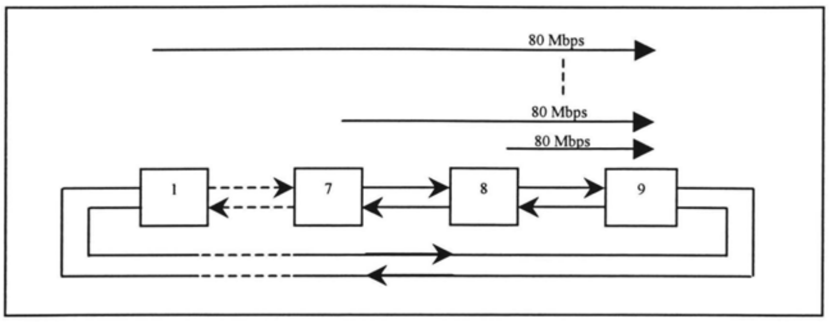


Figure 4.3: 9 Nodes Simple Sourced Traffic Scenario

Node	*ClassA1*			*ClassC*		
	Rate (Mbps)	*Dest. Node*	*Start Time (ms)*	*Rate (Mbps)*	*Dest. Node*	*Start Time (ms)*
1	20	9	30	80	9	1
2	20	1	30	80	9	1
3	20	2	30	80	9	1
…	…	…	…	…	…	…
8	20	7	30	80	9	1
9	20	8	30	-	-	-

Table 4.3: Traffic Generation Pattern

In this scenario the size of STQ is 228,000 bit and *N* is 8, according to Equation 4.6,

$$MAX(T_{\text{stq}}) = \frac{S}{8} \times \sum_{i=2}^{N-1} \{ \frac{1}{(i-1) \times (\frac{LS}{N} - R_{A1} + \frac{N-i}{i-1} R_B)} \} + \frac{NS}{(N-1) \times (LS - NR_{A1})}$$

$$= \frac{228000}{8} \times [\frac{1}{\frac{622.08\times10^6}{8} - 20\times10^6} + \frac{1}{2\times(\frac{622.08\times10^6}{8} - 20\times10^6)} \ldots$$

$$+ \frac{1}{6\times(\frac{622.08\times10^6}{8} - 20\times10^6)}] + \frac{8\times228000}{(8-1)\times(622.08\times10^6 - 8\times20\times10^6)}$$

$$= 0.00177\ (s) = 1.77\ (ms)$$

The theoretical result of total maximum STQ drain time of all nodes is 1.77 ms.

Table 4.4 shows the queuing information of STQ in each node while the tagged packet passing through, including STQ drain time and STQ depth (total packets length in STQ).

Node	*Receive PK At t_1 (ms)*	*Send PK At t_2 (ms)*	*STQ Drain Time $t_2 - t_1$ (ms)*	*STQ Depth (bit)*
2	30.81	30.82	0.01	0
3	30.93	30.94	0.01	0
4	31.05	31.07	0.02	0
5	31.18	31.32	0.14	0
6	31.43	31.48	0.05	3000
7	31.59	31.79	0.2	7500
8	31.90	32.13	0.23	10500
Total			0.66	21000

Table 4.4: Scenario Result of Constant Traffic In Each STQ

From table 4.4 one can see the tagged packet spent 0.66 ms to pass though 21000 bit of STQ.

From the result we can see when nodes send traffic in constant rate, the STQ depth in each node is maintained in lower level, some even zero; however in theoretical calculation we assume STQ in each node has at least STQLT bit and the congestion head node has *S* bit, therefore, the difference of STQ depth between real simulation and theoretical assumption causes the STQ drain time in real simulation far less than that in theoretical calculation.

4.4.3 Scenario 3 – Verify T_{stq} With Bursty Traffic

In this scenario, all the settings are kept as in the previous scenario except nodes generating bursty ClassA and ClassC traffic – the traffic generation pattern is shown in Table 4.5. The information of tagged packet passing through each node is shown in Table 4.6:

Traffic Type	ClassA1	ClassC
Start Time (ms)	30	1
ON Time (ms)	Exponential (10)	Exponential (10)
OFF Time (ms)	Exponential (10)	Exponential (10)
Interarrival Time (ms)	Exponential (0.3)	Exponential (0.075)
Packet Size (bits)	12000	12000
Traffic Rate (mbps)	20	80

Table 4.5: Traffic Generation Pattern Of Scenario 3

Node	*Receive PK At* t_1 *(ms)*	*Send PK At* t_2 *(ms)*	*STQ Drain Time* $t_2 - t_1$ *(ms)*	*STQ Depth (bit)*
2	67.63	67.65	0.02	0
3	67.76	67.77	0.01	0
4	67.88	68.01	0.13	4500
5	68.12	68.49	0.37	10500
6	68.60	68.77	0.17	6000
7	68.88	69.02	0.14	7500

8	69.13	69.59	0.46	19500
Total			1.3	48000

Table 4.6: Scenario Result of Bursty Traffic In Each STQ

From table 4.6 one can see the tagged packet spent 1.3 ms to pass though around 48000 bits of STQ.

From the result one can see when nodes send traffic in bursty rate, the STQ depth in each node is kept in higher level, therefore, the STQ drain time is also higher than that of the constant rate.

4.4.4 Scenario 4 – Verify FRTT

In this scenario, all the settings are kept as the above. The theoretical result for FRTT is:

$$\mathrm{MAX(FRTT)} = (\mathrm{N}-1)\times \mathrm{Tadv} + 2\times 5us\times D_{1,N} + \frac{S}{8}\times\sum_{i=2}^{N-1}\{\frac{1}{(i-1)\times(\frac{LS}{N}-R_{A1}+\frac{N-i}{i-1}R_B)}\}$$

$$+\frac{NS}{(N-1)\times(LS-NR_{A1})}$$

$$=(8-1)\times 5\times 10^{-5}+2\times 5\times 10^{-6}\times 14\times 7+\frac{228000}{8}\times$$

$$[\frac{1}{\frac{622.08\times 10^6}{8}-20\times 10^6}+\frac{1}{2\times(\frac{622.08\times 10^6}{8}-20\times 10^6)}\ldots+\frac{1}{6\times(\frac{622.08\times 10^6}{8}-20\times 10^6)}]$$

$$+\frac{8\times 228000}{(8-1)\times(622.08\times 10^6-8\times 20\times 10^6)}$$

$$=3.1 \qquad \text{(ms)}$$

The actual scenario is illustrated as follows: the time of congestion point formed between node 8 and node 9 was at 66.83 ms, node 8 sent the advertisement packet to its upstream node at 66.85 ms, node 1 received such advertisement packet at 67.42 ms, and the tagged packet sent out by node 1 right before receiving fairness information at 67.40 ms, this packet flowing through all the nodes in the ring and eventually sent out by node 8 at 69.59 ms, so the total FRTT was 69.59 – 66.83 = 2.76 ms.

4.4.5 Scenario 5 – Verify STQ Size

According to equation 4.12b, $S \geq \frac{8D\times(R_A+R_B)}{6+\lambda[1-8(E+F)]}$. When R_{A1}=40 Mbps, N = 8, then

$$\lambda = \frac{R_A+R_B}{LS} = 0.0643,$$

$$\mathrm{D} = (\mathrm{N}-1)\times \mathrm{Tadv} + 2\times 5us \times D_{1,N} = (8-1)\times 5\times 10^{-5} + 2\times 5\times 10^{-6}\times 14\times 7 = 1.33\times 10^{-3};$$

$$E = \frac{1}{8}\times\sum_{i=2}^{N-1}\{\frac{1}{(i-1)\times(\frac{LS}{N}-R_{A1}+\frac{N-i}{i-1}R_B)}\} = \frac{1}{8}\times$$

$$[\frac{1}{\frac{622.08\times10^6}{8}-20\times10^6}+\frac{1}{2\times(\frac{622.08\times10^6}{8}-20\times10^6)}\ldots+\frac{1}{6\times(\frac{622.08\times10^6}{8}-20\times10^6)}]$$

$$= 8.11\times10^{-9}$$

$$F = \frac{N}{(N-1)\times(LS-NR_{A1})} = \frac{8}{(8-1)\times(622.08\times10^6-8\times20\times10^6)} = 3.78\times10^{-9},$$

therefore,

$$S \geq \frac{8D\times(R_A+R_B)}{6+\lambda[1-8(E+F)]}$$

$$\geq \frac{8\times1.33\times10^{-3}\times40\times10^6}{6+0.0643\times[1-8\times(8.11\times10^{-9}+3.78\times10^{-9})]}$$

$$\geq 188354 \text{ (bit)}$$

From the calculation one can see when ClassA1 traffic is 40 Mbps for each node and there are 8 nodes in the congestion domain, if the STQ size set to be more than 188354 bits, the loss of data, theoretically, would not happen even in the worst-case scenario.

In the final scenario, let's setup a worst-case situation, where only node 1 and node 8 generate traffic. The generation rate of node 1 is 500 Mbps in bursty and node 8 is 100 Mbps in bursty. Figure 4.4 illustrates ClassC traffic generation pattern in this scenario. Moreover, each node in the ring is setup to generate 40 Mbps ClassA1 traffic in bursty. STQ's full size is set to 192000 bit in each node.

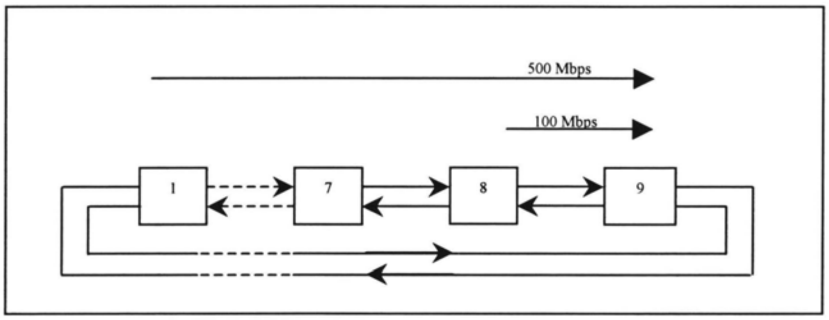


Figure 4.4: 2 Nodes Simple Sourced Traffic Scenario

Table 4.7 shows the maximal STQ depth occurred in each node. From the result one can see that the maximal STQ depth does not exceed the theoretical calculation.

Node	*Maximal STQ Depth (bit)*
2	24000
3	36000
4	24000
5	24000

6	36000
7	36000
8	96000

Table 4.7: Maximal STQ Depth In Each Node Of Scenario 5

Chapter 5: Conclusion and Future Work

5.1 Conclusion

This thesis has presented the architecture and simulation of Resilient Packet Ring, a next generation layer-2 protocol that is implemented in the high-speed switches or routers. The contribution achieved in this thesis are as follows.

5.1.1 Design and Implementation of the Simulation Model

The first contribution of this thesis is the design and development of a simulation model to serve future research purposes. The main requirement on such a model is that the simulation model must be able to accurately follow the protocol from the IEEE 802.17 proposal, otherwise, the simulation results generated from the model would be incorrect. Therefore, great efforts had been put to make sure the accuracy of the architecture and the simulation code in the model.

The model has been designed in four layers, upper layer, LLC layer, MAC layer and Physical layer. Upper layer simulates the IP traffic, LLC and MAC layer is where the RPR resides in, Physical layer simulates sending and receiving the packets. In the MAC layer, there are control units and queues in which major RPR protocols implemented. Topology Discovery, Ringlet Selection, Fairness Algorithms and Rate Control are the main protocols implemented in this in-house simulation bed.

5.1.2 Fuzzy Logic Control

An alternative fairness algorithm, Fuzzy Logic Control, has been designed and implemented to overcome the limitation of the Aggressive and Conservative algorithms proposed by IEEE 802.17 Work Group. Aggressive algorithm is fast in convergent but has large oscillation when high volume traffic is flowing in the ring, Conservative algorithm has less oscillation but is slow in convergent, furthermore it does not utilize the bandwidth efficiently. Fuzzy Logic Control is an intelligent bandwidth control algorithm that reduces the oscillation and has a fast convergent time.

5.1.3 FRTT And The Minimal Size Of STQ

Fairness Round Trip Time reflects the time for the effect of fairness algorithm takes effect on the traffic flow in the congestion domain. The upper bound of FRTT is derived and from it the minimal size of STQ is derived. Simulation results confirm the validity of the derivation.

5.2 Future Work

The future work should concentrate on the following topics.

5.2.1 Ring Protection

One of the outstanding features of RPR, which is similar to SONET/SDH, is that it can monitor the connectivity of the ring. Once it finds the link is broken it automatically uses another ring to transfer the traffic, hence keeping the connectivity for all the nodes in the ring. The performance of RPR during such link failure should be studied.

5.2.2 Improvement of Fuzzy Logic Control

In this thesis we introduced Fuzzy Logic Control as an alternative algorithm for bandwidth control. However the algorithm in FLC is in its primary stage. Our study concentrated mainly on whether it is possible to reduce the degree of oscillation by applying the concept of Fuzzy Logic. More study is required to derive a better membership functions and Fuzzy Logic Rules for RPR.

5.2.3 Improvement of QoS Control

Many areas are still left open for QoS control. The in-house RPR model can act as a research platform to investigate the QoS.

5.2.4 Improvement of FRTT

From the simulation results one can see that the FRTT bound derived here may not by very tight. A tighter bound is certainly more desirable for the estimation of STQ size.

5.2.5 Improvement of Fairness Algorithm

Fairness Algorithm is working well when upstream nodes try to compete for the bandwidth, however, when some of the upstream nodes generate far more less than fair share bandwidth, problem occurs. Figure 5.1 shows one example of such scenario, in which Node 1 generate 580 Mbps traffic to Node 3 and Node 2 only generate 30 Mbps traffic to Node 3. In this case when the link between Node 2 and Node 3 is congested, Fairness Algorithm is ignited in Node 2 and a fair-shared rate are calculated, which is 300 Mbps in this scenario and such rate is advertised out to Node 1. Once Node 1 receives such rate, it adjusts itself to accommodate the change, therefore, Node 1 generates 300 Mbps traffic to Node 3. However, Node 2 still sends the traffic at its original rate, so the total bandwidth of the link between Node 2 and Node 3 is 300 + 30 = 330 Mbps. Because the link is not congested at this moment, fairness algorithm directs Node 2 to advertise full rate to its upstream nodes. When Node 1 receives such advertised rate, it sends the traffic to Node 3 at its original rate, which forms another round of oscillation. Figure 5.2 shows such result.

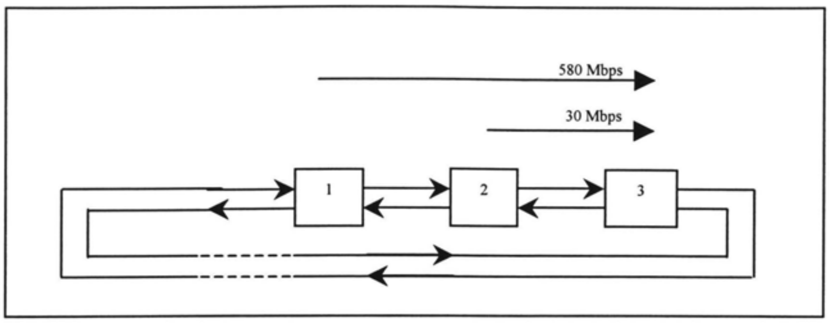


Figure 5.1: 2 Nodes High-Low Rate Scenario

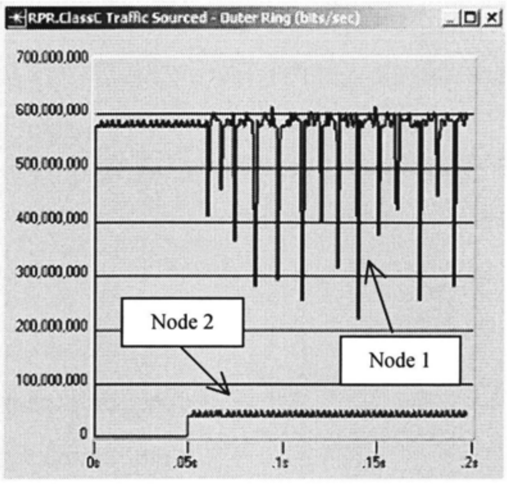


Figure 5.2: Simulation Result of High-Low Rate

Currently, this is an unsolved problem among RPR community. An open area for further research is to find a solution for this problem.

Appendix A: Building Topology Database

The following is the pseudo code[1] of constructing the topology database.

First, a packet structure and topology database structure are defined and named as `Packet` and `TopoMap` respectively.

```
struc Packet
{    int        src_address;
     int        dest_address;
     int        ri;           // Ringlet id.
     int        ttl;          // TTL, also used as hop count.
     char       pk_type[];
};

struct TopoMap
{    int        mac_address;
     int        hopcount_r0;
     int        hopcount_r1;
     Boolean    reachable_r0;
     Boolean    reachable_r1;
};

Packet*    pkptr;
TopoMap    topo_map[255]; // Maximum number of record.
```

At the startup of the simulation program, each node needs to initialize its own topology database by calling `InitTopoMap()` function.

[1] The code illustrated here is not a duplication of that in the source code of the simulation program. It is only used to demonstrate the functionality of the protocols.

```
void InitTopoMap()
{
      // Initialize this node.
      topo_map[0].mac_address = this_node_address;
      topo_map[0].hopcount_r0 = 0;
      topo_map[0].hopcount_r1 = 0;
      topo_map[0].reachable_r0 = TRUE;
      topo_map[0].reachable_r1 = TRUE;

      // Initialize other nodes.
      for (int i=1; i<255; i++)
      {
            topo_map[i].mac_address = 0;
            topo_map[i].hopcount_r0 = 0;
            topo_map[i].hopcount_r1 = 0;
            topo_map[i].reachable_r0 = FALSE;
            topo_map[i].reachable_r1 = FALSE;
      }

      return 0;
}
```

Each node on the ring broadcasts topology discovery packet to both rings at the initial phase of the simulation program and periodically later on. In another word each node on the ring receives topology discovery packet from time to time. When a node received such a packet, it retrieves the information from the packet to build or maintain its topology database. The following code demonstrates this functionality.

```
Boolean CreateTopoMap (Packet* pkptr)
{
      for (int i=1; i<255; i++)
           {
```

```
        // foreign node has record in this node.
        if (topo_map[i].mac_address == pkptr->src_address)
            break;
        // find the first available bland record.
        else if (topo_map[i].mac_address == 0)
            break;
        }

    if (i==255)
       {
       printf ("Error: Out of space for foreign node./n");
       return 1;
       }

    topo_map[i].mac_address = pkptr->src_address;

    if (pkptr->ringlet_id == 0)
       {
        topo_map[i].hopcount_r0 = pkptr->ttl;
        topo_map[i].reachable_r0 = TRUE;
       }
    else
       {
       topo_map[i].hopcount_r1 = pkptr->ttl;
       topo_map[i].reachable_r1 = TRUE;
       }
    return 0;
}
```

Appendix B: Implement of FLC in C Language

This section explains how C language is used to implement FLC into RPR.

B.1 Sample C Codes Of Membership Function

Low Membership function has three arguments, *a0, a1* and *x* that corresponds to equation E3.4, and the return value *y* corresponds to *Y1*. The function prototype of `f_low_trimf()` is:

```
double f_low_trimf (double a0, double a1, double x);
```

The sample C code of low membership function, `f_low_trimf()`, is given as follows.

```
/* Low Triangle MF or Negative Triangle MF.                */
double f_low_trimf (double a0, double a1, double x)
{
      double y, a, w;

      /* a is the middle point between a0 and a1.          */
      a = (a1 + a0) / 2;

      /* w is the half distance between a0 and a1.        */
      w = (a1 - a0) / 2;

      /* Get the return value based on the input.          */
      if (x < a0)
            y = 1.0;
      else if (x >= a0 && x < a)
            y = (a - x) / w;
      else
            y = 0.0;

      return (y);
}
```

Here the C code

```
if (x < a0)
      y = 1.0;
else if (x >= a0 && x < a)
      y = (a - x) / w;
else
      y = 0.0;
```

corresponds to E3.4.

Medium Membership function and High Membership function have the same structure as that of Low Membership function except the equation to get the result is different.

Let's use $x = 65$ Mbps in Figure 3.7 as an example, when we put x into low membership function, the result equals to 0; when we put x into medium membership function, the result equals to 0.8; when we put x into high membership function, the result equals to 0.2. In the Fuzzy Logic's point of view, we no longer refer to 65 Mbps as fast or slow; instead, we consider it to be in the state of 0 percent slow, 80 percent normal and 20 percent high.

B.2 Sample C Code Of If-Then Rule

Once we get the result from Membership Functions, we need to put the result of it into the If-Then Rule. The prototype of If-Then Rule function is:

```
double get_if_then_rule (int rule_id, double w, double percentage);
```

Here, the argument *rule_id* is described in Table 3.1, *w* is explained in Figure 3.7, *percentage* is the return value from Membership Functions.

The sample C code is shown below.

```
double get_if_then_rule (int rule_id, double w, double percentage)
      {
      double y;

      switch (rule_id)
            {
            case 1:  // rule 1.
                  {
                  y = (double) (w * percentage);
                  break;
                  }
            case 2:  // rule 2.
                  {
                  y = (double) (w * percentage / 4);
                  break;
                  }
            case 3:  // rule 3.
                  {
                  y = 0.0;
                  break;
                  }
            case 4:  // rule 4.
                  {
                  y = (double) -(w * percentage / 4);
                  break;
                  }
            case 5:  // rule 5.
                  {
                  y = (double) -(w * percentage);
                  break;
                  }
            default:
                  {
                  printf ("Error: Unable to determine rule id.\n");
```

```
                    break;
                    }
            }

        return (y);

        }
```

Each rule has a membership function associated with it. Their product multiplied by the weight determines the rate that the system should go for in the next step. However there are five rules in the system and the system can only generate one result, therefore we need to defuzzify these five rules so as to get only one result. Defuzzify is explained next.

B.3 The Sample C Code Of Defuzzify

The sample C code of Defuzzify is deprived from E3.7 and is shown as follows:

```
result = (int) ((output[0] * y[0][0] + output[1] * y[1][0] + output[2]
* y[1][1] + output[3] * y[1][2] + output[4] * y[0][2]) /
            (y[0][0] + y[1][0] + y[1][1] + y[1][2] + y[0][2]));
```

Here, the output array contains the return values from get_if_then_rule () function, y array contains the return values from membership functions.

B.4 The Sample C Code Of Fuzzy Logic Control

The following code is a sample of whole piece for Fuzzy Logic Control.

```
/* Triangle Fuzzy Algorithm   */
int fuzzy_trimf (double input1, double input1_a0, double input1_a1,
double input2, double input2_a0, double input2_a1, const double w)
        {
        double      y[2][3];
        double      output[5];
        int         result;
/* 1.1 Low membership function.                               */
y[0][0] = f_low_trimf (input1_a0, input1_a1, input1);

/* 1.2 Medium membership function.                       */
y[0][1] = f_med_trimf (input1_a0, input1_a1, input1);

/* 1.3 High membership function.                         */
y[0][2] = f_hi_trimf  (input1_a0, input1_a1, input1);


/* 2.1 Negative membership function.                     */
y[1][0] = f_low_trimf (input2_a0, input2_a1, input2);

/* 2.2 No-change membership function.                    */
y[1][1] = f_med_trimf (input2_a0, input2_a1, input2);

/* 2.3 Passive membership function.                      */
y[1][2] = f_hi_trimf  (input2_a0, input2_a1, input2);

/* If-Then Rules.                   */
output[0] = get_if_then_rule (1, w, y[0][0]);
output[1] = get_if_then_rule (2, w, y[1][0]);
output[2] = get_if_then_rule (3, w, y[1][1]);
output[3] = get_if_then_rule (4, w, y[1][2]);
output[4] = get_if_then_rule (5, w, y[0][2]);

/* Defuzzify.                       */
result  =  (int)  ((output[0]  *  y[0][0]  +  output[1]  *  y[1][0]  +
output[2] * y[1][1] + output[3] * y[1][2] + output[4] * y[0][2]) /
      (y[0][0] + y[1][0] + y[1][1] + y[1][2] + y[0][2]));


return (result);

}
```

Here, input1, input1_a0, input1_a1 are the arguments for add_rate membership functions; input2, input2_a0, input2_a1 are the arguments for Δadd_rate membership functions; w is half of the range FLC fluctuates within.

Reference

[1].IEEE Standard Department, "IEEE Proposed Draft Standard for Information Technology – Part 17: Resilient packet ring access method and physical layer specification," http://www.ieee802.org/17/documents/drafts/Darwin_v1_0.pdf, January 2002.

[2].RFC 2892, Cisco System, "The Cisco SRP MAC Layer Protocol", August 2000.

[3].K. Yip, N. Ma, H. Ghandehari, X. Zhang, and K. Raahemifar, "Design and Performance Analysis of RPR with Dual Stage Queues", RPR Report 1001, Ryerson University, Computer Networks, March 2003.

[4].Mathworks.com, Fuzzy Logic Tool Box, http://www.mathworks.com/access/helpdesk/help/toolbox/fuzzy/fuzzy.shtml

[5].OPNET Modeler 8.0 Online Documentation, OPNET Technologies, Inc.

[6].Alberto Leon-Garcia, Indra Widjaja, "Communication Networks – Fundamental Concepts and Key Architectures", McGraw-Hill, 2000.

[7].Internetworking Technology Handbook, Cisco Systems, http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/qos.htm

[8].An Introduction to Resilient Packet Ring Technology, Lantern Communications, http://www.lanterncom.com