Fairness Aspects of Buffer-Insertion Rings in General and Resilient Packet Rings in Particular

Doctoral Dissertation by

Bjørn Fredrik Davik

Submitted to the Faculty of Mathematics and Natural Sciences at the University of Oslo in partial fulfillment of the requirements for the degree Dr. Scient. in Computer Science

August 2005

To Jannicke, Gry Helene and Kaja

Abstract

The use of ring topologies in computer networks, was introduced in the late 1960s/early 1970s, with the Newhall ring being an example of an early implementation. Since then, many different solutions for ring networks have been proposed and implemented.

One of the fundamental problems in such networks, in the presence of several users with high demands for bandwidth, is to simultaneously achieve high utilization, low delay and fair access to the ring's bandwidth resources.

In this thesis, we cover issues related to the above problems in ring networks. However, we limit our focus to the above problems in the context of buffer-insertion rings and a particular type of buffer-insertion rings – namely the IEEE 802.17 Resilient Packet Ring.

As such, our published results have a strong focus on Resilient Packet Rings. However the general findings from our experiments should be applicable to buffer-insertion (as well as slotted) rings.

In our research contributions, we start by providing an introduction to the RPR standard and its problem area. This contribution provides a starting-point of study for the RPR novice, be it an network engineer or a university professor looking for an introduction to RPR technology.

Then, we proceed by presenting an analytical model of the RPR *aggressive* fairness mode. By this model, a starting point is provided for a safer configuration of a Resilient Packet Ring, as well as reducing the number of heuristics needed.

Next, we provide several contributions in which we analyze and propose improvements to various performance deficiencies in Resilient Packet Rings. Some of these deficiencies are commonly known in the RPR research community, while others are not so well known.

And finally, at the end, we present an application of RPR, which is made possible by the workings of the fairness and service differentiation capabilities provided by the RPR standard.

Acknowledgements

The work carried out in the context of this doctoral thesis project is made possible by the support of several.

I want to start by thanking my advisor prof. dr. Stein Gjessing, for providing invaluable insights, feedback and support at all phases of this thesis work. Gjessing has, by use of a clear and informal method, provided many valuable lessons on the methods of conducting research as well as on the writing of scientific papers for the dissemination of results obtained.

I am also very grateful for the support and resources provided to me by the Norwegian branch of Ericsson. I especially want to thank Odd Fredriksen, Ivar Versvik and Egil Ofstad whom supported me during a though time of company cutbacks. Further I want to thank Roar Kristensen, Arne Gjertsen, Geir Melby and Jo Herstad, for supporting my initial initiative to start this thesis work.

I also want to thank for the opportunity to conduct my thesis work at the Simula Research Laboratory's (SRL) facilities. The SRL infrastructure in terms of equipment and office facilities is excellent and so are the many people working here.

At SRL I want to thank the group of researchers, headed by prof. dr. Olav Lysne, that I have worked with, for contributing to the creation of a motivational and inspiring working environment. I especially want to thank dr. Tor Skeie and dr. Tarik Čičić for providing valuable input related to my thesis. Further, I want to thank dr. Thomas Sødring, Amund Kvalbein and Audun Hansen for proofreading my thesis and Viktor Eide for providing LaTeX advice as well as source-code for the preparation of this thesis. I also want to thank Amund Kvalbein for an interesting and inspiring cooperation on several papers, and Mete Yilmaz and Necdet Uzun whom coauthored the RPR tutorial (paper I) together with prof. Gjessing and myself.

Graduate students whom I have worked with or supervised during my thesis work have also been a source of inspiration and feedback on various aspects related to my research. My thanks goes to Petter Teigen, Kjell Ivar Haugnes and Bjørnar Libæk.

During my thesis work, I had the opportunity to work together with many of the members of the IEEE 802.17 Resilient Packet Ring working group. With many of them being experts within their fields, this was a very rewarding opportunity. I want to express my thanks to this large group of individuals for the many interesting discussions as well as good times.

A thanks also goes to the open source/free software community for making and maintaining a comprehensive set of tools, of which, many have been used in the preparation of this thesis as well as for the preparation of the individual papers in Part II of this thesis.

Finally, I want to thank my family for the good support during these years. In particular, I want to thank the love of my life, Jannicke, for taking on the ever increasing responsibility for our daughters Gry Helene and Kaja, as my working hours became increasingly longer during the last two years of my thesis work. Without your continued support Jannicke, this would never have been possible.

Bjørn Fredrik Davik
Fornebu, August 28, 2005

Preface

My thesis project has been organized as a standard 4-year PhD project, were 50% of the time is allocated to research, 25% of the time is allocated to teaching responsibilities at the Department of Informatics at the University of Oslo and an additional 25% of the time is allocated to mandatory courses at the Department of Informatics/University of Oslo.

My thesis work has been carried out in the period from March, 2001 to August 2005. In 2001, I was located at the Department of Informatics, while for the remaining period, I have been located at Simula Research Laboratory. My affiliations to both the Department of Informatics and Simula Research Laboratory has been with the Networks and Distributed Systems Departments.

Financial support for the project has been provided by the Department of Informatics, Ericsson and Simula Research Laboratory.

My thesis advisor has been prof. dr. Stein Gjessing.

Contents

List of Figures

List of Tables

Part I

Overview

Chapter 1

Introduction

1.1 Background

The OSI standard [1], defines a layered model for communication services and protocols for use by communicating nodes. In this model, the physical layer (layer 1), is responsible for transferring the individual bits of a message (e.g. a packet) between two locations over a specified communication medium. The second layer in the OSI model – the data link layer, is responsible for coordinating access to the physical layer, as well as performing functions such as packet delineation, error detection/correction, link-level addressing/switching and link-level flow control.

The task of coordinating access to the physical layer, as well as performing the tasks of packet delineation, error detection/correction, link-level addressing/switching and link-level flow control are usually referred to as Media Access Control (*MAC*). A specified or standardized method of performing the media access control is referred to as a MAC protocol or MAC method.

Since the late 1960's, we have seen a multitude of different architectures for computer communications [2]. Starting in the late 60's/early 70's with networking technologies and associated MAC protocols such as the Newhall Ring [3] on the wire-line side and ALOHA [4] on the wireless side, progressing to Ethernet [5] and Ethernet CSMA/CD [6, 7] in the late 70's/early 80's; Token Ring (IEEE 802.5) and FDDI [8–11], DQDB (IEEE 802.6) [12] and ATM in the late 80's/early 90's; 802.11 (WLAN) [13] in the late 90's; until we arrive at todays state of the art in communications solutions, with commercially available solutions such as multi-gigabit wire-line Ethernet and sub-100 Mbit WLAN.

In current state of the art commercial switches[1] and routers, there are products available that performs packet processing and switching at line-rates up to 40Gbit/s. Today, researchers envision possibilities for packet processing in the electrical domain at line-rates up to 160Gbit/s [15] within the next few years.

Regardless of whether the switching of traffic is performed in the electrical or optical domain, and whether the transfer of data on the physical layer is on a wire-line or wireless medium, the media access method used is crucial in obtaining the desired performance of the network. Thus, as we continually see advances in the physical layer methods for transporting data from A to B, we should expect a continued interest in the problem area of media access control protocols.

1.2 Media Access Methods and Fairness in Buffer-Insertion Rings

–Motivational Points for a Research Contribution

Our research department has previously participated in the development of the IEEE (Institute of Electrical and Electronics Engineers) standard for Scalable Coherent Interface (SCI) as well as reported on various findings related to the performance of SCI [16]. SCI utilizes the buffer-insertion principle (discussed in Section 2.6 on page 31) for media access on its uni-directional ring. The SCI technology is primarily targeted for small-scale communications networks, e.g. a cluster of workstations [17].

In early 2000, the IEEE Resilient Packet Ring (RPR) study group was formed, targeted at the development of a networking standard for MAN (Metropolitan Area Network) scale networks based on the buffer-insertion principle. With this development, we recognized an opportunity to make further contributions to the research area of media access protocols for buffer-insertion rings in general and Resilient Packet Rings in particular.

By joining the RPR working group, we would have access to a forum with a regular meeting schedule for the discussion of problems and ideas related to the development of MAC protocols for (RPR based) buffer-insertion rings. Given the solid commercial investments (in terms of man-hours, development cost of equipment and the cost of sending staff to the meetings, etc.) made by major equipment vendors in the standards development process,

[1] In this thesis, we follow the nomenclature argued by Seifert in [14, section 3.3], where a distinction between a switch and a bridge is just a marketing distinction introduced in the early 90's, where a switch is a bridge with a large number of ports, able to forward frames at wire-speed on all ports.

the "forum" (working group) was attended by many expert network designers as well as academic colleagues. Another advantage of joining the group, is that in addition to the traditional publication channels of network-oriented scientific journals and conferences, we would be provided with the opportunity to influence the contents of the later to become – Resilient Packet Ring standard.

Thus, given the opportunities presented above, a decision was made to invest department resources to investigate the problem area of MAC protocols in buffer-insertion rings. In addition to the motivational issues discussed above, other (more or less obvious) reasons for joining the RPR working group were: i) we wanted to, by publication of our findings, inform the networking community about the ongoing standardization effort; ii) we wanted to participate actively in the RPR project to contribute to the choosing of the best solutions in the presence of several alternatives; and iii) we wanted to establish a national knowledge base on the IEEE RPR initiative.

A side-effect of the third motivational point, in addition to this thesis, is the delivery of several Master theses in various RPR sub-topics, such as topology discovery [18], bridging [19] and multi-choke fairness[2] as well as a number of publications covering different topics within the Resilient Packet Ring problem area.

As in most PhD programs, the publication goal had a strong focus. While the publications from the work performed in the context of this PhD project has focused on various fairness aspects of the IEEE 802.17 Resilient Packet Ring, the general findings from our experiments should be applicable to buffer-insertion (as well as slotted) rings utilizing congestion-control methods to ensure a loss-free and fair communication over the ring. For a further discussion on buffer-insertion and slotted rings, refer to Chapter 2.6.

In the following, we start by briefly presenting some properties of the SONET/SDH and Ethernet communications technologies in Sections 1.3 and 1.4. Then, in Section 1.5, we present background information on the RPR standardization project. Next, we present our research goals in Section 1.6, followed by a a discussion on the research method we used in Section 1.7. Finally we conclude the chapter in Section 1.8, where we present the organization of this thesis.

1.3 The SONET/SDH Legacy

The use of ring-topologies for the interconnection of nodes is the prevalent method used in the area of MAN networks. Since its introduction in

[2]Ongoing work

1989, SONET/SDH has become a commonly used technology for long distance, high-speed transport of data in the Public Switched Telephone System (PSTN) [20]. On a SONET/SDH-based ring, when establishing a connection between two nodes on the ring, the bandwidth for this connection is guaranteed for the duration of its life-time. This is a good thing. The bad thing, is that if the capacity of this connection is not fully utilized, it cannot be used by other communications on the ring.

A strong property of the SONET/SDH rings, is that with the use of standardized protection protocols, in the event of single link or node failures, such networks are able to guarantee the restoration of connectivity within 50ms. Another strong property of the SONET/SDH rings, is that they allow for bidirectional communication on the ring, thus making shortest path delivery of data possible.

1.4 The Ethernet Legacy

Today, Ethernet[3] is the de facto standard for LAN (Local Area Network) networking. It has a massive-scale deployment, providing great cost of volume advantages. In addition, its massive-scale deployment provides a feedback-loop, helping in the effort to verify and improve all aspects of the protocol, technology and supporting software, under all conceivable operating conditions. Furthermore, Ethernet's support for variable packet sizes, provides an efficient (assuming a reasonable packet size distribution) means for encapsulation of IP-packets, in the Ethernet pay-load.

In the original Ethernet [5], any node on the network with data to send, can utilize the shared medium, as long as there are no nodes already sending data on the medium. This means that as long as the sum of the individual nodes' demand is non-zero, there will always be data available for transmission on the network. This resolves the problem of SONET/SDH, where (a temporary) inactivity of the individual nodes, leads to a reduced utilization of the network resources.

Once several sources have packets to send however, we have a contention situation, where the operation of the MAC protocol is crucial in obtaining a high (and fair) utilization of the transmission medium. Furthermore, the ability to provide any type of bandwidth guarantees in such a network, depends solely on the properties of the MAC protocol used (and that the nodes on the network adheres to it). In an Ethernet, although provisions have been

[3]Ethernet exists in a variety of different standard-versions, with a large span in the capabilities of the various versions.

made for supporting traffic priorities [21], the provisioning of bandwidth guarantees is still not supported.

In Ethernets comprising several networks connected by Ethernet switches, the Rapid Spanning Tree Protocol (RSTP) is used to ensure a loop-free topology, providing connectivity between all nodes in the network [21]. Further, in the event of link or switch failures, the spanning tree protocol recomputes the spanning tree, to restore (if possible) connectivity between the nodes on the network. Thus, RSTP provides a self-healing mechanism, ensuring connectivity between all nodes in the network, but with the constraint that loops in the resulting network graph (tree) is disallowed. A side-effect of this is that traffic for some flows may have to take longer paths, than would be the case if loops were allowed.

1.5 The Resilient Packet Ring Project

Being inspired by the success of Ethernet, while at the same time recognizing some of its weaknesses, an initiative was taken to standardize a new MAC protocol for packet rings. The ambition of the forces behind the initiative, hoped to combine the best of the capabilities of SONET/SDH and Ethernet, while gaining from Ethernet's cost of volume advantage.

Thus, in early 2000, a study group was started within the IEEE 802 group, responsible for maintaining and developing standards within the domain of LAN and MAN. The goal of the study group was to investigate the interest in the networking community for developing a new networking standard for ring-based networks.

Following a successful call for interest, in March 2001, an IEEE 802 working group was established, given the assignment of developing a standard for resilient packet rings and assigned the project number P802.17.

Some features considered important by the members of the working group were: bidirectional data transfer – allowing shortest path routing over the ring; fair allocation of network bandwidth; fast protection (sub 50ms) from single link or node failures; high link utilization; support for service differentiation between different service classes; and reuse of existing physical layer protocols (in particular those of Ethernet and SONET/SDH).

Other features were highly controversial, such as: the number of active links between the nodes on the ring; the use of congestion avoidance or congestion recovery, as the basic mechanism for regulating access to the medium; and the maximum size of packets that should be supported by the medium.

Thus as seen, in addition to the properties of fairness and bidirectional data transfer, the features desired from the Resilient Packet Ring were a

union of a subset of those provided by Ethernet and SONET/SDH (discussed in Sections 1.4 and 1.3 above).

Resulting from this starting point, about 3 years later (in mid 2004), the RPR working group concluded their work, followed by the release of the official IEEE standard for Resilient Packet Rings, 802.17-2004 in September 2004 [22]. Their end-product defines a standard for Resilient Packet Rings, defining a MAC protocol providing: bidirectional communication; fast (sub 50ms) protection; fair as well as guaranteed access to the medium bandwidth; shortest path routing of packets; support for service differentiation between different service classes; and finally – standardized interfaces to SONET/SDH and Ethernet physical layers.

1.6 Research Objectives

As discussed in Section 1.2 above, when this PhD project started, several opportunities were manifest for contributions to the research area of buffer-insertion rings as well as for the development and validation of ideas related to the Resilient Packet Ring standard. And as such, these opportunities were used as an initial loose set of targets for the author's PhD project. As the thesis work progressed, the goals were redefined and refined to the point were we ended up with the set of *Research Objectives* (*RO*s) as presented below.

These Research Objectives form a basis that can be used for a qualitative evaluation, to determine how well the research contributions, represented by this thesis and the individual research papers, meet the Research Objectives.

RO 1 *To investigate the problem area of flow control, congestion control and fairness in ring-based networks and to propose and discuss solutions for buffer-insertion rings, using Resilient Packet Rings as a test-environment for our proposed solutions.*

RO 2 *To contribute, by publication of results, to the dissemination of knowledge on the Resilient Packet Ring architecture, both within the network oriented research community as well as in the general networking community.*

RO 3 *To contribute to the development of the IEEE 802.17 standard and to aid the 802.17 working group in the development of a tested, working, error-free distributed fairness algorithm.*

RO 4 *To critically evaluate the algorithmic solutions chosen by the Resilient Packet Ring working group for the RPR fairness algorithm and to propose improvements or alternatives where appropriate.*

RO 5 *To evaluate applications of the RPR technology, to help ensure that there is match between a particular application's requirements and the functionality and associated guarantees provided by an RPR network.*

RO 6 *To develop a national knowledge base in the problem area of buffer-insertion and Resilient Packet Rings.*

1.7 Research Method Used

When investigating the performance of protocols in computer networks, several methods can be used [23, Chap. 3]. By use of analytical approximate models of the system under investigation, mathematical methods such as queueing theory are commonly used tools. Similarly, another well-known approximate method, is the use of a network simulator, where the behavior of the network protocol can be simulated at the desired granularity of events, depending on the performance metrics of interest. A third option, is to implement the protocol in a real device and to perform measurements on this device operating in a real network. A hybrid solution, where a trace based on measurements performed on traffic in a real network is applied to a simulator model, is another possibility.

Each of the methods may serve different purposes. The analytic / mathematical method is particularly suited for use when it is important to show an absolute property of a proposed or existing protocol solution. For instance, it may be important to prove that for a given set of boundary conditions, a proposed protocol is able to arrive at a steady-state within an analytically calculated maximum number of iterations. Another property of analytical methods, are that they are typically less resource demanding than their simulation and measurement counterparts and thus it is possible to obtain results faster. However, given the approximations that are usually undertaken, the general validity of the results may be diminished.

One interesting (and humorous) statement on the confidence of performance results obtained from analytical methods, is given by Raj Jain, in his book "The Art of Computer Systems Performance Analysis" [23]: "In general, analytical modelling requires so many simplifications and assumptions that if the results turn out to be accurate, even the analysts are surprised."

Similarly, a simulator can be used as a tool to strengthen or support the validity of findings from an analytical model (and vise versa), where such a model exists. The simulator can also be used to provide a set of results, making it possible to make both qualitative and quantitative claims on the performance of a proposed protocol for a given set of boundary conditions.

Based on the results for a given set of boundary conditions, it is often possible to predict the performance of the protocol in the general case.

The third option, of implementing the protocol in a real device, is often done as the very last step, to verify the performance of the protocol in a real network. The associated cost (in terms of man hours and equipment cost), often makes it necessary to perform as much of the protocol verification as possible by use of analytical and simulation methods. By this approach, the time spent on protocol implementation and verification in real hardware is minimized. In high-speed networking performance evaluation studies, the cost of performing performance evaluation by measurements is often prohibitably high, it may even be considered a function of the link-bandwidth, and as such is often undertaken at a very late phase in the product development cycle.

The hybrid solution, on the other hand, is particularly useful for comparing the performance of different solutions to some well defined problems/scenarios.

An interesting point, made by Bragg and Perros [24], is that the method of discrete event simulation is widely used to analyze the performance of low capacity communication networks, but is severely taxed when applied to higher capacity networks. Thus making it progressively more difficult to simulate a network at the granularity of individual packets in the system, as the bandwidth in the systems under observation increases. Furthermore, they point out that although analytical techniques based on e.g. queueing theory is a tool that have been successfully applied at an abstract level, when it comes to the detailed analysis of dynamic systems and the evaluation of feedback-based algorithms, these methods cannot be applied.

In this thesis, to obtain the desired level of accuracy of our experiments, performance evaluation by discrete event simulation is the main method used. In cases where we wanted to express particular properties of the protocols and algorithms under investigation, we have also applied analytical methods. The method of performance evaluation by measurements have not been a viable option, as this would require the use of resources not available to us. Further, if we were to undertake performance evaluation by measurements, it would most likely be used as a means of validating findings from our analytical and simulation experiments. Thus although we could have obtained a higher degree of confidence in our results, the added value of such measurements would probably be limited in the context of this thesis.

A decision was made at the start of this PhD project to use the commercial OPNET Modeler [25] discrete event network simulator environment for our simulation experiments. There are however no clear advantages on the use of OPNET Modeler over the use of other discrete event network

simulators like J-Sim [26] or ns [27]. When the project was started, there were no known publically available open source simulator models available for the simulation of an RPR (alike) network. Thus, regardless of the choice of discrete event simulator tool, we would have to implement the model from scratch.

1.8 Organization of this Thesis

This thesis is organized as an article collection.

To help the reader in achieving a better understanding of our contributions, we start by introducing the problem areas of flow control, congestion control and fairness in Chapter 2. This introduction presents a high level overview of the problems as well as a presentation of various previously as well as currently proposed methods of flow and congestion control.

Having completed this gradual introduction to the problem area, we are ready to present and discuss the most essential findings from our research contributions in Chapter 3.

Then in Chapter 4, we present our final conclusions on our contributions, with a particular emphasis on the fulfillment of the Research Objectives defined in Section 1.6, as well as some recommendations on possible directions for future work.

Finally, in Part II of this thesis, the seven selected individual research papers are presented.

Chapter 2

Background and Related Work

In any research area, the knowledge platform established by the efforts of one's predecessors in the area, is an important factor in achieving further advancement within the same research area. Even considering the 40 year history of the research area of flow control, congestion control and fairness in computer networks, the importance of the continual contributions to the field is undiminished as the types of networking technologies as well as their bandwidth capacity continues to increase. In fact, some claim that as networks grow larger, become more heterogeneous and carry more traffic, the congestion problem becomes more important than ever [28, 29].

Given the close relation between fairness and flow/congestion control, in the next Section, we introduce the concepts of flow control, congestion control and fairness.

Then, in Section 2.2, we discuss the origin of flow control in computer networks, before we introduce the problem area of fairness more thoroughly in Section 2.3. Then, in the succeeding sections, we present a literature review to some notable solutions developed during the history of the research field. In addition to a brief introduction to the principles of the various methods presented, we point to some of their weaknesses.

Having done this, we end the chapter with an introduction to flow control and fairness in Resilient Packet Rings before proceeding to Chapter 3, where we present our research contributions.

2.1 On the terminology

The principles of flow and congestion control are considered by many as separate solutions for separate problems that may occur in computer networks [20]. Flow control is often considered as a method where the sending rate of a

fast sender, is throttled to match the (temporarily) lower performance of its corresponding receiver (limited by e.g. buffer space or processing capacity). Congestion control on the other hand is considered to be the method used to reduce the number of packets in a network, so as to concurrently achieve maximum throughput and minimum packet delay [29]. That is, to maximize the ratio of throughput to delay, i.e. the resource power.

An aspect of flow and congestion control algorithms during times of congestion, is the resulting effective allocation of network resources to the individual packet flows. If all flows are considered equally important, then by use of a flow control algorithm during times of congestion, they should be provided equal access to the network resources available. The problem of providing equal or fair[1] access to network resources is known as a fairness problem. And thus the flow control algorithm used to obtain this in a network, is often referred to as a fairness algorithm. We discuss the fairness concept more thoroughly in Section 2.3.

In this thesis, we do not need the strict separation of the concepts of flow and congestion control as introduced above. Thus, with some minor exceptions called for by the discussion context, for the remaining sections of this thesis we use the term, flow control, as a collective description for both flow and congestion control. Before presenting the origins of flow and congestion control in Section 2.2, we first give a brief introduction to a high-level classification of methods for achieving congestion control. This classification depends on the previously mentioned concept of (resource) power.

The power concept is intended to provide an analytic expression, allowing a network analyst to evaluate the performance of a network in terms of its throughput/delay performance, as the load of the network varies. The analytic expression for power was introduced by Giessler et al. in [30] as shown in (2.1) below. Later, Kleinrock introduced a generalized power expression as shown in (2.2) [31], allowing an analyst to emphasize the relative importance of either throughput or delay, by the introduction of a throughput "weight".

$$power_{Giessler} = \frac{throughput}{delay} \tag{2.1}$$

$$power_{Kleinrock} = \frac{throughput^{\mathbf{r}}}{delay}, \; r : real \geq 0 \tag{2.2}$$

The behavior of throughput, delay and power (using a value of 1 for r) in a general computer network, not using congestion control, is illustrated in Fig. 2.1 below.

[1] Note that a fair allocation of network resources need not be an equal one.

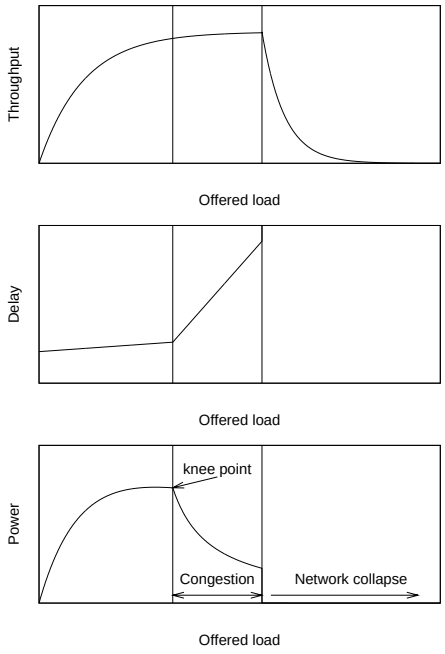


Figure 2.1: An illustration of throughput and delay as a function of offered load in a general network, ***not*** *using any method of congestion control. The top graph shows throughput as a function of the offered load, the middle graph shows the network delay as a function of the offered load, while the bottom graph shows power, i.e. the ratio of throughput to delay as a function of the offered load (specified in (2.2)). In the figure, we are using a value of 1 for r, meaning that in this illustration, throughput and delay are considered equally important in defining the optimum operational point. By definition, this point is found where power is maximized, represented by the first vertical line, crossing the three graphs at identical abscissa positions, and marks the onset of congestion. The graphs are an approximation of those found in [29].*

Several interesting observations can be done from this simple illustration. Firstly, as the offered load increases from 0, the resulting throughput is an approximate linear function. Then, as we approach the point marked by the first vertical line, we observe that the throughput starts to resemble an exponential ramp function, increasing asymptotically towards a maximum value. The delay on the other hand, continues to increase in an approximate linear fashion until we get to the first vertical line.

At this point, the growth rate of the delay increases significantly, while the growth rate of the throughput approaches zero. Thus at this point, we achieve the maximum value of the power function. This point, referred to as the knee-point of the system, marks the start of the region, where the network is said to be congested. Finally, from the point starting at the second vertical line and onwards, we have arrived at the region of network collapse, where the throughput decreases exponentially and the delay goes towards infinity.

When analyzing the performance of a given network, the setting of the throughput exponent r, should be so that the point, where we achieve the maximum ratio of $throughput^r$ to $delay$ (i.e. power), corresponds to the desired point of operation of the network. Thus, by use of a congestion control scheme, the goal is to keep the operation of the network such that

regardless[2] of the offered load, the point of network operation remains stable close to the point of maximum power.

Methods for achieving congestion control are usually divided into two types, namely "congestion avoidance" and "congestion recovery. With congestion avoidance, the goal is to avoid getting into a congested situation in the first place. With congestion recovery, the goal is to recover from a congested situation once this occurs [29, 32].

Regardless of the method (congestion avoidance or congestion recovery) in use, there must be some type of method to recover from a congestion situation, should it occur. In concluding this Section, we present an interesting observation made by Mankin and Ramakrishnan in 1991: "without congestion recovery, the network may cease to operate entirely (zero throughput), whereas the Internet has been operating without congestion avoidance for a long time" [32]. Thus, in the Internet, congestion recovery has been and continues to be the preferred method for handling of network congestion.

2.2 On the origin of flow control

In October 1965, an experiment was conducted where for the fist time (packet-based) communications between computers was demonstrated [33, 34]. The experiment was conducted between a computer at the MIT Lincoln Laboratory in Lexington, Massachusetts and a computer at the System Development Corporation in Santa Monica, California. In their experiment, Marill and Leiner used a simple (stop-and-wait) flow control protocol, where the transmitter was allowed to send one packet at a time and had to wait for an acknowledgement from the receiver before sending the next message.

A few years later, with the opening of the ARPANET network in 1969 [20, 35], the predecessor of today's Internet, the use of a flow control algorithm was included as a basal part for the handling of message flows between hosts on the ARPANET, interconnected via packet switches.

Following the successful establishment of ARPANET, in 1974 Cerf and Kahn came with their famous proposal – the Transmission Control Program[3] [36], which by means of an addressing and packet forwarding method and an end-to-end flow control protocol, allowed for the establishment of full-duplex, reliable communications channels between hosts on (separate) networks, connected via packet switches. The Transmission Control Pro-

[2]That is, assuming the offered load is equal to or greater than that at the point of maximum power.

[3]Not to be mistaken by the Transmission Control Protocol (TCP) to be introduced later.

gram was later split into two parts: the Internet Protocol (IP) specifying an addressing method allowing for an unreliable forwarding of datagrams and the Transmission Control Protocol (TCP), which enables the establishment of reliable full-duplex communications channels, by building on the service provided by IP and adding end-to-end flow control.

As the research on communication networks proceeded with incremental advances, both within the context of ARPANET as well as within other networking architectures, the research area of flow control was firmly established during the early 70's. With the introduction of LAN technologies (with Ethernet [5] being a prominent example), interconnected via slow long-haul serial links in the late 70's, the problem of congestion control became more important [28]. This was because these networks tended to become congested at their point of interconnection between the "high"-capacity LAN and the low-capacity long-haul link.

In the next Section, we discuss an important property of computer networks during times of congestion, that is, the property of fairly dividing the available network resources between the contending packet flows.

2.3 Fairness in Computer Networks

To obtain a fair sharing of resources in a computer network during times of congestion, two things must be in place: i) a fair (e.g. equal) distribution of the available bandwidth must be defined, and ii) a policy for this fair distribution of bandwidth must be implemented and enforced (i.e. in the form of a flow control algorithm).

In a network, the goal of achieving fairness while at the same time operating at the point of maximum power (discussed in Section 2.1) are desirable and orthogonal properties of a flow control algorithm and we discuss both by use of an example.

In Fig. 2.2 on the next page is an example of a network, where the demand for bandwidth over some links is larger than the links can support.

In the network, let us assume the use of a per-node flow control algorithm like e.g. Weighted Fair Queueing (WFQ is discussed in Section 2.4.3), using equal weights for all flows (i.e. all flows are considered equally important). Further, although not realistic in a larger network, let us for the simplicity of discussion assume that a flow is defined at the granularity of the pair of $\langle$source-node, destination-node$\rangle$ of individual flows in the network. This has the implication that for each switch in the network, the packets for each flow traversing the switch, is allocated to separate and dedicated buffers (of finite size).

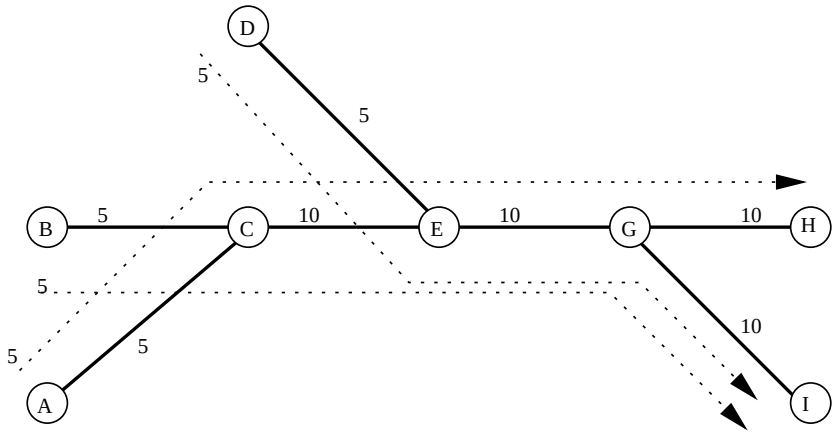


Figure 2.2: A general network represented by a graph. The vertices represents the nodes in the network, while the edges represent the interconnecting links. Each edge (link) has an associated bandwidth capacity, represented by the adjacent integer value. The data-flows and their corresponding demand for bandwidth in the network, are represented by the superimposed dotted arrows and adjacent integer values. A flow in the figure is specified in terms of its pair of ingress and egress nodes (e.g. flow (A,H)). A unidirectional link in the figure, is specified by the ordered set of its corresponding upstream and downstream nodes (e.g. l(E,G)).

With the network graph in Fig. 2.2, the aggregate bandwidth demand on link l(E,G) is given by the sum of the individual bandwidth demands of flows (A,H), (B,I) and (D,I), which equals 5+5+5=15. The link l(E,G) however, can only support a bandwidth of 10. Without any actions taken to reduce the traffic sent on the input-links of switch E, overflow of E's finite per-flow buffers will result.

However, regardless of the buffer-overflow situation, the use of the WFQ algorithm ensures that the capacity of the congested link is fairly (i.e. equally) shared between the three contending flows. Thus we achieve fairness, but clearly, we do not achieve maximum power. That is, given that we are loosing packets at node E due to overflow of its buffers, we are operating in the region of network collapse as illustrated in Fig. 2.1.

On the other hand, by use of a flow control algorithm like that used in e.g. Ethernet (discussed in Section 2.4.1), the amount of traffic accepted for the three flows (A,H), (B,I) and (D,I) is reduced so that buffer-overflow at node E is avoided. Given the previously stated equality of the individual flows however, the resulting[4] bandwidth allocation to the individual flows will not

[4]With the introduction of Ethernet flow control first in Section 2.4.1, for the clarity of discussion, we just state this as a fact. The unfairness property of Ethernet will become clear after having read Section 2.4.1.

be fair. Thus we achieve congestion control and operation at (or closer to) the point of maximum power, but we do not achieve fairness.

In concluding, it should be clear that the provisioning of fairness and maximum power during network congestion situations are desirable but orthogonal properties of congestion control. These properties received an increasing level of attention in the research community during the late 70's, and continues to be an important issue in the design of many flow control algorithms [22, 28, 29, 32, 37, 38]. For the remainder of this chapter, we look further into the problem of providing fairness and flow control in computer networks, in context of various network topologies and technologies.

2.4 Flow Control and Fairness in Arbitrary Topology Networks

In this Section, we present five notable flow control algorithms for arbitrary topology networks. With Ethernet being one of the best known networking technologies, we start by presenting Ethernet flow control. Next, in Section 2.4.2, we present Max-Min flow control, a theoretical fairness model, with various implemented proposals. Then, in Section 2.4.3, we present Weighted Fair Queueing (WFQ). Similar to Ethernet's flow control, WFQ flow control operates at the granularity of single switches only. Thus WFQ provides a per hop, rather than a network level flow control. Following this, in Section 2.4.4, we present proportional flow control, a flow control model, targeted to optimize the revenue for a network operator, based on the willingness of individual users to pay for access to network services. Finally, in Section 2.4.5, we present a more recent flow control algorithm for general topology networks, based on a virtual logical ring (MetaNet).

Having discussed flow control and fairness in arbitrary topology networks, in Section 2.5, we proceed to the discussion of flow control and fairness in ring networks.

2.4.1 Ethernet

In the original Ethernet [5], flow control was not supported and thus had to be provided by higher layer protocols if needed. With the evolution of switched Ethernet, supporting higher bandwidths and full duplex links, it became apparent that some means of flow control was needed to avoid excessive packet loss at the switches. A very simple example of this problem is shown in Fig. 2.3. In this scenario, the aggregate of traffic from sources S1 and S2 is larger than the capacity of the link to the receiver, D1. Without Ethernet

flow control, this would lead to overflow of the switch buffer associated with the output port connected to D1.

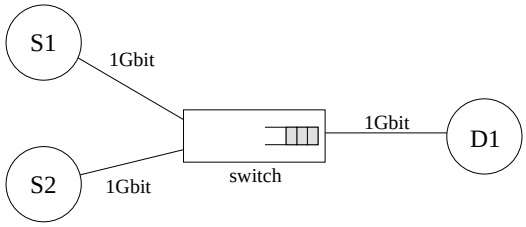


Figure 2.3: A switched Ethernet consisting of two active senders (S1 and S2) sending traffic to server D1. If the aggregate of traffic from S1 and S2 is higher than 1Gbit/s, the switch's buffer associated to the corresponding output port will overflow without Ethernet flow control. With Ethernet flow control, the switch will limit the sending of S1 and S2, thus preventing buffer overflow.

With Ethernet flow control (which is now a part of the Ethernet standard [39]), once the buffer occupancy exceeds a certain threshold (let's call this threshold *high*), the switch utilizes flow control messages to inform the senders attached to the respective switch ports to stop[5] transmission of packets. Once, as a result of the previously transmitted flow control messages, the buffer occupancy falls below another, lower, threshold (let's call this threshold *low*), flow control messages are sent on the switch ports, notifying the attached senders that they can resume transmission of packets. Given this "on-off" flow control of packet sources, with infinite demand senders and correctly dimensioned and configured buffers, we will have a cyclic variation in switch buffer occupancy, which is confined around the range defined by the *low* and *high* thresholds.

To achieve correct operation of Ethernet flow control, the size of the buffers and the configuration of the two thresholds, should be selected so that given the delay from the time a flow control message is issued, until the response is observable at the switch, the buffer should not overflow or become empty. If the buffer overflows, packets will be lost, and correspondingly, if the buffer becomes empty, capacity of the output-link will be wasted.

A weakness[6] of Ethernet flow control, is that it operates per switch only. Thus in a switched Ethernet, information about congestion status learned

[5] Actually, the semantics of the flow control messages is that it specifies that the receiving node must stop its transmission for a duration specified by the contents of the flow control message. Thus, if the sending must be stopped for a longer period of time, additional flow control messages must be sent. Similarly, a flow control message specifying a stop duration of 0, indicates that transmission can be resumed immediately (i.e. it is really a start message).

[6] Work is ongoing in the IEEE 802.3ar Congestion Management Task Force, addressing

by one switch (e.g. congestion at its downstream connected neighbor), is not propagated to the next (e.g. its upstream connected neighbor).

This means that corrective actions taken by the flow control algorithm in one switch, is based on local information only. A side-effect of this, is that it will take longer (as opposed to directly signalling the congestion state to the edges of the network) before the congestion state of the network is propagated to the network edges, resulting in a longer delay before the traffic sources are notified about a congestion state and adjust their sending rate.

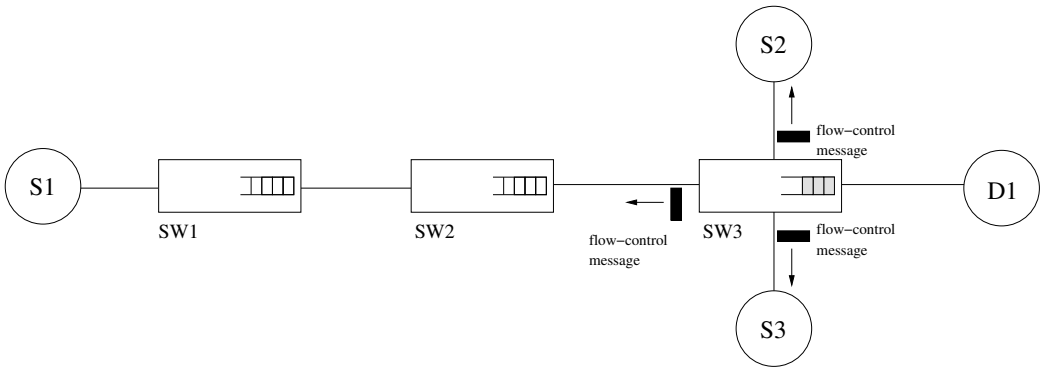


Figure 2.4: A switched Ethernet consisting of three switches (SW1-SW3) and three active end-nodes (S1-S3) sending traffic to server D1. Assuming equal link-capacities and that S1-S3 send at full speed, the buffer of switch SW3 will fill, resulting in the sending of flow control messages to S2, S3 and SW2. Flow control messages are not propagated further by SW2. Similarly, when throttled by the downstream switch (SW3), the buffer of SW2 will fill, resulting in the sending of flow control messages from SW2. Thus, we see a gradual buildup of buffer-occupancy in the network's switches. First when the buffers of switches attached directly to the end-nodes are filled, does the end-nodes throttle their sending rates (when receiving flow control messages).

This is illustrated in Fig. 2.4. In the figure is a simple network with three active senders (end-hosts), three switches and one receiver (server). The three senders, S1-S3, send traffic to server D1. Assuming an infinite demand of the three senders and equal capacity links in the network, the buffer occupancy of the three switches will gradually start to increase. First at SW3, when its buffer occupancy exceeds its *high* threshold, it sends flow control messages to its attached nodes S2, S3 and SW2. Then, as a result of the throttled capacity of its output-link, the buffer-occupancy in SW2 will start to increase. As a result, once its buffer occupancy exceeds its *high*

congestion control across an Ethernet network. It remains to be seen what the outcome of this effort will be.

threshold, SW2 sends flow control messages to its neighbors SW1 and SW3. Finally, when the buffer occupancy of SW1 exceeds its *high* threshold, it sends a flow control message to S1 and SW2. Thus at this point, all traffic from the end-hosts into the network has been throttled, and as the buffer occupancy of the three switches fall below the *low* threshold again (starting with SW3), flow control messages are sent, allowing the end-nodes to resume their sending activity. Hence we are back to the starting point, leading to a new network-wide cycle of increase/decrease behavior of the switch buffer occupancies. Thus, as a side-effect of the local operation of the Ethernet flow control algorithm, we incur increased[7] delay and jitter on traffic sent through the network.

Another problem of Ethernet flow control is that a traffic source sending traffic to several destinations, is not able to differentiate its throttling of traffic to different destinations, as the flow control messages from a switch, do not provide any information on the congestion state of its individual switch ports. Neither does a traffic source know the path of its traffic through the network to the destination.

Yet another problem is that the Ethernet flow control does not differentiate between different priority levels. Thus if a switch output port is blocked due to congestion in one of its priority queues, all input traffic is stopped by use of flow control messages, regardless of the level of its individual priority queues.

The problems discussed may appear rather serious (and they are), but the traditional way of configuring Ethernets, have been to add capacity to the network as the long-term traffic demand start to exceed the capacity of the network. The flow control algorithm on the other hand, is meant as a solution for reducing packet loss in the network resulting from short-term network transients. But as noted above, work is in progress addressing several of these problems.

An important improvement of Ethernet flow control is that loss of packets is avoided, and the buffering of packets, to a larger degree, is moved to the sending nodes (e.g. S1, S2) at the network edges, rather than in the network (switch buffers). Reduced packet-loss in the network, leads to significant improvements in performance of e.g. TCP, as we avoid reductions in throughput caused by time-outs in the TCP protocol as is otherwise incurred as a result of packets lost in the network [40].

Within a single Ethernet switch, fairness between the individual packet flows is decided by the scheduling algorithm used. In an Ethernet switch sup-

[7] As opposed to a global flow control algorithm, operating at the point of maximum power.

porting different traffic priorities, the use of a strict priority scheduling policy may lead to starvation of lower priority traffic. Another popular scheduling policy for Ethernet switches is Weighted Fair Queuing (WFQ) (discussed in Section 2.4.3), this may be used separately, or in combination with e.g. strict priority scheduling [41].

Let us consider the effect from the use of a particular queue scheduling policy, by use of the simple "parallel parking-lot" scenario shown in Fig. 2.5. The name come from the scenario's resemblance to a large parking-lot, consisting of several parallel parking isles. Each isle contains parking spots for a number of cars and has a separate access road connected to the same main-road leading towards the exit. When used in a performance evaluation scenario, the context is that the parking-lot is full and all cars are ready to leave concurrently (e.g. outside a movie theater, after the end of a screening).

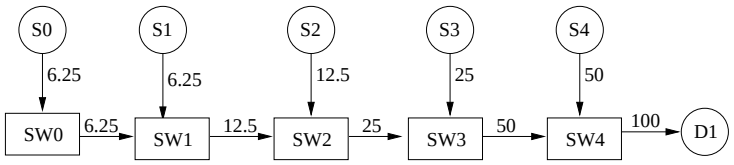


Figure 2.5: A "parallel parking-lot scenario" (also known as a "hot-receiver" or "hub" scenario) for a switched Ethernet with five sources (S0-S4) sending traffic to server D1. In the figure, S0-S4 can be considered the parallel isles providing parking spaces for local packets, while the links connecting S0-S4 to the switches can be considered access roads. The links interconnecting the switches constitutes the main road, leading toward the exit (D1). Finally, the switches can be considered the intersections between the access roads and the main road leading toward the exit. The network has equal-capacity links, with per-switch flow control, utilizing an output-port scheduling policy, ensuring per-switch fairness between contending flows on the granularity of input-port aggregates of traffic. The resulting link utilization of the individual links is indicated by labels above the links. The resulting per-flow bandwidth assignments is indicated by the labels at the source nodes. All figures are in units of percentages of the link-capacity.

In this scenario, we have six end-hosts (S0-S4,D1) interconnected by five Ethernet switches SW0-SW4, interconnected by equal-capacity links. All end-hosts (except D1) have an infinite bandwidth demand and send traffic to server D1 only (e.g. a "hot-receiver" scenario). Under the assumption of the WFQ scheduling algorithm, operating on the granularity of the individual input ports, using equal weights per input port, the local traffic from S4, receives 50% of the bandwidth of the downstream link, connecting SW4 to server D1 (the bottleneck link of the hot-receiver scenario). The remaining 50% is allocated to the upstream switch SW3, where the local end-host

(S3) get 50% of the available bandwidth fraction to its local traffic, and the remaining 50% of the same fraction is allocated to its upstream switch neighbor. Thus the resulting allocated bandwidth shares (shown in the figure) are clearly a function of the individual switches' spatial location in the network, relative to the hot-receiver.

Is this a fair sharing of bandwidth resources? The answer depends on the fairness definition in use. If considering all traffic sources as equals, regardless of their distance relative to the bottleneck link, they should all receive an equal portion of the bottleneck bandwidth (i.e. $bandwidth/5 = 20\%$). Thus according to this fairness definition, the achieved bandwidth allocation shown in Fig. 2.5 is not fair.

If we change the topology to a star, putting a switch in the center, as shown in Fig. 2.6, equal bandwidth allocations, regardless of spatial position, are ensured for the individual ingress aggregated flows[8], regardless of destination. The fairness property comes at a cost however, namely the changed structure of the wire connections as well as the existence of single point of failure (center switch). This extra cost is often unacceptable in a typical MAN scenario, where the existing fiber base is predominantly arranged in ring topologies, and a change in cable structure could require digging-up of ditches.

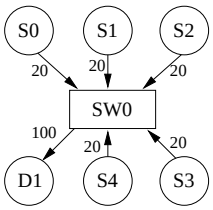


Figure 2.6: An alternative network configuration. By use of a star topology and a switch at the center of the star, fairness (based on the principle of equality of per input-port ingress aggregated flows) is ensured, regardless of spatial position in the network. However, the cost incurred is a change in wiring structure, as well as the existence of single point of failure (center switch). The resulting link utilization of the individual links is indicated by the associated labels.

2.4.2 Max-Min Flow Control

Max-min flow control is one of the classical reference models for the definition of fairness on a $\langle source, destination \rangle$ granularity of flows in a computer

[8]That is, the per input-port aggregates of traffic.

network [42, 43]. The idea is to "maximize the network use allocated to the flow with the minimum allocation" [42].

In terms of an iterative algorithm, this can be conceived as a method where, for each flow crossing a bottleneck link: the bandwidth allocated to the flow can be increased to the point, where a further increase, would lead to a reduction in bandwidth for another flow receiving an equal or less bandwidth share than the first.

If we go back to the example network shown in Fig. 2.2, how will this affect the allocation of bandwidth resources between the contending flows? If we consider bottleneck-link $l(E, G)$ (with a capacity of 10), the flows (A, H) and (D, I) can be allocated a capacity limited upwards to 10/3 each, without affecting the magnitude of flow (B, I) which has an equal demand ($\frac{10}{3} + \frac{10}{3} + \frac{10}{3} \leq 10$). If they are allocated a higher capacity than this however, this would require a decrease in the bandwidth allocation to flow (B, I), and would thus violate the previously stated max-min fairness constraint.

Many implementations have strived to implement the rule-set specified by the max-min fairness reference model. Variations of these implementations support such features as traffic priorities and dynamic traffic demands in a distributed environment.

The scheduling of packets over a link for contending flows can be expressed by a conflict graph, i.e. a graph representation of conflicts in the scheduling of network resources. An interesting observation by Bar-Noy et al. is that the computation of exact max-min fair rates for a conflict graph is NP-hard in the general case, while it is possible in polynomial time for some special classes of conflict graphs [44].

2.4.3 Weighted Fair Queueing and Generalized Processor Sharing

The Generalized Processor Sharing (GPS) approach [45, 46] for sharing of bandwidth of a congested link (e.g. one output link of a switch) was presented by Parekh and Gallager in 1992 and is a natural generalization of uniform processor sharing [47].

A very similar approach to that used by GPS, known as *Weighted Fair Queueing* (WFQ), was presented earlier by Demers, Keshav and Shenker [48, 49]. The WFQ mechanism by Demers et al. solved the problem of Fair Queuing proposed by Nagle [50, 51]. In essence, the problem was that a flow with a larger mean size of packets, could achieve a higher portion of the capacity of a switch's output link, than a flow with a smaller mean packet size. We will keep the discussion in this Section, within the context of GPS. In

a later work by Parekh and Gallager, they extended their models to analyze networks containing more than one node [52].

The goal of GPS, when used together with the leaky bucket admission control method [53] (discussed in Section 2.7.2), is to provide worst-case performance guarantees in terms of throughput and delay. In their proposal, a GPS server is assumed to be work-conserving[9] and operating at a fixed rate, r. Their analytical model is relatively simple and elegant: for each flow i through the congested node, each flow is assigned a weight ϕ_i, so that each flows gets its proportional share, g_i, of the available bandwidth, r, as shown in (2.3) below.

$$g_i = \frac{\phi_i}{\sum_{j \in [flows]} \phi_j} r \tag{2.3}$$

The GPS model is however more of a theoretical construct than a practical one. It relies on some impractical assumptions such as the ability of the server to divide its capacity infinitesimally between multiple flows.

Thus the authors present an approximation to GPS, termed *Packet-By-Packet Transmission Scheme (PGPS)*, where the rate control scheme operates on the granularity of the size of (variable sized) packets received. Under assumptions of a maximum packet size, the authors are able to show the worst case PGPS deviation from flow guarantees when compared to those of their theoretical GPS model. Finally, the authors propose a practical implementation of the PGPS scheme, which they call *Virtual Time Implementation of PGPS*.

As stated above, the GPS scheme needs to be complemented with a (leaky bucket) access control scheme, throttling a flow i's average flow rate to that provided by the GPS server, g_i. If over time, the average flow rate of flow i is larger than the rate, g_i, by which it is being serviced by the GPS server, this may lead to packet loss at the server as the associated flow queue may overflow. Furthermore, if the number of flows being served by the GPS server varies, according to (2.3), so does the individual flow service rates (assuming $\forall j : \phi_j > 0$).

The number of queues required by a PGPS server equals the number of flows traversing the server. Then, for a server servicing n flows, utilizing the Virtual Time Implementation of PGPS, what are the implications for the scheduler? In the worst case, to determine which packet is next in line to

[9] Work-conserving is a property of the server, where, as long as there are packets awaiting service, the server will never be idle. Thus throughput capacity will never be wasted needlessly.

be scheduled on the output link, the PGPS scheduler has to examine the time-stamp of the head-of-line packets of n queues.

Thus clearly, this solution will not scale to a backbone IP-network, with flows defined at the granularity of a $\langle$src-ip, dst-ip$\rangle$ pair. It may however work at the granularity of a DiffServ Code Point value (DSCP), carried within the packet's DS field [54] in a DiffServ enabled IP-network.

2.4.4 Proportional Flow Control

From the perspective of a network owner, it should come as no surprise that when selling network connectivity (i.e. bandwidth), the goal will often be to maximize the profit. The basic concept of providing fairness in a computer network, is not in conflict with this goal. However, traditionally, the relation between the amount of network resources consumed by the user and the amount of money charged by the owner of these resources has not been explicitly expressed by the fairness model.

In [55], Kelly presents a framework which has attracted a lot of attention, containing a mathematical model expressing on one hand the optimization problem for a *network owner* to maximize, for a given set of user data-flows, i) the aggregate of the individual users' network utility (e.g. their rates) and ii) the network revenue. On the other hand, they present a corresponding optimization problem for the individual user, where the user is allowed to freely vary the amount of traffic added to the network, when charged, λ_s, per unit flow. The optimization problem for the user then becomes to maximize the aggregate utilization for the given charge. Further, the optimization problem has a given set of constraints, related to networks capacity and connectivity and charge, λ_s, per unit flow from the user.

By use of this model, and a particular utility function, they introduce a fairness model termed *proportional fairness*.

In summary, for a given class of utility functions and a given set of user flows, there exist a price vector that will maximize the user's and the network (owner's) utility as well as the network revenue.

The use of this framework is further demonstrated by Kelly et al. for two classes of rate control/fairness algorithms [56].

It is clear that the use of a dynamic pricing mechanism as a means of obtaining fair access to network resources during congested periods appear to have a definitive merit. A lot of research have been undertaken in this area, but also, a lot remains to be done [57].

2.4.5 MetaNet

In this Section, we present flow control and fairness in MetaNet [58]. However, as MetaNet builds on the principles of deflection routing [59], we start by briefly introducing the principles of deflection routing, before proceeding to the presentation of MetaNet.

In the deflection routing method proposed by Maxemchuk, a regular network structure called the Manhattan Street Network (MSN), consisting of an even number of rows and columns, has a node placed at each intersection between a row and a column. An example of a Manhattan Street Network is shown in Fig. 2.7. Each node has unidirectional links connecting it to its nearest row and column neighbor. For nodes at the edge of the network, the connections may wrap around and connect the node to the node furthest away, located at the same row or column. Thus we have logically interconnected rings, where nodes in the same column are connected in a ring and similarly, nodes in the same row are connected in a ring. If a packet enters a node on a link, and its preferred output link (given by the routing algorithm in use) is busy, the packet is immediately transmitted on the other link (in a buffer-less node). In a buffered node, the packet is immediately forwarded if the link buffer is full.

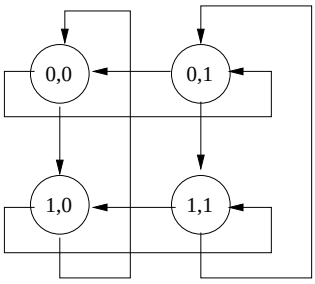


Figure 2.7: Small Manhattan Street Network. The structure is regular like the streets and avenues of Manhattan, hence the name. The unidirectional links along the rows and columns form rings, interconnected at the nodes. If a packet enters a node on a link, and its preferred output link (given by the routing algorithm in use) is busy, the packet is immediately transmitted on the other link (in a buffer-less node). In a buffered node, the packet is immediately forwarded if the link buffer is full.

MetaNet is a loss-free variant of the deflection routing method for arbitrary topology networks. Mayer, Ofek and Yung propose a fairness algorithm for the MetaNet, providing minimum bandwidth guarantees [60]. In MetaNet, to ensure the forward progress of packet, a spanning tree is maintained at all times. Based on this spanning tree, a virtual ring is constructed,

which is the ring formed by traversing the spanning tree on an Euler-tour[10] (a.k.a Euler-cycle). As a result of this, the links in the network are divided into tree-links (those included in the spanning tree) and thread-links (those not included in the spanning tree). Based on the concept of the virtual ring, the buffer-insertion principle (discussed in Section 2.6) is applied, resulting in priority of traffic already on the ring.

The operation of the routing method is such that once a flow's packet arrives at a node, and there is no conflict with other flows' packets at the same node, the output link (either a tree- or a thread-link) that provides the most progress (according to established virtual ring), towards the destination node is chosen for the further forwarding of the packet. If that link is busy, another path, ensuring forward progress towards the destination node is chosen.

If a node, due to transiting traffic received from upstream nodes on the virtual ring, is unable to send local traffic on the ring, the flow control algorithm is invoked. By use of back-pressure messages, travelling upstream along the virtual ring, upstream senders are choked, by use of so-called *regulate* control messages. Upon reception of a *regulate* control message, the node is allowed to send an additional amount of data restricted upward to one quota (the quota size Q, is a network parameter). As a result, when at a later time the node becomes uncongested, it may itself send one quota on the ring. Following this, the node sends an *unregulate* control message, allowing the upstream nodes to continue their transmission.

In their proposal, Mayer et al. do not include support for service differentiation, although it appears reasonable to believe that the scheme could be generalized to include this. Another apparent problem with this scheme, is the risk of the reordering of packets on their way towards the destination. As every individual node performs forwarding of packets depending on the status of the individual candidate output links, there is always a risk that some packets may use longer time on their path from sender to receiver. By this, reordering of packets may occur.

2.5 Flow Control and Fairness in Token Rings

In this Section, we present a single, well known flow control method for ring networks, namely the token-based flow control method, used in several more or less well known ring technologies. After having presented the token-based flow control method, in Section 2.6, we proceed to the discussion of flow control and fairness in buffer-insertion rings.

[10] A path through a graph which starts and ends at the same vertex and includes every edge exactly once [61].

In token ring networks, access to the shared communication medium, the ring, is controlled by the use of a (single) circulating token. In a token ring, when a node wants to send data on the ring, it first has to acquire ownership of the token. Once the ownership of the token is secured, the node can send data for a period of time, before passing on the token to its downstream neighbor. If a node has no data to transfer, the token is forwarded immediately. Thus the token functions as a primitive, providing access to the shared communication medium by mutual exclusion, i.e. the holder of the token is ensured exclusive access to the ring, while the other nodes are excluded. This concept was introduced very early, in the so-called Newhall Ring, by Farmer and Newhall [3]. The interest in this access method was renewed in the late 70's and resulted in two distinct token ring standards published in the mid/late eighties: the IEEE 802.5 Token Ring and the ANSI Fiber Distributed Data Interface (FDDI). In addition to fair access to the communication medium (the ring) and support for different traffic priorities, an enhanced version of the FDDI standard, named FDDI-II, allowed for circuit-switched transfer of data. While the 802.5 version operates at a transmission rate of either 4 or 16Mbit/s on twisted-pair cables, the FDDI standard is specified for a data rate of 100Mbit/s using optical fibers.

Numerous protocol proposals for packet rings, utilizing the token-based access method exist. For an overview, we refer the reader to e.g. [2, 10]. In this Section however, we limit the discussion to the IEEE 802.5 and FDDI media access protocols.

By limiting the time a node can hold the token, fairness between the contending nodes is ensured. In Token Ring 802.5, the data sent by a node is stripped from the ring by the sending node, after having circulated the ring once. Only when the data has been removed from the ring, can the token be passed on to the next node. The time required for the transfer of the token however, results in a waste of transmission opportunities by the nodes on the ring. Thus longer links will result in lower link-utilization.

This deficiency is rectified in FDDI where, once a node has finished its transmission of data, the token is immediately passed on to the next node. By this, enabling a quicker transfer of token ownership between the contending nodes on the ring.

Additionally, the throughput and delay performance of the ring (of both 802.5 and FDDI), is clearly related to the maximum token holding time for the nodes, as well as their aggregate demand for bandwidth resources. E.g. assuming all nodes have data to transmit for the entire period where they are allowed to (the maximum token holding time), this behavior will maximize the throughput on the ring (the fraction of idle time vs sending time is minimized). On the other hand, by this scheme, access delay is maximized.

This is because all other nodes on the ring, must wait for the arrival of the token, progressing node by node on the ring, as the individual nodes transmits their "quota". Thus the larger the demand of the nodes, and the longer the time nodes are allowed to hold the token – the longer the nodes have to wait between their successive sending periods.

2.6 Flow Control and Fairness in Buffer Insertion Rings

In the previous Section, we discussed the problem of flow control and fairness in ring topology networks, in terms of rings using token based access methods. In this Section, we start by introducing the principles of slotted and buffer-insertion rings. Following this, we focus on fairness algorithms for buffer-insertion rings. Then, in Section 2.7, we present flow control applied to a particular type of buffer-insertion rings, namely the IEEE Resilient Packet Ring [22].

In 1972, Pierce proposed a slotted media access method for rings, where the capacity of the ring is divided into a number of fixed-size slots circulating the ring [62]. In his MAC method, access to the slots is regulated by a busy-free bit, carried in the header of the slots. A major advantage of this access method, as opposed to the token-based method, is that it allows for uncoordinated access to the transmission medium. Thus allowing for concurrent transmissions on separate segments on the ring.

The media access method outlined above has one major deficiency. No restrictions are imposed on a single node's utilization of the available ring bandwidth. I.e. a single node may acquire all of the ring's bandwidth, thus resulting in starvation of the other nodes on the ring.

Another important weakness of the slotted-ring scheme, is that in its basic form, its does not efficiently allow for the transmission of variable size packets. For packets smaller than the slot-size, medium capacity will be wasted as a result of partially unused transmission slots. For packets exceeding the slot size, the packets have to be fragmented into several transmission slots before being transmitted onto the ring. Then at the receiver, the fragments must be reassembled to restore the contents of the transmitted packet. This fragmentation of packets contributes to a waste of bandwidth by both the creation and transmission of per fragment headers, as well as the waste when the last packet fragment does not fill the transmission slot completely. Additionally, the end-node processing of packet fragments requires greater processing capacity.

In 1974, a media access method, known as *buffer-insertion* was introduced by Hafner, Nenadal and Tschanz [63]. By introduction of a per node insertion-buffer, providing storage for at least one maximum sized package, the problem of allowing variable sized packets was overcome. But at the same time, the property of uncoordinated access to the transmission medium (ring) was retained. In fact, the buffer-insertion method can even be combined with the principle of a slotted transmission medium, allowing for immediate access to the transmission medium if required by a node (for an example of this, see Section 2.6.1). In principle, this means that the fairness algorithms discussed in context of the buffer-insertion rings, may apply to slotted rings as well.

Fig 2.8 shows the design of a node, utilizing the buffer-insertion principle, as it was presented in the paper by Hafner et al.

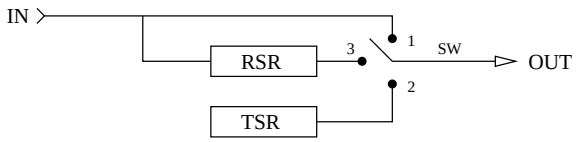


Figure 2.8: Original design of a node utilizing the buffer insertion principle. First presented by Hafner et al. in [63]. RSR (Receive Shift Register) and TSR (Transmit Shift Register) are shift-registers storing messages awaiting transmission on the outgoing link. RSR holds message(s) received from upstream nodes, while the node adds traffic from the TSR. TSR holds local traffic, while waiting for the RSR to become empty, so that local traffic can be transmitted onto the outgoing link. The position of switch SW decides whether the output data are fetched from the input, the RSR or the TSR.

In their paper, they evaluated three different strategies, which they named α, β and γ for (re)use of bandwidth on the ring: α) traffic is removed from ring by source node, β) traffic is removed by destination, γ) any node on the ring can reuse bandwidth occupied by data-messages marked as "received". The effect of strategy α is much like that of transmissions in a Token Ring, where only a node holding the token is allowed to transmit on the ring. In the case where the buffer insertion ring uses strategy α however, one may consider ownership of all available transmission "slots" as having separate tokens. Ownership of the transmission "slots" (the "token") is released, once the packet returns to the source node.

The difference between strategies β and γ seems somewhat vague. By use of strategies β and γ, the average "token holding time" is reduced as the packets and their corresponding "slot tokens", are stripped/released once the packets reach their destination. Thus the receiving node or any of its downstream neighbors, may reuse the bandwidth that would otherwise be occupied by the packet on the path from the destination back to the source

node. This effect is known as spatial reuse and may, depending on the traffic pattern, result in a higher aggregate throughput on the ring.

By taking advantage of traffic locality and destination stripping of traffic, the slotted and buffer-insertion rings **may** provide great increases in aggregate throughput as well as reduced access delay over that provided by token-based rings. However, the problem of starvation of nodes, caused by upstream nodes adding large amounts of traffic to the ring, remains. Considered too serious to ignore, the challenge then became to implement a method of ensuring a *fair* sharing of the ring's bandwidth resources between the contending nodes. Thus, in the following sections, we present several well-known methods for providing a fair allocation of resources between contending users in a slotted or buffer-insertion ring network.

2.6.1 Cyclic-Reservation Multiple-Access (CRMA-II)

The CRMA-II MAC protocol [64, 65] was introduced by van As et al. as a generalization of the CRMA protocol, presented by Zurfluh et al. [66, 67]. While the operation of the CRMA protocol was limited to bus-topologies, the CRMA-II protocol covers rings as well. Actually, in CRMA-II, the main focus has been on ring topologies. The operation of CRMA-II is based on a slotted ring principle, with support for ring access by buffer-insertion for packets exceeding the slot size and for packets requiring immediate access to the ring. The MAC protocol allows for spatial reuse by removing packets at their destination node.

The fairness mechanism used in CRMA-II is reservation-based, where during light loads, a node is not required to reserve bandwidth before sending data. During periods with heavy load, each node with data to send may request bandwidth. Based on the sum of requests from individual nodes and their amount of previously transmitted traffic, the "master-node" (termed *scheduler* in CRMA-II) calculates[11] and distributes a fairness threshold value. When received by the individual nodes, all nodes having transmitted an amount of data below the threshold are allowed to send data, while nodes having sent an amount of data in excess of the threshold, must refrain from sending (for a particular number of slots). In their paper, van As et al. also proposed a generalization of their scheme, whereby the introduction of a set of class-based thresholds, support for multiple service classes may be obtained.

One weakness of CRMA-II, is that it does not take advantage of traffic

[11]The process of calculating the fairness threshold is a non-trivial task and requires careful handling to achieve maximum throughput on the ring.

locality in its reservation scheme. Reservations are undertaken on a ring global granularity, where reserved cells can only be used by nodes holding confirmations of their previous reservations. This means that unused reserved cells, passing by a node not holding any confirmations, can not be utilized by the current node. This is OK as a means of ensuring that downstream nodes holding reservations gets to send their fair share. However, if the traffic the current node wants to send is destined to another node, and there are no intermediate nodes holding reservations on the path from the current node to the destination node, then the capacity of the passing reserved cells will be waisted along this path. One, very simple improvement to this deficiency, could be to always allow transmission using such cells, if the destination node is only one hop away.

2.6.2 MetaRing

The MetaRing architecture, proposed by Cidon and Ofek in various versions, is a quota-based allocation scheme for bidirectional data-transfers on a slotted or buffer-insertion ring, using hardware signals termed SATs [68–70]. The SAT-signal is propagated on the ring opposite of that for which it controls the data flow (e.g. for the ring carrying data flowing in the clockwise direction, the corresponding SAT-signal is sent on the ring carrying data flowing in the counter-clockwise direction). In the original version presented in [70], each node is allocated a transmission quota of l packets. In MetaRing, a node's equivalent of the Transmit Shift Register described above for [63], is called an output buffer.

If a node, upon reception of the SAT signal and since the reception of the previous SAT signal has either i) emptied the contents of its output buffer or ii) transmitted its allocated quota, l, it immediately forwards the SAT signal upstream. Furthermore, upon forwarding the SAT signal upstream, its quota is refreshed, allowing the transmission of l new packets. If, on the other hand, neither of the conditions in i) or ii) are fulfilled (due to excessive transmissions by the upstream nodes), the SAT signal is not forwarded before either of these conditions are fulfilled. I.e. the upstream active nodes must stop their transmissions once they exhaust their allocated quota. By this, starvation is avoided.

The original MetaRing architecture has some serious drawbacks with respect to ring access delay and throughput performance as the load of the ring increases. When the quota allocation, l, becomes too large compared to the number of nodes and their individual demand for capacity, the result will be an increase in ring access delay (i.e. the time a packet has to wait at the head of the node's output buffer before being transmitted onto the ring).

Similarly, a quota allocation that is too small, results in a lower aggregate throughput on the ring. These problems are the same as those discussed for token ring networks, in Section 2.5 on page 29. The authors address and propose improvements to these deficiencies in [69].

The essence of the improvement in terms of access delay, is that during a rotation of the SAT signal on the ring, each node requests, by increasing a counter field introduced in the SAT signal, a quota allocation in accordance with its own demand. Thus in effect, once the SAT signal returns to the node, the value of the counter field reflects the demand of all the nodes on the ring, and the quota allocated to the node is the fraction of the node's requested demand and the total demand (represented by the value of the counter field) multiplied by the maximum quota allocation. With this improvement, the authors show an analytical upper bound for access delay on the ring. However, it means that in the worst case, it will take up to two round trip times of the SAT signal, from when a node receives local data to be transmitted, until its gets its quota allocation.

The essence of the improvement in terms of throughput is with the introduction of a new signal, termed INFO. This signal is basically a hop-count, informing a node that it can transmit in excess of its allocated quota for traffic destined to nodes located at a distance equal to or less than the specified hop-count. The hop-count value of the INFO-signal is either increased (if the node gets to send at rates within its requested quota) or set to one (if the node does not get to send at rates within its requested quota) by each node it passes. Although, while this appears to be a significant improvement, it does contain an element of unfairness, as the node furthest upstream, receiving the signal (with a value greater than one) and having traffic to send will be able to starve its downstream neighbors competing for this "unreserved" bandwidth share.

2.6.3 Cyclic Queueing Multiple Access

Cyclic Queueing Multiple Access (CQMA) was presented by Schuringa, Remšak and van As to the RPR working group as an alternative fairness mechanism for Resilient Packet Rings [71]. It is a rate-based, proactive congestion control (i.e. congestion avoidance) mechanism. Calculation of fair rates for contending packet flows is done on the granularity of a $\langle source, destination, priority \rangle$ tuple. The fairness operation presented in their paper has two modes of operation, where the second mode is a simplified approximation of the first.

In the first mode, a $N \times N \times P$ array (where N is the number of nodes on the ring, and P is the number of priority classes) circulates on the ring, encapsulated in a control message. When received by a node, the node is

responsible for filling in information on its current demand for bandwidth for each of its flows and priority classes (i.e. the number of bytes queued in each of its virtual destination queues). Furthermore, for each iteration of the calculation cycle, each node calculates (based on the node's local copy of the circulating table) a corresponding set of rates for its individual flows.

In the second mode, three arrays of size N (or a two-dimensional array of size $3 \times N$) circulates on the ring, encapsulated in a control message. The first two arrays respectively contain the total demand for low and medium priority traffic, while the last contains the capacity unused by high-priority traffic (what the authors refer to as the "remaining capacity"). Transmission of the array is initiated by a designated node, which after the expiration of a timer, sets the contents of the control message to 0 before filling in information on its aggregate demand for the two traffic classes, as well as information on the "remaining capacity" of its outgoing link. The message is then forwarded upstream, where each respective node fills in (actually adds to the current value) its corresponding information in the three arrays. Having circulated the ring once, the control message now contains information on the demand and "remaining capacity" of each link and node on the ring. Thus, when received by the node for the second time, each node calculates the fair division of rates for the respective links and traffic priorities. This is done by each node on a per destination basis by taking the fraction of the individual and the total demand for bandwidth over each link, multiplied by the available bandwidth over the same link.

This mode of operation is very similar to that of the improved version of MetaRing [69], with the major difference that in MetaRing, it is done on a ring global level, rather than on the level of individual links

For the first mode, the worst case is that for each iteration of the algorithm, a new link becomes a bottleneck link and that for each priority class, $N/2$ nodes[12] have P or more flows crossing the link. E.g. for an arbitrary node i and link l_i connecting i to its downstream neighbor $i+1$ on the ring, the link l_i carries $P * N/2$ flows originating from node i, destined for nodes $j \in \{i+1, .., i+N/2\}$. The upstream neighbor of i, node $i-1$, sends $P*(N/2-1)$ flows across link l_i, destined for nodes $j \in \{i+1, .., i+N/2-1\}$. This configuration, where the number of flows crossing link l_i, sent by the progressive nodes, as we move one hop further upstream from node i, is reduced by P, is expressed in (2.4). Thus for an N-node ring, with P priority classes, where the maximum distance traversed by any flow is $N/2$ hops, we

[12]Remember we use shortest path routing on an RPR ring, thus, on a N node ring, where N is odd, no flows traverse more than $N/2+1$ hops. For a N node ring, where N is even, no flows traverse more than $N/2$ hops on the ring.

get the maximum number of flows f crossing link l_i as shown in (2.4) below.

$$f = P \cdot \sum_{i=1}^{N/2} i = \frac{P}{4}\left(\frac{N^2}{2} + 1\right) \tag{2.4}$$

Thus, for the worst case, when all links on the ring are a bottleneck, the fair rate calculation requires $N \cdot f = \frac{P}{4}\left(\frac{N^3}{2} + N\right)$ iterations. Which clearly is $O(N^3)$ complexity.

Although the second mode appears simpler than the first mode, it still has an upper bound on complexity of $O(N^3)$, equal to that of the first mode. The main reason for this complexity (for both modes) is that the algorithm works at the granularity of the $\langle source, destination, priority \rangle$ tuple. I.e. it requires the use of virtual destination queueing and rate control for all priority classes.

2.6.4 Distributed Virtual-Time Scheduling in Rings

The fairness algorithm introduced by Gambiroza et al. called Distributed Virtual-Time Scheduling in Rings (DVSR) [72], builds on the principles of Generalized Processor Sharing (GPS) algorithm by Parkekh et al., discussed in Section 2.4.3. In their paper, Gambiroza et al. also introduce a reference model for fairness in packet rings, called *Ring Ingress Aggregated with Spatial Reuse (RIAS)* fairness. In this Section, we start by describing their RIAS reference model, before describing their DVSR fairness algorithm, where they attempt to realize the properties of their reference model.

Fairness Reference Model (RIAS)

Gambiroza et al. presented a reference model, defining what they call *Ring Ingress Aggregated with Spatial Reuse (RIAS)* fairness [72]. In their model, they assume a traffic model, where active nodes have infinite demands, and traffic on the ring can be described in terms of fluid arrivals and services.

A RIAS fair allocation of network resources, is described in terms of a vector $\mathcal{R}$ of allocated fair flow rates R_{ij}, for individual flows, defined at the granularity of the flows' pair of $\langle ingress, egress \rangle$ points on the ring.

The essence of their model, is that a condition for the vector $\mathcal{R}$ to be RIAS fair, every flow (i, j) must have a bottleneck link with respect to $\mathcal{R}$.

To create a vector $\mathcal{R}$ containing a set of fair rates, the individual assignments of rates over bottleneck links (i.e. links, where the aggregate of traffic F, equals the capacity of the link[13], C) must be so that:

[13] In an RPR network, all links have the same capacity.

i None of the individual flow rates within a single Ingress Aggregated (IA) flow[14], must be allocated such that another flow within the same IA aggregate, already receiving an equal or less bandwidth share than that of the first flow, as a result of this allocation receives a reduced or zero bandwidth share.

ii No IA aggregates must be allocated a bandwidth share so that another IA aggregate, already receiving an equal or less bandwidth share than that of the first IA aggregate, as a result of this allocation receives a reduced or zero bandwidth share.

iii As a special case of ii), when a set of IA aggregates crosses two bottleneck links, the allocation of bandwidth for an individual IA aggregate, must be so that no other IA aggregate, already receiving an equal or less bandwidth share than that of the first IA aggregate, as a result of this allocation receives a reduced or zero bandwidth share. Note that in this scenario, a bandwidth share for an IA aggregate, is calculated as the sum of the IA aggregate's bandwidth share at both bottleneck links.

The process of allocating rates can be thought of as an iterative process, which is repeated until none of the individual flow rates can be increased further, without breaking the rules outlined in the list above. Thus, the result is a max-min allocation of rates (max-min flow control is discussed in Section 2.4.2), both between IA aggregates, as well as between individual flows within an IA aggregate.

The RPR standard adopts a weighted approximation of the RIAS reference model, where the MAC layer provides a weighted fair division on the granularity of IA flows over bottleneck links. The RIAS properties of fairness on the granularity of individual flows (i.e. flows with different egress points) within IA aggregates, as well as for a set of IA aggregates crossing several bottlenecks is not mandated by the standard.

Below, we describe their DVSR fairness algorithm, implementing their RIAS reference model.

DVSR Fair Rate Estimation

In DVSR, each node on the ring maintains a set of byte-counters measuring the number of bytes serviced (transmitted) for each IA flow serviced by the

[14]That is, the aggregate of flows with the same ring ingress point (node) but with different egress points (nodes).

node. Using the contents of these registers, accumulated during the current observation interval, a virtual time equivalent[15] is calculated for the individual IA flows. Based on these virtual time equivalents, the max-min fair division of bandwidth of the node's link, connecting it to its downstream neighbor, is estimated.

Below, we start by presenting a simple generic example before we describe the DVSR fair rate estimation process. For simplicity of the discussion, the description below differs slightly from that presented in Section V of their paper [72]. The result however, is the same.

In a scenario, where a number i of infinite demand IA flows are serviced by a node, using a non- or equally weighted bandwidth allocation scheme, each IA flow should receive an equal share of the bandwidth, R, available. Thus, the individual (i.e. fair) bandwidth share b_j allocated to each node is $b_j = R/i$. If however some of the IA flows have a demand, $b_k < R/i$, i.e. less than their fair share, the remaining nodes, having an infinite demand, can be allocated a corresponding larger fraction of bandwidth. Thus receiving a bandwidth share, $b_l > R/i$.

In DVSR, the actual per-node calculation of bandwidth allocations, is achieved by creating a sorted list of the byte-counts of the individual IA flows, serviced by the node. Thus, for a set of k IA flows, with a corresponding set $B = \{b_1, b_2, .., b_k\}$ of byte-counts, a list $\hat{B} = \{\hat{b_1}, \hat{b_2}, .., \hat{b_k}\}$ is created, sorted in increasing order of byte-counts. The complexity of this sort operation is $O(k \log k)$

For a congested link, starting with the first element, $\hat{b_1}$, in the list, for each element $\hat{b_i} < R/\hat{k}$ (where $R = \sum_{\hat{B}} \hat{b_i}$ and $\hat{k}$ is the number of elements in $\hat{B}$ and both are updated for each iteration of the loop), this element is removed from the list. This is repeated until the first element $\hat{b_i}$ is encountered, where $\hat{b_i} \geq R/\hat{k}$. Thus, when exiting the loop, R represents the sum of byte-counts for all flows, utilizing at least their fair share of the link bandwidth. Then by calculating $R/\hat{k}$, we get a byte-count estimate, representing what the system believes is the max-min fair division of bandwidth of the congested link. The result is that IA flows with a demand that equals or exceeds their fair share, must equally share the bandwidth not used by less demanding IA flows.

If this estimate is incorrect, and as a result the link becomes uncongested, the next time the algorithm is executed, a new max-min fair rate estimate is calculated. In this case, the estimate is based on the byte-count of the flow with the largest byte-count and the fraction of the time the link is idle.

In their paper, Gambiroza et al. propose an implementation where the

[15] In their implementation, the actual representation of this value is calculated as the individual flows' usage of the link's total capacity.

max-min fair rate estimate calculated by the individual nodes, is inserted into a rate message circulating on the ring. This message contains max-min fair rate estimates from all nodes on the ring. Thus every time the message passes by a node, the node updates its corresponding fair rate estimate, contained in the message. This results in a distributed algorithm, providing each node on the ring with global knowledge on the max-min fair rate restrictions in effect on the ring. Based on this knowledge, the individual nodes may calculate the rate values to use for their per destination traffic shapers, according to the rules specified by the RIAS reference model.

The process of calculating these rate values is not discussed in their paper. Let us consider the problem in terms of the ordered set of destination flows $\mathcal{D}$ originating from node i (ordered by increasing order of hop distance from i) and the corresponding ordered set of links $\mathcal{K}$ crossed by these flows. The set member $d_j \in \mathcal{D}$ is one if node i has a flow destined to node j, and zero otherwise.

For the set of links $\mathcal{K}$, there is a corresponding set of rates $\mathcal{R}$, containing for each element $k_i \in \mathcal{K}$ the corresponding amount of unused bandwidth $r_i \in \mathcal{R}$, to be divided fairly (according to the RIAS reference model) among the contending flows of $\mathcal{D}$, crossing link k_i. A flow $d_j \in \mathcal{D}$ crosses link $k_l \in \mathcal{K}$ if $j > l$.

The problem then becomes to find the member (link) $k_j \in \mathcal{K}$, with a corresponding available amount of bandwidth r_j, located closest to node i, and giving the minimum fair bandwidth allocation:

$$\forall k_j \in \mathcal{K}, \text{ find j, so that } R_{rias} = \frac{r_j}{\sum_{l>j} d_l} \text{ is minimized} \tag{2.5}$$

If there are more than one value of j, resulting in the same minimum value of R_{rias}, the one resulting from the minimum value of j is chosen.

Once found, remove from $\mathcal{K}$ the ordered subset of elements starting at k_j, remove the corresponding elements of $\mathcal{R}$, and remove from $\mathcal{D}$ the ordered subset of elements starting at the element following d_j. Finally, subtract R_{rias} from each remaining element in the list $\mathcal{R}$.

This process has to be repeated until the sets $\mathcal{D}$, $\mathcal{R}$ and $\mathcal{K}$ are empty. In the worst case, a node sending data to all other nodes on the ring, for each iteration of the loop, the link k_j returned from (2.5) is the last link in the set, thus, the sets are only reduced by 1. For a ring with k nodes, this results in $k-1$ iterations, each leading to a search among all the elements in the set, followed by a set of assignments. This leads to an upper bound on complexity of $O(k^2)$ for the assignment of RIAS fair rates, to the per destination flows, within a single Ingress Aggregated flow.

2.7 Flow Control and Fairness in Resilient Packet Rings

In the previous Section, we presented four flow control methods for buffer-insertion rings. Each supporting fairness in bidirectional buffer-insertion rings on various levels of flow granularity. Common for all the methods is that they support destination removal of packets, thus by enabling concurrent transfer of packets on non-overlapping ring segments, the aggregate network throughput achieved may be increased significantly, when compared to that provided by the mutual exclusive property of flows in token-based ring networks such as Token Ring or FDDI.

The following question may be posed: with such a set of eligible[16] candidates for fairness control in Resilient Packet Rings, is it worthwhile pursuing alternative methods for fairness control?

We will address this question in the next section, were we briefly discuss the RPR standardization project. Then, in Sections 2.7.2–2.7.5, we conclude this chapter by the discussion of various aspects of flow-control in Resilient Packet Rings.

2.7.1 The RPR Standardization Project

When developing a new (communications) standard, several aspects may typically be of importance for the forces driving the standardization process. One important aspect is the complexity of the proposed technical solutions. A second aspect is the performance of the proposed technical solutions. A third important aspect is the existence of intellectual property rights associated with the proposed technical solutions.

While the use of an available, proven solution for the provisioning of flow control in the RPR standard might have been a good technical choice, it could incur high financial risks on the project as it might incur financial obligations to holders of intellectual property rights associated with the chosen flow control method.

Thus in the RPR standardization project several different, more or less well proven and documented, fairness algorithms were proposed. One of the proposals was based on Cisco's Spatial Reuse Protocol (SRP) [73], with the intellectual property holders being one of the companies driving the stan-

[16]The CQMA and DVSR fairness algorithms were developed independently in the context of the RPR standardization project, thus it is reasonable to believe that these particular research efforts were motivated from the ongoing research activity resulting from the RPR project.

dardization project. During the course of the standardization project, the SRP algorithm was modified and another rate calculation method was developed to satisfy interest groups having slightly different targets for the RPR fairness algorithm.

Also, inspired by the success of another IEEE 802 standard, namely 802.3 Ethernet; an important goal for the RPR standardization project was to keep the technical solutions simple, so as to minimize the equipment cost for RPR products.

2.7.2 The RPR Node Functionality

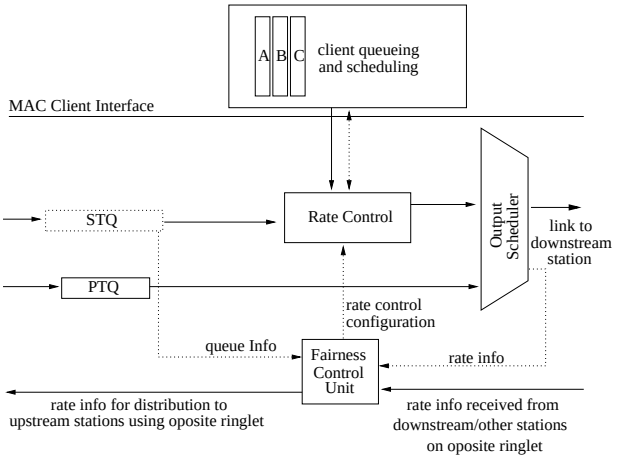
To provide the reader with a further understanding of fairness and flow control in Resilient Packet Rings, we provide a short introduction to the architecture and functionality found in a typical RPR node, illustrated in Fig. 2.9. The discussion uses parts of the text and illustrations, introduced in some of our papers.

The RPR standard basically defines the operation of the functionality located below the line termed MAC Client interface, that is, it defines the RPR MAC layer. The standard allows for two different node architectures – the 1TB and the 2TB-architectures, referring to the existence of one or two transit buffers in the transit path respectively. The optional presence of the STQ is indicated by the dotted lines in Fig. 2.9.

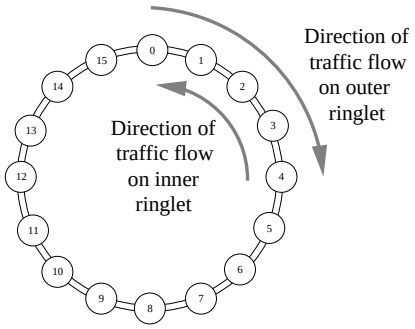
In the 2TB-architecture, high-priority transit traffic is assigned to the Primary Transit Queue (PTQ), while the remaining (medium and low priority) transit traffic is assigned to the Secondary Transit Queue (STQ). In the 1TB-architecture, all transit traffic is assigned to the PTQ buffer.

The difference in operation of the 1TB and 2TB-architectures, can be illustrated in terms of network power, as discussed in Section 2.1 on page 13. When utilizing the 1TB-architecture, more stringent restrictions imposed on the utilization of links in the network, places the system's knee-point further to the left than does the use of the 2TB-architecture in the same network. Using the 2TB-architecture, the extra transit-buffer provides increased flexibility, allowing for higher network throughput while maintaining the delay guarantees of high priority traffic. The increased flexibility comes at the cost of added node complexity however, e.g. in the form of an extra transit queue and increased complexity in the scheduling of packets, contending for the output-link.

The rate control of ingress traffic at a node is performed by the rate control block. In RPR, a token bucket scheme much alike that of the leaky bucket scheme presented by Turner in [53] can be used (with separate instances per traffic priority). The method presented by Turner uses a scheme



a) Generic node design for control of data flowing in one direction on an RPR ring



b) Example RPR topology

Figure 2.9: The generic RPR node design in the left figure, shows a node's attachment to the ring for transferral of data in the downstream direction and transferral of fairness messages (fair rate estimates) in the upstream direction. The solid arrows indicate the flow of data/control messages through the node. The dotted arrows indicate the exchange of control/configuration information between node internal function blocks. To control the sending of data on both ringlets, every node has two instances of the functional blocks and interconnections below the MAC client interface. I.e. in the example topology in the right figure every node has one instance controlling the data flowing on the outer ringlet and another instance controlling data flowing on the inner ringlet. The optional presence of the STQ is indicated by the dotted lines.

where packet transmissions are allowed as long as a counter, being incremented for each packet transmitted, stays below a threshold. This counter is then decremented periodically at a configured rate. In RPR, packet transmissions are allowed[17] as long as the corresponding counter equals or stays above a certain lower threshold. Then, upon transmission of a packet, the counter value is decremented by the byte size of the packet to be transmitted. Further, the counter value is increased periodically (at a per traffic priority rate) towards an upper threshold (rather than decreased), as long as there are packets awaiting transmission. If there are no packets awaiting transmission, the counter is not allowed to exceed the lower threshold.

While the rates of high and medium priority traffic is specified by node configuration, the rate control settings of low-priority traffic is provided by the Fairness Control Unit (FCU), executing the distributed fairness algo-

[17]Subject to some additional priority and rate constraints not to be discussed here.

rithm. Thus providing different rates at which credits are accumulated for the different traffic priorities.

When the demand for bandwidth over a link is higher than the available capacity, the link becomes congested. We can also say that the node upstream from this link is congested. The RPR standard defines a node as *congested* differently, depending on its transit queue architecture. If the node uses the 1TB-architecture, a node becomes by definition congested, once either i) its long term transmission rate of (local and transit) fairness-eligible traffic exceeds a configured threshold termed *rateLowThreshold* or ii) the access delay experienced by fairness-eligible traffic exceeds a configured threshold. A node using the 2TB-architecture becomes by definition congested once either i) its STQ occupancy exceeds a threshold termed *stqLowThreshold* or ii) the sum of (local and transit) fairness-eligible traffic exceeds its allowed portion.

For the remainder of this Section, when referring to the various thresholds, we use the terminology *low*, *med* (medium) and *high*, rather than explicitly stating whether we refer to the STQ occupancy or the transmission rate of fairness-eligible[18] traffic.

When a node has become congested, it is the responsibility of the rate calculation/fairness algorithm (termed rate calculation/fairness mode in RPR) of the FCU to calculate the node's fair rate estimate by use of either the *aggressive* or *conservative* rate calculation/fairness mode. The congestion situation for a node is checked and a new fair rate estimate is calculated with periodic intervals, every so-called *aging interval*.

While many of the previously discussed fairness methods for buffer-insertion rings (e.g. CRMA-II and MetaRing) are leaning more towards the principles of "congestion avoidance" by reservations, the two fairness modes of RPR belongs to the "congestion recovery" type of congestion control methods. The ultimate goal of the two fairness modes are the same – i.e. by use of congestion control methods, to provide a fair sharing of the bandwidth resources not reserved for or utilized by higher priority traffic; provide bounds on latency and jitter values for high and medium priority traffic; provide throughput guarantees for high- and medium priority traffic and to allow for a high utilization of the ring capacity.

The method used by the two rate calculation modes in trying to achieve these goals differs significantly and are presented in Sections 2.7.4 (*aggressive* mode) and 2.7.5 (*conservative* mode) below.

In a generalized view of the problem, the RPR congestion control mechanism uses a method where, by use of back-pressure messages, the congested

[18]Traffic that is rate controlled by the fairness algorithm at its ring ingress point.

node tries to relieve the congestion situation by limiting the ingress traffic at upstream nodes. These back-pressure messages contains the congested node's fair rate estimates and are advertised to upstream nodes at periodic intervals, every so-called *advertisingInterval*.

Before discussing the inner workings of the two rate calculation modes, we present an instrumental tool in achieving high utilization for Resilient Packet Rings, by taking advantage of traffic locality, the so called *Congestion Domains*, described in the next Section.

2.7.3 Congestion Domains

Both modes (*aggressive* and *conservative*) of the RPR fairness algorithm work with a concept known as a *congestion domain*. A congestion domain defines a consecutive collection of nodes, of which some or all contribute to a congestion situation for a given node/link.

The congestion domain is confined within a region specified by two boundary nodes. At one end of the region resides a node, denoted the *head*, which is attached upstream of the most congested link in the region. At the opposite end of the region resides a node, denoted the *tail*. Nodes upstream of the *tail* are considered as not being contributors to the congestion situation at the head. Fig. 2.10 illustrates the congestion domain concept for a network scenario containing two congestion domains.

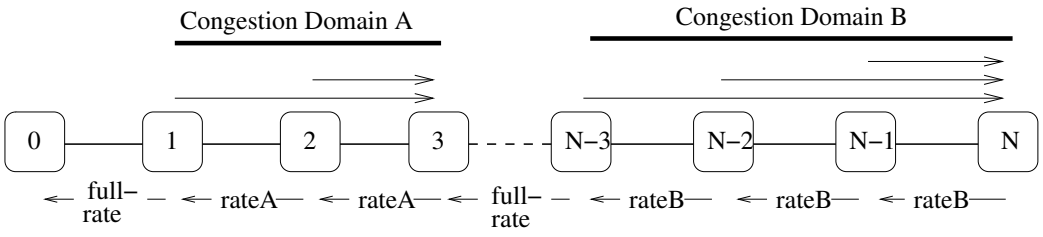


Figure 2.10: A network scenario containing two congestion domains, A and B. The propagation of fairness messages generated by the heads of the respective domains is stopped by their respective tails. Thus preventing the propagation of fairness messages, generated by the head of congestion domain B (node N), beyond its corresponding tail (node N-3). The tail (if not congested) signals, by use of full-rate messages, to its upstream neighbors that they are allowed to increase their sending rate of traffic.

The responsibility of the head is to maintain a rate value, used to provide a fair sharing of bandwidth among contending nodes sending fairness-eligible traffic over the bottleneck link. The calculated result, the fair rate estimate,

is distributed to upstream nodes in the congestion domain using special RPR packets, termed fairness messages (FCMs).

The tail on the other hand, is responsible for stopping the propagation of fairness messages, received from the head. By doing this, the intention is to maximize spatial reuse opportunities, resulting from traffic locality. With the operation of the tail however, some side-effects may be introduced that may lead to network instabilities and unfairness. We address this problem in the context of Chapter 3, where we describe our research contributions.

2.7.4 Aggressive Mode Rate Calculation

The RPR *Aggressive* fairness mode originates from Cisco's Spatial Reuse Protocol (SRP) [73]. It is a rather simple and elegant construct of $O(1)$ complexity, where the head of a congestion domain uses its own (low-pass filtered) send rate as an estimate of the fair rate. Thus, as a node becomes congested, and its send rate gradually drops towards 0, this gradual reduction in the send rate will be enforced (by the head) on the node's upstream neighbors by use of fairness messages. When, as a result of these rate messages, the amount of transit traffic received by the head decreases, more of the the capacity at the head's output-link will be available for use by the head. The head will then utilize this bandwidth, leading to an increase in the head's send rate. The increased send rate by the head is then distributed to the head's upstream neighbors, allowing the upstream neighbors to track the increasing send rate of the head. Ultimately, this will lead back to the point where the capacity of the head's output-link becomes fully utilized, and as a result, the head once again has to start decreasing its send rate. Thus we have a cyclic behavior where, in the ideal case, the cyclic rate increase/decrease behavior converges towards a narrow band of rates centered around the theoretical fair division of rates. The simplicity comes at a cost however and there are cases where the cyclic rate increase/decrease behavior results in reduced network performance, such as failure to converge towards the fair division of rates and reduced link utilization. We also address this problem in the context of Chapter 3, where we describe our research contributions.

2.7.5 Conservative Mode Rate Calculation

The paper by Wang et al. present the *Enhanced Conservative Mode* for the RPR fairness algorithm, which is the end-result of a joint contribution to the IEEE 802.17 working group [74].

This contribution[19] is now a part of the RPR standard where it is termed the *conservative* mode for rate adjustment in resilient packet rings.

The primary goals of this rate adjustment method, is to achieve, by use of a congestion *recovery* method, fair division of the ring bandwidth as well as providing high link-utilization. Although, to achieve high aggregate throughput in the network, steps are taken to ensure that the network does not operate too far left of the knee-point of the system (discussed in Section 2.1 on page 13), further care is taken to recover fast from a congested state (i.e. operation on the right-hand side of the knee-point) than is present in the *aggressive* fairness mode.

This makes the *conservative* fairness mode particularly well suited for use with the single transit queue option of the transit buffer architecture. It is also well suited for use with the dual transit buffer architecture where the secondary transit queue is of limited size.

In the *conservative* fairness mode, once a node becomes congested, actions are taken to i) keep the STQ occupancy (termed $|STQ|$) between two configured thresholds (2TB-architecture) or ii) to keep the aggregate rate of fairness-eligible (termed R_{fe}) traffic between two configured thresholds (1TB-architecture).

As the name implies, rate adjustments are performed in a *conservative* manner. That is with the exception of the detection of a serious[20] congestion situation, rate adjustments are performed only when the congested node decides (i.e. believes) that its previously issued rate adjustment has taken effect. That occurs with the expiration of a timer with a period termed *Fairness Round Trip Time (FRTT)*. The FRTT timer is discussed further in a separate Section below.

Thus, when the FRTT timer has expired and the congested node inspects its congestion status, one of three things occurs: i) the fair rate estimate is increased; ii) the fair rate estimate is decreased or iii) the fair rate estimate is left unchanged.

Action i) is performed to avoid (further) reductions in link utilization when the congestion ($|STQ|$ in the 2TB-architecture or R_{fe} in the 1TB-architecture) is below the *low* threshold.

[19]There are some minor discrepancies between some of the claims in their paper and the actual implementation of the *Conservative* fairness mode in the RPR standard. One example of this is the initial value of the local nodes fair rate estimate, L. In their paper, L is set to its maximum allowed value, while in the standard, it set to 0, requiring a gradual ramp-up before reaching the maximum allowed value. Another example is that the calculation of active weights is not mandatory for the *conservative* fairness mode.

[20]Upon detection of a serious congestion situation, the fair rate estimate is decreased immediately.

Action ii) is performed to reduce the severity of the congestion, when the congestion exceeds the *medium* (for the 2TB-architecture) or *high* (for the 1TB-architecture) thresholds.

Action iii) (nothing) is performed, when the congestion is between the two thresholds: $|STQ| \in [low, med]$ (2TB-architecture) or $R_{fe} \in [low, high]$ (1TB-architecture).

In summary, a choice between the three actions is made, based on the comparison of the current congestion status (i.e. STQ occupancy or aggregate rate of fairness-eligible traffic) to some pre-configured thresholds.

The adjustments performed in action i) consist of a simple adjustment of the fair rate estimate, based on its current value and the difference between the available and the utilized bandwidth for fairness-eligible traffic. The adjustments performed in action ii) consist of a simple adjustment of the fair rate estimate, based on its current value.

To conclude – the *Conservative* fairness mode is relatively simple, with a complexity of $O(1)$. Compared to the *aggressive* fairness mode however, its state machine is more complex, requiring both more state in the form of pre-configured parameters and state-variables in addition to more calculations. The *Conservative* fairness mode also requires the periodic measurement of delay incurred on rate adjustments, for the setting of the so called Fairness Round Trip Timer.

Fairness Round Trip Time Calculation (FRTT)

As discussed above, the period of the FRTT timer decides the frequency at which rate adjustments are made by a congested node using the *conservative* fairness mode. The FRTT timer period is adjusted periodically, based on measurements of the sum of delays incurred on the propagation of a type of RPR control packets, termed Fairness differential delay (FDD) Frames, reserved for this purpose.

When a node becomes tail, and at periodic intervals (while being tail), it sends such FDD frames, enabling the calculation of a new value for the FRTT period when received by the head.

This achieves a self-adjusting capability that adjusts the frequency of rate increases/decreases according to the amount of traffic in, as well as the physical size of the congestion domain.

2.8 Summary

In this chapter, we introduced some aspects of flow control and congestion control and fairness in general topology networks as well as in ring networks. We also discussed how, by the use of node local flow control (like e.g. Weighted Fair Queueing), fairness may be achieved at the node level. To achieve network level fairness, it is necessary to extend the flow control beyond that of a single node (like e.g. CRMA-II).

The goal of congestion control algorithms is to put the operation of the network near the knee-point of the network, where the power (ratio of throughput to delay) is maximized. A commonly used classification of congestion control algorithms divides congestion control algorithms into two classes: i) congestion avoidance algorithms, trying to avoid congestion in the first place and ii) congestion recovery algorithms, using methods to recover from congestion when this occurs. Common for both methods is that they require methods for recovering from congestion, should this occur.

We also discussed how, although congestion control and fairness are orthogonal properties of flow control protocols, it is often desirable that the operation of flow-control protocols result in a fair sharing of network resources.

In concluding, we presented some recent results from flow control algorithms in buffer-insertion rings before presenting a flow control method for a specific type of buffer-insertion rings, namely the recent IEEE 802.17-2004 Resilient Packet Ring standard.

In the next chapter, we present our research contributions within the area of flow control in buffer-insertion rings and a particular implementation of buffer-insertion rings – the Resilient Packet Ring.

Chapter 3

Contributions to the Research Area

In this chapter, we describe the seven selected papers from my PhD project, contributed to the research area of buffer-insertion rings, with a particular focus on fairness aspects of the recently standardized IEEE buffer-insertion ring – IEEE 802.17-2004 Resilient Packet Ring. The main topic covered is the problem of achieving fairness in such networks. Our contributions in the problem areas of all seven papers, have been published or accepted in various publication channels: one in an IEEE Magazine (Communications Magazine), two in major conferences organized by the IEEE Communications society (GLOBECOM'05 and ICC'05), and three in other conferences (ICN'05, CIC'05 and Networking 2005).

The various conferences have typically hard page limitations, constraining the discussion of the problems investigated, the solutions proposed and the corresponding performance evaluation. Thus four of the papers included in this thesis, are the full versions of the conference papers, presented in the form of technical reports. The three remaining papers included in this thesis, are identical to the conference version. In Sections 3.1-3.4 where we present the individual contributions, further information is provided on this issue where this is relevant.

In our approach to presenting our research contributions, we have followed a presentation style that is best described as a "top-down, bottom-up" approach. In this approach, each individual paper is assigned to one of four research areas (to be introduced below). Resulting from this, we start by introducing the problem area of Resilient Packet Rings. Then, we present our contributions on various fairness aspects of buffer-insertion rings in general and Resilient Packet Rings in particular. And finally, at the end, we present a user-application of RPR, which is made possible by the workings

of the fairness and service differentiation capabilities provided by the RPR standard.

Before we proceed to the discussion of the individual contributions, we provide a quick summary, presenting the individual research areas in somewhat more detail. A detailed summary of our contributions, organized by the four research areas, is provided in Section 3.5.

The first area, "An introduction to the RPR standard and its problem area", presented in Section 3.1, is dedicated to the introduction of the Resilient Packet Ring standard. In the context of this area, we discuss aspects of our extended[1] RPR tutorial.

The second area, "Analytical Model of Aggressive Fairness Mode", presented in Section 3.2, is dedicated to our work on providing an analytical model for the RPR fairness algorithm, or more precisely, on the *aggressive* fairness mode.

The third area, "Rate Calculation and Distribution", presented in Section 3.3, summarizes our work on analyzing and improving various performance deficiencies in Resilient Packet Rings. Some of these issues are commonly known in the RPR research community, while others are not so well known.

In the fourth and last area, "An RPR Application", presented in Section 3.4, we present a paper on a possible application of the RPR technology, i.e. RPR used in a DiffServ context.

After having presented our contributions to the research area in this chapter, in Chapter 4, we proceed to the presentation of our final conclusions along with a set of opportunities for future work.

Some other published papers, considering various Resilient Packet Ring issues, that the author has been involved with during his PhD project, can be found in [76–80]. These papers are not part of this thesis however.

3.1 An introduction to the RPR standard and its problem area

3.1.1 Paper I

Paper I, "IEEE 802.17 Resilient Packet Ring Background and Overview" is the extended version of our paper, "IEEE 802.17 Resilient Packet Ring Tutorial", presented in the March 2004 issue of the IEEE Communications Magazine [75, 81].

[1] That is, an extended version of our IEEE Communications Magazine RPR tutorial [75].

In this paper, we use a tutorial style presentation to introduce the main building blocks of the Resilient Packet Ring standard. We discuss some of the architectural decisions that were made by the RPR working group during the standardization process as well as presenting some of the underlying principles the standard is based on. The properties of the *aggressive* and *conservative* fairness modes are illustrated by use of some simulation results. Further, the delay properties of the three RPR service classes A, B and C are also illustrated by simulations.

3.1.2 Discussion (Paper I)

Several RPR tutorial papers exist. The tutorial by Yuan et al. [82], gives a good background on the motivation of the RPR technology, but does not provide an accurate description of the RPR *conservative* fairness mode. The tutorial by Remšak and Schuringa [83], provides a brief introduction to the RPR standard. As far as we know, our paper is the most accurate and correct tutorial style paper on RPR available.

3.2 Analytical Model of Aggressive Fairness Mode

3.2.1 Paper II

In our paper "An Analytical Bound for Convergence of the Resilient Packet Ring Aggressive Mode Fairness Algorithm", presented at ICC'05, we use an alternative analytical approach, based on control theory, to present an analytical model for the RPR *aggressive* fairness mode [84]. This paper can be considered an extension/improvement of our paper, "The Stability of the Resilient Packet Ring Aggressive Fairness Algorithm", presented at LANMAN 2004 [76] (not included in this thesis).

Based on our analytical model, we use control theory to show that the stability of the *aggressive* fairness mode for a given parameter setting is dependent on the size of the network. Further, we express an analytic bound for the maximum size of a network (or congestion domain), that can be safely controlled for a given value of an RPR node's parameter setting.

Using this method, rather than heuristics guidelines, we obtained a set of analytically based rules suitable as guidelines for obtaining a safe configuration of the *aggressive* fairness mode for a given network topology.

3.2.2 Discussion (Paper II)

In approaching the challenge of creating an analytical model of the RPR *aggressive* fairness mode, we found some useful hints on how to approach the problem. Firstly, in a paper by Bragg and Perros, a claim is made that analytical models based on queueing theory cannot be applied to feedback-based systems [24]. Secondly, in a paper by Yang and Reddy, another claim is made that a control theory that masters the behavior of computer network systems, has yet to be established [29].

Thus, to apply any of these methods in an RPR context, some simplifications and assumptions have to be done (while adhering to the cautionary note of the applicability of analytical methods presented in Section 1.7).

With this in mind and based on the operation of RPR's feedback-based rate control (fairness) algorithm, we decided to create a simplistic analytical model, utilizing a control theoretic approach.

In Section IV.B of our paper, we discuss the accuracy of our analytical model, and state that the deviation between our analytical and simulation results are believed to be caused mainly by scheduling and queueing delays on the path between head and tail in the congestion domain. After a closer investigation, it became clear that there exists an additional and more significant cause of deviation. Namely the periods, where the furthest upstream node in a congestion domain is allowed to send at excessive rates. This behavior severely impedes the stability of the *aggressive* fairness mode. In our paper, presented in Section 3.3.3, this problem is investigated, and a method to resolve this problem is proposed and evaluated.

In a recent paper by Shokrani et al. [85], an analytical model is proposed, which is somewhat similar to that proposed by us, to study the stability of the RPR Aggressive mode fairness algorithm. Their model applies to an RPR network with nodes utilizing the 1TB-architecture. In their paper however, a model is proposed where a condition for the validity of the model, is that the round-trip propagation delay of a congestion domain is always shorter than the aging interval (discussed in Section 2.7.2). The RPR standard restricts the allowed values for the duration of an aging interval as shown in (3.1) below.

$$agingInterval = \begin{cases} 400[\mu s], \text{link-rate} < 622Mbits \\ 100[\mu s], \text{otherwise} \end{cases} \tag{3.1}$$

Thus, given a ring using a link-rate of 1Gbit/s, resulting in a value for $agingInterval$ of $100[\mu s]$, this restricts the size of a congestion domain to $length < \frac{100 \cdot 10^{-6}[s] \cdot 2 \cdot 10^{8}[m/s]}{2} = 10^{4}[m]$. Such a size-limitation is clearly not

sufficient for a MAN network. Further, if adhering to their condition, the system under analysis do no longer act according to an "aggressive" behavior, because it will always have seen the complete effect of a previously issued rate-message, before sending a new. In fact, the system under observation behaves more like a "*conservative* mode system". Another problem with this approach, is that a weighting coefficient $0 < \alpha < 1$ is introduced. However, the relationship between this coefficient and the actual parameters of the *aggressive* fairness mode (i.e. $lpCoef$ and $ageCoef$) is not discussed.

Another attempt of evaluating aspects of the RPR fairness algorithm by analytical methods, is in a paper by Schupke and Riedl [86]. In their paper, queueing theory is used to analyze the delay properties of an RPR network. In their paper, an assumption is made that the ring is operating in what is referred to as a "normal" operation state, which is defined to be a state where the ring is not trying to recover from a congestion or unfair situation.

Such an assumption is however not realistic, as the RPR fairness algorithm is always seeking towards the congested state, to maximize utilization of the network resources. This behavior is a result of the RPR feedback-control system.

3.3 Rate Calculation and Distribution

In the following sections, we discuss a paper proposing an alternative fair rate calculation mode for buffer-insertion rings, evaluated within an RPR context, in Section 3.3.1. Then, in Section 3.3.3, a paper is discussed where we analyze and propose an improved method for the distribution of fair rate estimates in RPR. Finally in Sections 3.3.4 and 3.3.5, we discuss two papers evaluating certain performance metrics of our proposed modified rate distribution method.

Our paper [87], accepted at GLOBECOM'05, combines the contributions of our papers III and IV (technical reports) [88, 89]. In the combined conference version, the main problems addressed by the two technical reports are discussed, along with proposed solutions and a short presentation of their performance.

In this thesis, only the full length versions (i.e. technical reports) are included. This allows us to present as complete a set as possible of our findings of problems related to the (RPR) rate calculation and distribution, along with proposed solutions as well as an evaluation of their performance.

3.3.1 Paper III

The RPR standard currently contains two different operational modes for its fair rate calculation process. The first is known as the *aggressive* mode while the second is known as the *conservative* mode.

It is well known that in the presence of a so called unbalanced traffic scenario [90], where the head of a congestion domain utilizes less than its fair share of bandwidth, the *aggressive* fairness mode suffers from severe oscillations that results in reduced link-utilization of the congested link.

Several research groups have tried to solve this problem. Most have done this by proposing a fairness algorithm that does not fit into the framework given by the RPR standard.

In our paper, "Improvement of Resilient Packet Ring Unbalanced Traffic Scenario", we propose a novel rate calculation method for the RPR fairness algorithm of $O(1)$ complexity, which we have called the *moderate* fairness mode [87, 88]. Our proposed method utilizes state information available in the RPR standard to maintain a fair rate estimate, independent of the actual sending behavior of the congested node and the number of upstream nodes transmitting over the congested link. That is, by monitoring the STQ occupancy as well as the STQ growth direction, the local fair rate estimate is either increased or decreased within a dynamically maintained rate region. By successive iterations of the *moderate* fairness mode, the fair rate estimate as well as the rate region is centered around the theoretical fair rate value while keeping the mean STQ occupancy at the STQ *lowThreshold*.

Our method, while requiring somewhat longer convergence time than the *aggressive* fairness mode, resolves the problems for unbalanced traffic scenarios and works equally well, regardless of the sending behavior of the head of the congestion domain. In steady-state (given a stable load), our algorithm maintains an average *STQ* occupancy at the STQ *low* threshold and thus provides better[2] delay/jitter properties than the *aggressive* fairness mode fairness algorithm. Our method also inter-operates with the *aggressive* fairness mode. Further, we also showed that for the tested scenarios, our algorithm outperforms[3] the *conservative* fairness mode in terms of convergence time and link-utilization.

While focusing on a solution applicable for the 2TB (Transit Buffer) node architecture (discussed in Section 2.7.2), the method appears to be applicable for the 1TB node architecture as well.

[2]The *aggressive* mode fairness mode maintains an average *STQ* occupancy at the STQ *high* threshold (for converging scenarios).

[3]In terms of link-utilization, the difference between the *moderate* and the *conservative* modes is less significant once in steady-state.

In the 1TB architecture, rather than adjusting the fair rate estimate based on the transit-buffer occupancy and growth-direction, adjustments could be carried out based on the level of traffic rates as well as their corresponding growth direction.

In addition to the *moderate* fairness mode, this paper also proposes a measurement method to enable a node to determine whether or not it is utilizing its fair bandwidth share over its outgoing link. This allows for a construct where, based on a node's bandwidth utilization, different rate control methods may be used.

3.3.2 Discussion (Paper III)

In contrast to the CQMA and DVSR solutions discussed in Sections 2.6.3 and 2.6.4, our proposed *moderate* fairness mode requires no changes in the RPR node architecture, no new measurements of traffic in the nodes and no modification of the format or semantics of the RPR fairness messages. The complexity of the proposed *moderate* fairness mode is $O(1)$, just like the *aggressive* and *conservative* fairness modes.

The DVSR algorithm on the other hand, requires a sort operation with an upper bound on complexity of $O(k \log k)$ (where k is the number of flows on the granularity of $\langle ingress\ point, egress\ point \rangle$ on the ring). Additionally, the DVSR algorithm requires per node assignments for individual (i.e. per destination) flow rates within an IA aggregate flow, having an upper bound on complexity of $O(k^2)$.

The per-node complexity of the CQMA algorithm has an upper bound on complexity of $O(N^3)$, where N is the number of nodes on the ring. This complexity comes from the calculation where, based on the knowledge of all nodes' demand for bandwidth on the ring, each node calculates the rates to use for each of its individual flows (i.e. per destination flows).

Another method, proposed by Alharbi and Ansari [91], solves the problem for the case of the unbalanced traffic scenario, by applying the methods established by the DVSR algorithm. The proposed method is termed Low Complexity Distributed Bandwidth Allocation (LCDBA). Rather than performing the operation on the list structure proposed by the DVSR method, in addition to the per IA-based byte-counters, a variable is maintained, A, representing the aggregate of the individual IA flows usage of their allocated bandwidth.

At the end of the observation period, a new fair rate $F = \frac{C_u}{max(1,\tilde{N})}$ is estimated (where C_U is the bandwidth not reserved for high priority traffic). The value $\tilde{N}$ represents the sum of the individual IA flows' fractional usage of

the previously calculated fair rate estimate (calculated from the A variable). The properties of the method, provides a result, where the new fair rate estimate resides in the region $\left[\frac{C_u}{N-1}, C_u\right]$ (where N is the number of nodes in the ring). The complexity of the LCDBA algorithm is $O(1)$, but just as for the CQMA and DVSR flow control methods, the LCDBA method requires per destination scheduling in the RPR MAC client. In the paper, the LCDBA method is evaluated by discrete event simulation for two scenarios and results comparable to those of the DVSR method are obtained.

Yet another paper claiming to solve the problem for the case of the unbalanced traffic scenario is reported in a paper by Zhou et al, [92]. In this paper an alternative fair rate calculation method is proposed, to be used when a proposed testing method indicates that the demand of the head is less than its fair share (i.e. the case of unbalanced traffic). Secondly, a proposition is made to change the semantics of the fairness control message sent upstream, so that when received by an upstream node, the rate value in the rate message does not indicate a rate constraint in effect over a congestion point, but is rather an indication of the bandwidth available downstream, not used by the head.

There are several problems with this approach: firstly, in the case of unbalanced traffic scenarios, the resulting fair rate estimate does not consider dynamics in the traffic received from upstream nodes, rather the demands are considered equal and fixed. Thus the fair rate estimate will not converge if the demand of upstream nodes deviates from the assumed model. Secondly, measurements are used, which are based on occupancy of the head's stage queue[4] to decide whether to use the RPR *aggressive* fairness mode rate calculation method (discussed in Section 2.7.4) or the RPR-RE rate calculation method proposed in the paper. To support this, the implementation of a physical stage queue becomes mandatory, something that is not the case in the current RPR standard. In the RPR standard, the stage queue is described as "a logical construct that might or might not correspond to any physical structure" [22, pp. 92].

Lastly, the simulation experiments described in the paper covers two scenarios, leaving many aspects of the RPR-RE rate calculation method to be shown. The first scenario is an unbalanced traffic scenario with two active nodes. In this scenario, although the oscillations are limited, the link utilization is only ≈86% (for our *moderate* fairness mode, we achieved full link-utilization in a similar scenario with four active senders). The second

[4]In paper III, the correctness of the use of the head's send rate as a fair rate estimate is discussed in the context of their earlier paper [93], this discussion applies directly to their more recent paper [92].

scenario, illustrates the performance of the RPR *aggressive* fairness mode in a scenario with three active senders, producing an equal sharing of the congested link's bandwidth for the greedy senders (as should be expected).

3.3.3 Paper IV

The RPR standard works with a concept known as congestion domains, which was introduced in Section 2.7.3. A congestion domain is a consecutive collection of nodes of which some or all contribute to a sustained congestion state on a given link. By definition [22, pp. 262–263], when a node receives fairness-eligible transit traffic traversing the downstream congested link, at a rate lower than the fair rate over the congested link, it considers its upstream neighbors as not being contributors to the congestion situation downstream. When this happens, the node assumes the role as tail of the congestion domain. As part of this role, the congestion tail does not forward rate information, received from the head, to its upstream neighbors.

This behavior leads to some previously[5] unknown side-effects, where nodes not receiving information on downstream rate restrictions, may transmit excessive amounts of traffic. In short, this problem can be described as "tail-induced"[6] oscillations. In our paper, "Congestion Domain Boundaries in Resilient Packet Rings", we analyzed the outlined problem and proposed and evaluated a solution to this problem [89].

While the side-effect introduced is the same for both fairness modes, the degree of seriousness is different.

For the *aggressive* fairness mode, not evaluating its previous rate adjustment before making the next, the duration of the periods of excessive sending typically are longer, and occurs more often. This leads to an increase in convergence time for a given network size as well as reducing the size of the network that may be operated safely. For the *conservative* fairness mode, with shorter duration of these periods and a lower frequency of occurrence, the symptoms are mainly a somewhat increased convergence time as well as some degree of unfairness.

In our paper, we proposed a simple improvement, which for a certain class of scenarios doubles the size of the networks that can be safely operated for a given set of parameter settings. In other scenarios, the effects of our improvements are negligible. We have not found any scenarios for which

[5] We reported this behavior by e-mail to a subset of the RPR Working group on October 29, 2003. In the e-mail, we described the root of the problem briefly, along with a proposed solution. The solution outlined in the e-mail, differs from the one we present in our current paper.

[6] This is the term used by Robichaud and Huang in their paper [94].

our proposed improvement has yielded a performance, less than that of the original (standardized) algorithm.

This problem has been further explored with respect to throughput and delay performance for various simulation scenarios. This is the topic of our contributions in papers V and VI, presented in Sections 3.3.4 and 3.3.5 respectively.

3.3.4 Paper V

In our paper "Performance Evaluation and Improvement of Non-Stable Resilient Packet Ring Behavior", we evaluated the performance of the RPR ring during oscillations [95]. This paper was published in the proceedings of the 4th International Conference on Networking (ICN'05).

In our performance analysis, we focused on the analysis of throughput and fairness performance for two main types of scenarios in a network with nodes using the *aggressive* fairness mode.

The first scenario type, is the traditional "parallel parking-lot" scenario which was illustrated for an Ethernet network in Fig. 2.5 on page 23. In this scenario, all nodes on the network are so-called greedy senders (i.e. they send as much traffic as possible).

The second scenario type uses the same network topology, but in this scenario, the nodes send at a limited rate, inducing a behavior on the fairness algorithm not previously shown (to the best of our knowledge).

By observing the throughput and fairness behavior on a short and a long time-scale, we find that that, in most cases, RPR allows for full link-utilization and fair bandwidth distribution of the congested link.

In paper IV (presented in Section 3.3.3), we proposed a modification to the RPR fairness algorithm. In this paper (paper V), we applied this improvement to our network scenarios and compared the performance of the modified fairness algorithm to that of the original.

For all evaluated scenarios, we found that the modified algorithm performs at least as well as the original. For some problem scenarios, we found that the modified algorithm performs significantly better than the original.

3.3.5 Paper VI

In our paper, "Resilient Packet Ring Low Priority Traffic Latency", we evaluated the performance of the RPR ring during oscillations [96]. The paper was published in the proceedings of the 2005 International Conference on Communications in Computing (CIC 2005).

The work performed in the context of paper V focused on fairness and throughput properties for two main types of scenarios in a network with nodes using the *aggressive* fairness mode.

By the the results obtained while working with paper V, we were inspired to investigate the delay and jitter properties for these scenarios as well. Thus, in paper VI, we performed in essence the same experiments, but this time the performance analysis focused on the delay and jitter induced on traffic sent through the network.

What we found was that for scenarios where the fair rate estimation does not converge, or takes a long time doing so, the transit delay imposed on traffic passing through the bottleneck varies more. Another side-effect is that the access delay for traffic to be inserted onto the ring will also vary more for nodes upstream of the head. This variation is mainly caused by the oscillation in the fair rate estimate, leading to variations in the time it takes the nodes to accumulate[7] the amount of credits required for adding traffic onto the ring.

By applying the modification proposed in paper IV (described in Section 3.3.3), we found for all evaluated scenarios that the delay/jitter performance of the modified algorithm is at least as good as that of the original. For some scenarios where the original algorithm did not converge, we found that the modified algorithm performs significantly better than the original.

3.3.6 Discussion (Papers IV, V and VI)

Robichaud and Huang have presented results in the paper "Improved fairness algorithm to prevent tail node induced oscillations in RPR" where an improvement is proposed, which is claimed to solve the same problem [94]. However, in the scenarios used in supporting their theory of "tail-induced" oscillations, the throughput oscillations are caused by the fair rate estimates from a node operating as head of a congestion domain, not as tail of a congestion domain.

In fact, the problems which are described and the associated cited related work refer to the commonly known scenario of unbalanced traffic, first published by Knightly et al. [90]. For such scenarios, when using the *aggressive* fairness mode, during congested periods (i.e. when the STQ occupancy exceeds the low threshold), the use of the head's own add-rate as the fair rate estimate leads to the emptying of the STQ and resulting underutilization of the head's (previously congested) output link. This is the same problem that we address in paper III, discussed in Section 3.3.1.

[7] The operation of the RPR rate control block is discussed in Section 2.7.2 on page 42.

In the solution by Robichaud and Huang, rather than eliminating the problem, the symptoms are simply reduced. E.g. for a 2 node scenario, where the head transmits at 1% of the line rate and its upstream neighbor sends transit traffic at 99% of the line rate, the oscillations are reduced from a magnitude of 99% of the line-rate, to $\approx$ 18% of the line rate.

We are not aware of other work, addressing the problem of "tail-induced" oscillations.

3.4 An RPR Application

Differentiated Services (*DiffServ*) [97] was introduced as a "simple" and scalable framework for providing IP-based service class differentiation in an IP-network and is today considered as the most promising framework for providing IP-based service differentiation on a (Inter)network level [98].

In a DiffServ enabled network, IP packets are classified and marked at the network's ingress node(s). Based on some classification rule at the ingress node, the packet is assigned a DiffServ Code Point value (DSCP), carried within the packet's DS field [54]. This value maps the packet onto the network's available Per Hop Behaviors (PHB[8]), resulting in a specific packet forwarding at each DiffServ compliant node traversed by the packet.

Below, we provide a short description of three DiffServ PHBs of particular importance for the paper described in this section.

DiffServ's Expedited Forwarding (EF) [99] PHB is specified with the intent of providing a DiffServ building block that can be used for the provisioning of services that provides low loss, low delay and low jitter.

The DiffServ Assured Forwarding (AF) PHB is a specification of PHB group, aimed to provide guaranteed throughput, with support for different drop probabilities within an AF class as well as no reordering of packets within an AF micro-flow.

Finally, DiffServ's default PHB [54] is as a best-effort (BE) forwarding behavior. For the reminder of this thesis, we refer to traffic belonging to the default PHB, as BE traffic.

3.4.1 Paper VII

In our paper [100], "Applying the DiffServ Model to a Resilient Packet Ring Network", we evaluated the RPR technology when applied in a DiffServ context. By this we tested RPR's capabilities of providing traffic differentiation

[8] A single PHB is a special case of a PHB group.

as well as delay/jitter and throughput guarantees. A short version of this paper was accepted at the conference Networking 2005 [101].

In the paper, we evaluated RPR's abilities to meet the DiffServ Per Hop Behavior (PHB) group compliance requirements for a set of standardized DiffServ PHB groups, namely the DiffServ EF, AF and default (BE) PHBs.

Further, we introduced a simple analytical model based on invariants, expressing the relative service differentiation of the RPR service classes. Thus, given an RPR network, with a particular set of configured rate constraints, it should be possible to make some predictions about the network's throughput-behavior for a particular offered load. We also introduced an analytical expression for worst case access delay of high-priority traffic.

By use of the analytical model and the DiffServ PHB compliance requirements, a quantitative as well as a qualitative evaluation was done when comparing the achieved results to the DiffServ PHB compliance requirements.

3.4.2 Discussion (Paper VII)

The timeliness of this work is good, as there is ongoing work within the IETF addressing aspects of these problems. Currently, there is a preliminary Internet draft available (first version published July 10, 2005) from the IETF IPORPR Working Group[9]. The main focus of the draft however, is the detailed mapping proposal between various DiffServ PHBs (as well as mapping for MPLS traffic) and RPR service classes. The Internet draft has no discussions on the fulfillment of DiffServ PHB conformance requirements, neither does it address the implementation of various drop priorities for the AF PHB group.

Apart from the ongoing work in the IETF IPORPR working group, we have only found one other paper addressing the problem of using RPR in a DiffServ context. One of the problems with the work by Wang et al. [102] is that a proposal is made to map the DiffServ AF PHB to RPR's A1 service class and to map the DiffServ EF PHB to RPR's A0 service class. By this, the use of the 1TB node architecture is effectively prohibited. Further, for a node using the 2TB-architecture, the design of the RPR MAC is done so that the MAC client never knows whether a high-priority packet is sent as an A0 or an A1 packet. Thus with the mapping proposed, one will never know, whether an EF or AF packet will be transported using RPR service classes A0 and A1. In effect this makes it impossible to provide a clear service differentiation between DiffServ traffic of the EF and AF PHBs, traversing an RPR network.

[9] http://www.ietf.org/html.charters/iporpr-charter.html

	DiffServ PHB Requirements				**RPR Service Class Guarantees**	
DiffServ PHB	**Jitter**	**Bandwidth**	**Is Mapped onto**	**RPR Service Class**	**Jitter**	**Bandwidth**
EF	low	guaranteed	⇒	A	low	guaranteed
AF	n.a.	guaranteed	⇒	B	bounded	guaranteed
BE (default)	n.a.	n.a.	⇒	C	no guarantees	

Table 3.1: Proposed mapping between DiffServ PHBs and RPR service classes.

In our mapping proposal, we used the mapping shown in Table 3.1. This ensures that traffic of the DiffServ EF PHB has exclusive access to the RPR high-priority traffic class. Whether an individual data packet is transported as an A0 or A1 packet, makes no difference to the packet once accepted by the MAC layer for transmission onto the ring. For traffic belonging to the DiffServ AF PHB, it is left to the MAC client to provide differentiated service (i.e. drop priorities) for the various AF PHB classes. The actual transport of traffic is done using RPR service class B. Finally, traffic belonging to the DiffServ default (best effort) PHB is transported using RPR service class C.

3.5 Summary of Contributions

The method of presenting our results for buffer-insertion rings in this thesis, have followed what we described as a "top-down, bottom-up" approach. That is, in paper I, we started by presenting an RPR "tutorial", describing in essence all the features provided by the RPR technology. Then in later papers, we addressed individual sub-areas of RPR, with a strong focus on fairness issues. Finally, at the end, we presented a user-application of RPR, which is made possible by the workings of the fairness and service differentiation capabilities provided by the RPR standard.

Starting with paper I, we presented an RPR tutorial, suitable as a starting-point of study for the RPR novice, be it an network engineer or a university professor looking for an introduction to RPR technology.

From this starting point, in paper II, we continued by presenting an in-depth study of the behavior of the RPR *aggressive* fairness mode. By use of a control-theoretic approach, accompanied by simulation results, we investigated the operational limits of this rate calculation mode. The main result,

is an analytically rooted expression that can be used for determining some configuration parameters that may be used when deploying an RPR network.

Next, in paper III, we addressed some serious performance deficiencies of the *aggressive* fairness mode in the context of so called "unbalanced" traffic scenarios [90]. This problem is considered important by the RPR research-community and has been addressed by several members in this community. However, the approaches showing the best results in addressing the problem were incompatible with the framework provided by the RPR standard and are also relatively complex in their algorithmic solutions. Hence it was a problem that justified further research.

The finding of suitable solutions for the unbalanced traffic scenario, proved to be a harder task than first anticipated. In the end, we arrived at a solution, where by use of a fair rate estimate, adjusted according to the occupancy and growth direction of the STQ buffer and constrained within a dynamic rate range, the end-result appeared very promising. We chose to name the rate calculation mode *"moderate"*, that is, something that lies in-between the *aggressive* and the *conservative* fairness modes in its eagerness to adjust the fair rate estimate.

For unbalanced traffic scenarios, our *moderate* fairness mode outperforms the *aggressive* fairness mode in terms of e.g. link-utilization and convergence-time (the *aggressive* fairness mode does not converge). Further, our *moderate* fairness mode outperforms (see footnote 3 on page 56) the *conservative* mode in terms of convergence-time and link-utilization.

The complexity of our algorithmic solution is the same as that of the *aggressive* and *conservative* fairness modes (i.e. $O(1)$). Further, the solution can be used within the framework provided by the RPR standard.

Continuing with paper IV, we addressed a little known deficiency of the RPR fairness algorithm that affects the performance of both the *aggressive* and *conservative* fairness modes.

This deficiency originates from the behavior of the tail. By concealing[10] from its upstream neighbors, rate constraints in effect over a downstream congestion domain head, this has the side effect of allowing upstream nodes to transmit traffic at excessive rates (i.e. more than their fair share).

Some symptoms of this excessive transmission is the failure of the fairness mode to converge to the fair division of sending rates (*aggressive* mode) as well as increased delay/jitter of traffic in the congestion domain (both modes).

By the introduction of a *passedTail*[11] bit, such rate constraints are no

[10] That is, by removing fairness messages received from downstream nodes from the ring.
[11] Using one of the currently unused and reserved bits of the RPR fairness messages.

longer removed by the tail. In our proposed solution, the tail marks (by setting the passedTail bit to one) the fairness message before sending it further upstream. The fairness message is removed from the ring once it is received by the head of an upstream congestion domain. If there is no such upstream congestion domain, the fairness message is removed from the ring by the originating node, once having circulated the ring once.

By this modification of the RPR rate distribution functionality, nodes upstream of a congestion domain tail are prevented from transmitting traffic at excessive rates through the downstream head. This leads to significant improvements in the performance of the *aggressive* fairness mode, as well as some improvements in the performance of the *conservative* fairness mode.

In papers V and VI, we investigated further the behavior of the *aggressive* fairness mode with our proposed improvement. Our findings indicate that for several of the investigated scenarios, our proposed improvement leads to significant performance improvements in terms of throughput/fairness and delay/jitter. In none of the evaluated scenarios did our proposed improvement lead to a reduced performance.

Finally, in paper VII, we evaluated the performance of an RPR network within the context of the DiffServ framework. In this context, the service differentiation properties of RPR is used to provide transport services for a DiffServ enabled network based on RPR. In the paper, we proposed a simple analytical framework based on invariants that can be used to predict the service differentiation in an RPR network for a given set of RPR service class configuration settings. An analytical expression for the prediction of worst case access delays for high priority traffic was also provided.

Further, we proposed a simple mapping between the DiffServ PHBs EF, AF and Default and the RPR service classes A, B and C. By use of this mapping, we performed a set of performance evaluation experiments. Our results indicate, that by use of this mapping, the conformance requirements of the three DiffServ PHBs are partially fulfilled. However, the support of individual drop priorities for AF PHB classes has to be provided outside the RPR MAC layer (e.g. in the MAC client). This should come as no surprise, as the RPR MAC is lossless.

3.6 Concluding Remarks

In our contributions, although we have a strong focus on RPR, most of the problems and solutions outlined for RPR fairness and rate control applies to buffer-insertion rings in general. In a generalized view of the problem, the challenge is to obtain a rate control mechanism that ensures fairness, while

at the same time seeking to maximize throughput and minimize delay. As discussed in Chapter 2.1, there are two types of congestion control methods for achieving this, namely "congestion avoidance" and "congestion recovery". With congestion avoidance, the goal is to avoid getting into the congestion operational part of the network in the first place. With congestion recovery, the goal is to recover from a congested situation once this occurs [29, 32].

RPR uses the "congestion recovery" type of congestion control methods for the handling of congestion on the ring. At a congested node, depending on the node design and fairness mode in use, congestion recovery is handled differently. In the node design utilizing two transit buffers (2TB design), a threshold-based method is used, which is based mainly on the occupancy of the secondary transit buffer, storing medium and low priority transit traffic. In the node design utilizing one transit buffer (1TB design) a similar threshold-based method is used. In the 1TB design however, the thresholds are rate and delay-based.

In our analytical and simulation research work, we have utilized the 2TB node design for our experiments. However, as the 1TB node design utilizes a similar threshold-based congestion recovery approach, we believe that our findings may apply to a network utilizing the 1TB node design as well.

3.7 Summary

In this chapter, we presented our contributions to the research area of buffer-insertion rings. We also discussed the contributions of 7 individual papers, broken down over four distinct areas, namely: "An introduction to the RPR standard and its problem area"; "Analytical Model of Aggressive Fairness Mode"; "Rate Calculation and Distribution"; and "An RPR Application".

Our findings have been accepted or published in various publication channels: one in IEEE Communications Magazine, two in major conferences organized by the IEEE Communications society (GLOBECOM'05 and ICC'05), and three in other conferences (ICN'05, CIC'05 and Networking 2005).

Three of the papers presented in Part II, are identical to the published versions of the papers. The remaining four are extended versions of the corresponding conference versions in the form of technical reports, containing a more comprehensive analysis, performance evaluation and discussion.

In the next chapter, we present our final conclusions along with a set of opportunities for future work.

Chapter 4

Conclusions

In the previous chapter, we presented our contributions, in terms of our publications to the research area of buffer-insertion rings.

In this chapter, we present the final conclusions on our contributions, with a particular emphasis on the fulfillment of the Research Objectives, defined in Section 1.6. Following this, in Section 4.3, we provide some recommendations on possible directions for future work. Finally, in Part II of this thesis, the individual research papers are presented.

4.1 General Conclusions

The recently standardized IEEE standard for Resilient Packet Rings, IEEE 802.17-2004 [22], is currently the last branch on the tree of buffer-insertion ring standards, with the root being the work presented by Hafner, Nenadal and Tschanz [63]. The RPR standard provides several features not provided by that of Hafner et al. Among those are differentiated classes of service, automatic restoration of connectivity in case of node or link failure (i.e. protection) and fairness.

Among the goals of this thesis, was the investigation of the problem of fairness in buffer-insertion rings, with a particular emphasis on the IEEE Resilient Packet Ring. Thus the first challenge encountered, was the buildup of knowledge on problems related to the provisioning of fairness in buffer-insertion rings as well as understanding the, at the time, recently started Resilient Packet Ring standardization effort. Concurrently, a discrete event simulator had to be developed, so that we had a tool to provide an in-depth understanding of the basic concepts and ideas related to buffer-insertion rings and RPR.

Given this starting point, we proceeded[1] to cover the four areas that were presented initially in Chapter 3 on page 51. In presenting our research contributions in this thesis, we started by discussing our RPR tutorial paper, covering most aspects of the RPR technology. Then we proceeded via an in-depth analysis of the RPR *aggressive* fairness mode, to the resolution of a set of problems related to the calculation and distribution of fair rate estimates in an RPR congestion domain. Finally, we concluded with the investigation of an application utilizing the RPR technology, namely RPR used in a DiffServ context.

In terms of contributions to the area of flow control and fairness in buffer-insertion rings, our most important contributions are as follows:

i) We have shown that congestion is adequately recovered from, when controlled by the node experiencing the congestion. By this, we have shown that the rate estimates calculated by the congested node, in attempting to recover from congestion, should be distributed to all[2] upstream nodes, regardless of whether they contribute to the congestion or not.

ii) We have shown that the accuracy of fair rate estimates can be improved significantly, by maintaining the fair rate estimate separately, based on observing the local congestion situation of a node, rather than calculating it directly based on the sending behavior of the congested node.

In terms of contributions to the area directly related to Resilient Packet Rings, our most important contributions are as follows:

iii) We have, by contributions to the RPR standardization group in the form of discussions, submitted draft comments and simulation experiments, contributed to the improvement of the RPR standard.

iv) We have provided an RPR tutorial to the network community, which constitutes a low-threshold entry point to the study and understanding of the Resilient Packet Ring standard.

v) We have derived an analytically rooted expression, that may be used to correctly configure a node on an RPR ring, utilizing the *aggressive* fairness mode.

vi) We have shown, by use of analytical and simulation methods, that the use of RPR to provide DiffServ EF (Expedited Forwarding), AF (Assured Forwarding) and Default (Best Effort) appears feasible.

[1] Note that the order in which the various issues are presented in this thesis, does not correspond directly to the order in which the research was executed.

[2] That is, all nodes upstream of the congestion point up-to the next congestion point on the ring. If there are no upstream congestion point, the rate estimate circulates the ring once before being removed.

4.2 Fulfilment of Research Objectives

In Section 1.6, we defined a set of Research Objectives (ROs) to be used at the conclusion of this thesis, to determine how well the thesis meets the established objectives. Six research objectives were defined, and we address them sequentially in the continuation of this Section.

4.2.1 Research Objective 1

The first research objective (RO1) was "To investigate the problem area of flow control, congestion control and fairness in ring-based networks and to propose and discuss solutions for buffer-insertion rings, using Resilient Packet Rings as a test-environment for our proposed solutions."

In investigating these problem areas, we traced efforts leading back to the Newhall Ring [3] in the late 60's, utilizing a token-based access scheme. By restricting the amount of traffic the individual nodes can transmit while holding the token, the token-based access methods are inherently fair. However, the mutual exclusive token-based sending behavior may lead to a lower link-utilization as compared to slotted and buffer-insertion rings. The access delays of token-rings are typically also higher than in slotted and buffer-insertion rings, because once a node has released the token, it has to wait for the token to propagate node by node on the ring before returning to it.

Two other classes of ring networks, namely the slotted and the buffer-insertion rings [62, 63], resolve the mutually exclusive property of the token-based rings by dividing the bandwidth capacity of the ring into sub-units. In the slotted ring, these sub-units are of a fixed size, and circulate the ring in the form of slots, with a corresponding slot-header. In the buffer-insertion ring, the introduction of an insertion buffer allows for sub-units of variable size, limited upwards to a defined maximum. Further, the individual transmission "slots" are marked by the absence of traffic (bits) rather than by a slot header. Thus, a node with data to send, that is not currently transmitting, receiving or has any data queued from upstream nodes, can transmit its own traffic onto the ring. Finally, the properties of the buffer-insertion principle allows it to be combined with a physical medium using slotted access. Thus, in some sense, it can be seen as a generalized version of the slotted ring.

We have discussed several MAC protocols for buffer-insertion rings: MetaRing [68–70], CRMA-II [64, 65], DVSR [72] and CQMA [71].

Various aspects, with a strong focus on fairness, of the resent IEEE MAC protocol for buffer-insertion rings – the 802.17-2004 Resilient Packet Ring standard [22] have been studied and our findings are reported in the 7 papers

in Part II of this thesis.

Although we may lack results for fairness for Resilient Packet Ring utilizing nodes with the 1TB-architecture (1 transit buffer), we argue that the analysis of and solutions developed for the 2TB-architecture are relevant and applicable for the 1TB-architecture as well (see Sections 3.3.1 and 3.6 for a further discussion).

In summary, we consider that we, by this, meet Research Objective 1.

4.2.2 Research Objective 2

The second Research Objective, RO2, was: "To contribute, by publication of results, to the dissemination of knowledge on the Resilient Packet Ring architecture, both within the network oriented research community as well as in the general networking community."

Most[3] of the results in the 7 papers in part II of this thesis have been published in international magazines and conferences with referee. One of the contributions was presented in the highly visible IEEE Communication Magazine, the number three most-cited journal in telecommunications and the number fifteen cited journal in electrical and electronics engineering in 2003. Two contributions were presented/accepted at IEEE Communications Society flagship[4] conferences (ICC'05 and GLOBECOM'05), while three contributions were presented at other conferences (Networking 2005, ICN'05 and CIC'05).

Based on this, we consider RO2 to be fulfilled in full.

4.2.3 Research Objective 3

The third Research Objective, RO3, was: "To contribute to the development of the IEEE 802.17 standard and to aid the 802.17 working group in the development of a tested, working, error-free distributed fairness algorithm."

Several contributions was made by the author, contributing towards the fulfilment of this Research Objective. First, the author has submitted (according to a rigorous formal process) both editorial and technical comments to various draft versions of the RPR standard. The summary of comments is shown in Table 4.1. The reason why most of the editorial (i.e. comments relating to the document format – text, graphics, layout, etc.) were rejected, was that changes to the document had made them obsolete. As seen, 94% of

[3] While the main results from all the papers have been accepted/published in publication channels with referee, strict page limitations of the various conferences forced us to omit parts of our findings from these papers.

[4] According to IEEE's web-page describing the conferences.

Editorial Comments			Technical Comments		
Submitted	Accepted	Rejected	Submitted	Accepted	Rejected
20	8	12	31	29	2

Table 4.1: Number of comments submitted to the IEEE 802.17 standardization process.

the technical comments (i.e. comments on technical aspects of the document – formulas, state machines, definitions, etc.) were accepted. In addition to these comments, we tested out various ideas and concepts for the RPR working group's fairness ad-hoc sub-group.

The work carried out during a standardization processes is substantial in terms of man-hours. Both for the individual contributors and for the project as a whole. Although the author could certainly have worked full time with the RPR standardization project for the full-period of the PhD project, his time had to be divided in a reasonable manner between **all** the defined Research Objectives.

Still, the hours spent in acquiring the knowledge required to be able to provide intelligible comments has been significant in the context of the time available for the author's PhD project. As such, by the contribution (and acceptance) of a number of comments to the RPR standardization work as well as the work performed for the RPR working group's fairness ad-hoc sub-group, we consider RO3 to be fulfilled in full.

4.2.4 Research Objective 4

The fourth Research Objective, RO4, was: "To critically evaluate the algorithmic solutions chosen by the Resilient Packet Ring working group for the RPR fairness algorithm and to propose improvements or alternatives where appropriate."

This Research Objective was addressed by several of our papers as well as in direct communication with other member of the RPR working group. In e.g. papers IV, V and VI, we address one specific weakness in the methods used to constrain the propagation of fairness within a congestion domain. Further, we investigate its side-effects and propose an improvement.

In paper III, we address a weakness in the RPR *aggressive* fairness mode rate calculation method and propose an improvement, namely, the *moderate* fairness mode.

In conclusion, we consider RO4 to be fulfilled in full.

4.2.5 Research Objective 5

The fifth Research Objective, RO5, was: "To evaluate applications of the RPR technology, to help ensure that there is match between a particular application's requirements and the functionality and associated guarantees provided by an RPR network."

The only effort in meeting this Research Objective is in the context of paper VII. In paper VII, we evaluate the use of the RPR network as a carrier of DiffServ traffic. Given that RPR is a technology aimed at providing service differentiation for different traffic classes as well as fairness for low-priority traffic, DiffServ is perhaps the most important application of an RPR network, when used for the transport of IP-traffic. In terms of paper VII, we evaluate the ability of RPR to meet the conformance requirements of the DiffServ PHBs EF, AF and Default. The results obtained however, also apply to an RPR network, used as a pure layer-two network, providing service differentiation for various types of layer-two traffic. Another valid application of the results, is within an RPR network, using the high-priority traffic class as a bearer for circuit-switched traffic, while using the remaining service classes for IP-traffic.

Thus given that service differentiation is a very important functionality targeted by the RPR technology, we argue that the evaluation of RPR used in the context of the framework considered most important for providing service differentiation in an IP network (i.e. DiffServ) [98] meets RO5 sufficiently.

4.2.6 Research Objective 6

The sixth and final Research Objective, RO6, was: "To develop a national knowledge base in the problem area of buffer-insertion and Resilient Packet Rings."

It is difficult to assess whether this goal has been achieved or not. Of course, some knowledge has been gathered as well as disseminated by and to the various people the author has been working with during his PhD project. In addition to this, the author has presented various aspects of the RPR technology to members of Norwegian industry and academia.

Further, in addition to this PhD thesis, parts of this knowledge is presented in two finished master theses covering topology discovery [18] and bridging [19] as well as one ongoing master thesis covering multi-choke fairness.

As in any research project, in addition to the results published, most of the knowledge obtained follows the people having worked on the actual problems. Thus, the knowledge base exists in the form that the people, having worked

on the problems, are located in various branches of Norwegian research and industry. As far as we know however, there is no work ongoing in Norwegian research or industry, resulting from this research project.

Thus, in conclusion, we say that RO6 has been partially fulfilled.

4.3 Further Work

As we conclude the presentation of this thesis work, some comments about unaddressed issues as well as issues that are addressed, but that could be improved by an additional effort, are appropriate.

We have divided these issues into respectively: immediate research, discussed in Section 4.3.1; short-term research, discussed in Section 4.3.2; and long-term research, discussed in Section 4.3.3.

Immediate research is work that can be considered extensions of results published within the context of this thesis, and that are feasible to be undertaken within a short-term (i.e. months).

Short-term research, is work requiring further effort to investigate, and that could be considered a candidate topic in the context of a Masters thesis.

Finally, long-term research, is work that would require significantly more work, and that could be a candidate topic for e.g. a PhD thesis.

4.3.1 Immediate Research

Based on the results of our work in the context of this thesis, some issues that qualify as candidates for immediate investigation are:

i) Automatic adjustment of per-node configuration parameters as a result of topology changes.

In paper II, we derived an analytic expression showing the relationship between a configuration parameter termed *lpCoef* and the size of a congestion domain that could be supported. This relationship is applicable for the *aggressive* and *moderate* fairness modes only. This relationship allows for the automatic configuration of a running network using the *aggressive* or *moderate* fairness modes. This can be done on a ring global basis, or on a per congestion domain basis.

ii) An evaluation of our proposed *moderate* fairness mode using the 1TB (Transit Buffer) node-design.

As discussed in Section 3.3.1, our *moderate* fairness mode has focused on the 2TB node architecture. We do however believe that the principles used in the proposed rate calculation mode are applicable to the 1TB node architecture as well.

4.3.2 Short-Term Research

In [19], Kvalbein evaluated two different methods for bridging of RPR rings. His work focused on the evaluation of methods for the forwarding of traffic between nodes on different rings. An issue not provided by his work however, is the provisioning of fairness on a global network scale.

When interconnecting RPR rings, there is no back-pressure between interfaces connected to different rings. This means that just as when interconnecting Ethernet switches not utilizing Ethernet back-pressure (discussed in Section 2.4.1), packets may be lost. This happens when the aggregate of input traffic destined for one output-port is larger than the capacity of the output-port.

Another apparent problem is that if there is a bottleneck between the bridge and the destination of traffic which it bridges, there is no means of providing fairness between contending traffic sources located behind the bridge (i.e. at another ring). An example of such a scenario, illustrating this behavior is shown in Fig. 4.1.

By the definition of some global fairness reference model, as well as an algorithm enforcing or approximating the defined model, it should be possible to avoid packet loss at the bridge, as well as achieving some type of global fairness.

4.3.3 Long-Term Research

The IEEE 802.17-2004 standard requires equal bandwidth links on the ring. One issue suitable for investigation, is the use of non-equal bandwidth links. By this, individual link segments could be upgraded as the long-term demand for bandwidth over individual links gets close to the capacity of the link.

Furthermore, the links between the individual nodes must be bidirectional. Another approach, could be to allow for a number of links (limited downwards to 1) in each direction.

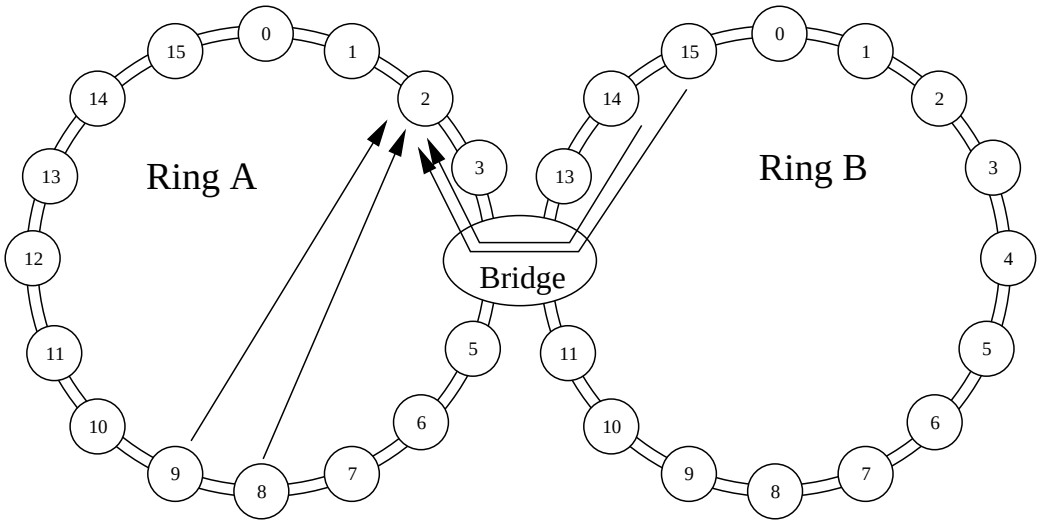


Figure 4.1: A network scenario containing two rings, A and B. Node 8 and 9 (A8 and A9) on ring A send traffic to node 2 (A2) on ring A. Additionally, nodes 14 and 15 (B14 and B15) on ring B send traffic to the same node. If the sum of traffic from nodes B14 and B15 is larger than the fair share of the bridge on ring A, the bridge buffers will overflow (resulting from rate restrictions in effect at the ingress point of ring A). The bridge has no means of conveying rate restrictions in effect on ring A to nodes on ring B for which it bridges traffic. Further, the bridge has no means of ensuring a fair division of bandwidth between the flows of B14 and B15.

Bibliography

[1] I. T. Union, "Standard X.200. Information Technology - Open Systems Interconnection - Basic Reference Model: The basic model," July 1994.

[2] H. R. van As, "Media access techniques: The evolution towards terabit/s LANs and MANs," *Computer Networks and ISDN Systems*, vol. 26, no. 6-8, pp. 603–656, March 1994.

[3] W. D. Farmer and E. E. Newhall, "An experimental distributed switching system to handle bursty computer traffic," in *Proceedings of ACM Symposium on Problems in the Optimization of Data Communication*, October 1969, pp. 1–33.

[4] N. Abramson, "The ALOHA system – another alternative for computer communications," in *Proceedings of the 1970 fall joint computer conference*, November 17-19 1970, Houston, TX, USA.

[5] R. M. Metcalfe and D. R. Boggs, "Ethernet: distributed packet switching for local computer networks," *Communications of the ACM*, vol. 19, no. 7, pp. 395–404, 1976.

[6] "Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications," 1985, aNSI/IEEE Std 802.3-1985.

[7] F. Tobagi and V. Hunt, "Performance analysis of carrier sense multiple access with collision detection," in *Proceedings-of-the-Local-Area-Communications-Network-Symposium*, Boston, MA, USA, May 7-9 1979.

[8] "Fiber distributed data interface (FDDI) - token ring media access control (mac)," 1987.

[9] "Token ring access method and physical layer specifications," September 29 1989, IEEE Std 802.5-1989.

[10] W. Bux, "Token-ring local-area networks and their performance," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 238–256, February 1989.

[11] F. E. Ross, "An overview of FDDI: the Fiber Distributed Data Interface," *IEEE Journal on Selected Areas in Communications*, vol. 7, no. 7, pp. 1043–1051, September 1989.

[12] "Distributed queue dual bus (DQDB) subnetwork of a metropolitan area network (MAN)," July 3 1991, IEEE Std 802.6-1990.

[13] "Wireless LAN medium access control (MAC) and physical layer (PHY) specifications," November 18 1997, IEEE Std 802.11-1997.

[14] R. Seifert, *Gigabit Ethernet*. Addison Wesley, 1998.

[15] I. Keslassy, S.-T. Chuang, K. Yu, D. Miller, M. Horowitz, O. Solgaard, and N. McKeown, "Scaling internet routers using optics," in *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA: ACM Press, 2003, pp. 189–200.

[16] "IEEE Standard for Scalable Coherent Interface (SCI)," August 2 1993, IEEE Std 1596-1992.

[17] D. Gustavson and Q. Li, "The scalable coherent interface (SCI)," *IEEE Communications Magazine*, vol. 34, no. 8, pp. 52–63, August 1996.

[18] P. Teigen, "Nettverkskartlegging i "reslient packet ring"," Master's thesis, University of Oslo/Department of informatics, 2003. [Online]. Available: http://wo.uio.no/as/WebObjects/theses.woa/wa/these?WORKID=10475

[19] A. Kvalbein, "Bridging in RPR networks," Master's thesis, University of Oslo/Department of informatics, 2003. [Online]. Available: http://wo.uio.no/as/WebObjects/theses.woa/wa/these?WORKID=10558

[20] A. S. Tanenbaum, *Computer Networks*, 4th ed. Prentice Hall, 2003.

[21] IEEE Computer Society, "IEEE Std 802.1D-2004 media access control (MAC) bridges," June 9 2004.

[22] ——, "IEEE Std 802.17-2004 Resilient packet ring (RPR) access method and physical layer specifications," September 24 2004.

[23] R. Jain, *The Art of Computer Systems Performance Analysis.* John Wiley & Sons, Inc, 1991.

[24] A. Bragg and H. Perros, "Modelling and analysis of ultra high-capacity optical networks," in *Proceedings of Advanced Simulation Technologies Conference 2004 (ASTC 2004)*, April 18-22 2004.

[25] OPNET Technologies, Inc, "OPNET Modeler." [Online]. Available: http://www.opnet.com/

[26] H. Tyan, "Design, realization and evaluation of a component-based compositional software architecture for network simulation," Ph.D. dissertation, Ohio State University, 2002.

[27] S. McCanne and S. Floyd, "ns–network simulator," http://www.isi.edu/nsnam/ns/.

[28] R. Jain, "Congestion control in computer networks: issues and trends," *IEEE Network*, vol. 4, no. 3, pp. 24–30, May 1990.

[29] C.-Q. Yang and A. Reddy, "A taxonomy for congestion control algorithms in packet switching networks," *IEEE Network*, vol. 9, no. 4, pp. 34–45, July-August 1995.

[30] A. Giessler, J. D. Haenle, A. König, and E. Pade, "Free buffer allocation - an investigation by simulation," *Computer Networks*, vol. 2, no. 3, pp. 191–208, July 1978.

[31] L. Kleinrock, "Power and deterministic rules of thumb for probabilistic problems in computer communications," in *International Conference on Communications*, Boston, Massachusetts, June 1979, pp. 43.1.1–43.1.10.

[32] A. Mankin and K. Ramakrishnan, "Gateway congestion control survey," August 1991, IETF, RFC 1254.

[33] B. M. Leiner, V. G. Cerf, D. D. Clark, R. E. Kahn, L. Kleinrock, D. C. Lynch, J. Postel, L. G. Roberts, and S. S. Wolff, "The past and future history of the Internet," *Commun. ACM*, vol. 40, no. 2, pp. 102–108, 1997.

[34] T. Marill and L. Roberts, "Toward a cooperative network of time-shared computers," in *Proceedings of the Fall AFIPS Conference*, October 1966.

[35] F. Heart, R. Kahn, S. Ornsstein, W. Crowther, and D. Walden, "The interface message processor for the ARPA computer network," in *Proceedings of the AFIPS 1970 Spring Joint Computer Conference*, vol. 36, 1970, pp. 551–567.

[36] V. G. Cerf and R. E. Kahn, "A protocol for packet network intercommunication," *IEEE Transactions on Communications*, vol. 22, no. 5, pp. 637–648, May 1974.

[37] M. Gerla and L. Kleinrock, "Flow control: A comparative survey," *IEEE Transactions on Communications*, 1980.

[38] H. Zhang and S. Keshav, "Comparison of rate-based service disciplines," in *SIGCOMM '91: Proceedings of the conference on Communications architecture & protocols*. New York, NY, USA: ACM Press, 1991, pp. 113–121.

[39] IEEE Computer Society, "IEEE Std 802.3-2002 (Ethernet)," March 8 2002.

[40] W. Noureddine and F. Tobagi, "Selective back-pressure in switched Ethernet LANs," in *Global Telecommunications Conference, (GLOBECOM '99)*, December 1999.

[41] R. Seifert, *The Switch Book*. Wiley, 2000.

[42] D. Bertsekas and R. Gallager, *Data Networks*, 2nd ed. Prentice-Hall, 1992.

[43] J. M. Jaffe, "Bottleneck flow control," *IEEE Transactions on Communications*, vol. 29, no. 7, pp. 954–962, July 1981.

[44] A. Bar-Noy, A. Mayer, B. Schieber, and M. Sudan, "Guaranteeing fair service to persistent dependent tasks," *SIAM Journal on Computing*, vol. 27, no. 4, pp. 1169–1189, 1998.

[45] A. Parekh and R. Gallager, "A generalized processor sharing approach to flow control in integrated services networks-the single node case," in *INFOCOM '92. Eleventh Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 2, May 4-8 1992, pp. 915–924.

[46] ——, "A generalized processor sharing approach to flow control in integrated services networks: the single node case," *IEEE Transactions on Communications*, vol. 1, no. 3, pp. 344–357, June 1993, this article was presented in part at INFOCOM '93.

[47] L. Kleinrock, *Queueing Systems.* Wiley, January 1976, vol. 2, Computer Applications.

[48] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm," in *SIGCOMM '89: Symposium proceedings on Communications architectures & protocols.* New York, NY, USA: ACM Press, 1989, pp. 1–12.

[49] ——, "Analysis and simulation of a fair queueing algorithm," *Internetworking - Research and Experience*, vol. 1, 1990.

[50] J. Nagle, "Congestion control in IP/TCP Internetworks," January 6 1984, IETF, RFC 896.

[51] J. B. Nagle, "On packet switches with infinite storage," *IEEE Transactions on Communications*, vol. 35, no. 4, pp. 435–438, April 1987.

[52] A. K. Parekh and R. G. Gallagher, "A generalized processor sharing approach to flow control in integrated services networks: the multiple node case," *IEEE/ACM Transactions on Networking*, vol. 2, no. 2, pp. 137–150, 1994.

[53] J. S. Turner, "New directions in communications (or which way to the information age?)," *IEEE Communications Magazine*, vol. 24, no. 10, pp. 8–15, October 1986.

[54] K. Nichols, S. Blake, F. Baker, and D. Black, "Definition of the Differentiated Services field (DS field) in the IPv4 and IPv6 headers," December 1998, IETF, RFC 2474.

[55] F. Kelly, "Charging and rate control for elastic traffic," *European Transactions on Telecommunications*, vol. 8, no. 1, pp. 33–37, January 1997.

[56] F. Kelly, A. Maulloo, and D. Tan, "Rate control for communication networks: shadow prices, proportional fairness and stability," *Journal of the Operational Research Society*, vol. 49, no. 3, pp. 237–252, March 1998.

[57] L. Burgstahler, K. Dolzer, C. Hauser, J. Jahnert, S. Junghans, C. Macian, and W. Payer, "Beyond technology: the missing pieces for QoS success," in *Proceedings of the ACM SIGCOMM workshop on Revisiting IP QoS.* ACM Press, 2003, pp. 121–130.

[58] M. Ofek, Y.and Yung, "Metanet: principles of an arbitrary topology LAN," *IEEE/ACM Transactions on Networking*, vol. 3, no. 2, pp. 169–180, April 1995.

[59] N. Maxemchuk, "Routing in the Manhattan street network," *IEEE Transactions on Communications*, vol. 35, no. 5, pp. 503–512, May 1987.

[60] A. Mayer, Y. Ofek, and M. Yung, "Local and congestion-driven fairness algorithm in arbitrary topology networks," *IEEE/ACM Transactions on Networking*, vol. 8, no. 3, pp. 362–372, 2000.

[61] P. Black, "Dictionary of algorithms and data structures."

[62] J. R. Pierce, "Network for block switching of data," *Bell Systems Technical Journal*, vol. 51, no. 6, pp. 1133–1145, 1972.

[63] E. Hafner, Z. Nendal, and M. Tschanz, "A digital loop communication system," *IEEE Transactions on Communications*, vol. 22, no. 6, pp. 877–881, June 1974.

[64] H. R. van As, W. W. Lemppenau, P. Zafiropulo, and E. Zurfluh, "CRMA-II: A Gbit/s MAC protocol for ring and bus networks with immediate access capability," in *Proceedings of the Ninth Annual European Fibre Optic and Local Area Networks Conference (EFOC/LAN'91)*, London, June 1991, pp. 262–277.

[65] H. van As, W. Lemppenau, H. Schindler, and P. Zafiropulo, "CRMA-II: A MAC protocol for ring-based Gb/s LANs and MANs," *Computer Networks and ISDN Systems*, vol. 26, no. 6-8, pp. 831–840, March 1994.

[66] E. Zurfluh, "A Gbit/s LAN/MAN prototype implementation of the CRMA protocol," in *4th IEEE MAN Workshop*, Fort Meyers, FL, November 1990.

[67] E. Zurfluh, R. Cideciyan, P. Dill, R. Heller, W. Lemppenau, P. Mueller, H. Schindler, and P. Zafiropulo, "The IBM Zurich research laboratory's 1.13 Gb/s LAN/MAN prototype," *Computer Networks and ISDN Systems*, vol. 26, no. 2, pp. 163–184, 1993.

[68] I. Cidon, L. Georgiadis, R. Guerin, and Y. Shavitt, "Improved fairness algorithms for rings with spatial reuse," in *Proceedings of the 13th Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 3, June 12-16 1994, pp. 1103–1111.

[69] ——, "Improved fairness algorithms for rings with spatial reuse," *IEEE/ACM Transactions on Networking*, vol. 5, no. 2, pp. 190–204, 1997.

[70] I. Cidon and Y. Ofek, "MetaRing - a full duplex ring with fairness and spatial reuse," *IEEE Transactions on Communications*, vol. 41, no. 1, pp. 110–120, January 1993.

[71] J. Schuringa, G. Remšak, and H. R. van As, "Cyclic Queuing Multiple Access (CQMA) for RPR networks," in *Proceedings of the 7th European Conference on Networks & Optical Communications (NOC2002)*, Darmstadt, Germany, June 2002, pp. 285–292.

[72] V. Gambiroza, P. Yuan, L. Balzano, Y. Liu, S. Sheafor, and E. Knightly, "Design, analysis, and implementation of DVSR: a fair high-performance protocol for packet rings," *IEEE/ACM Transactions on Networking*, vol. 12, no. 1, pp. 85–102, 2004.

[73] D. Tsiang and G. Suwala, "The Cisco SRP MAC layer protocol," August 2000, IETF, RFC 2892.

[74] D. Wang, K. Ramakrishnan, C. Kalmanek, R. Doverspike, and A. Smiljanic, "Congestion control in Resilient Packet Rings," in *Proceedings of the 12th IEEE International Conference on Network Protocols, 2004. ICNP 2004*, October 5-8 2004, pp. 108–117.

[75] F. Davik, M. Yilmaz, S. Gjessing, and N. Uzun, "IEEE 802.17 Resilient Packet Ring tutorial," *IEEE Communications Magazine*, vol. 42, no. 3, pp. 112–118, March 2004.

[76] F. Davik and S. Gjessing, "The stability of the Resilient Packet Ring Aggressive fairness algorithm," in *Proceedings of the 13th IEEE Workshop on Local and Metropolitan Area Networks (LANMAN2004)*, Mill Valley, CA, April 25-28 2004, pp. 17–22.

[77] S. Gjessing and F. Davik, "Avoiding head-of-line blocking using an enhanced fairness algorithm," in *Proceedings of the International Conference on Telecommunication (ICT 2002)*, Beijing, China, June 21-26 2002, pp. Paper C089, 6 pages.

[78] ——, "Improved fairness and class of service behaviour in a Resilient Packet Ring," in *Proceedings of 12th IEEE Workshop on Local and Metropolitan Area Networks (LANMAN 2002)*, Stockholm, Sweden, August 11-13 2002.

[79] ——, "Performance evaluation of back-pressure fairness in RPR," in *Proceedings of the 7th European Conference on Networks & Optical Communications (NOC2002)*, Darmstadt, Germany, June 18-21 2002, pp. 293–300.

[80] A. Kvalbein, S. Gjessing, and F. Davik, "Performance evaluation of an enhanced bridging algorithm in RPR networks," in *Proceedings 3rd International Conference on Networking (ICN'04)*, vol. 2, Guadeloupe, French Caribbean, February 29-March 4 2004, pp. 760–767.

[81] F. Davik, M. Yilmaz, S. Gjessing, and N. Uzun, "IEEE 802.17 Resilient Packet Ring background and overview," Simula Research Laboratory, Tech. Rep. 2003-11, December 2003. [Online]. Available: http://www.simula.no

[82] P. Yuan, V. Gambiroza, and E. Knightly, "The IEEE 802.17 media access protocol for high-speed metropolitan-area Resilient Packet Rings," *IEEE Network*, vol. 18, no. 3, pp. 8–15, May-June 2004.

[83] G. Remšak and J. Schuringa, "Resilient Packet Ring - IEEE 802.17," *Elektrotechnik und Informationstechnik*, vol. 121, no. 6, pp. 235–238, June 2004.

[84] F. Davik, A. Kvalbein, and S. Gjessing, "An analytical bound for convergence of the Resilient Packet Ring Aggressive mode fairness algorithm," in *Proceedings of the 40th annual IEEE International Conference on Communications*, Seoul, Korea, May 16-20 2005.

[85] A. Shokrani, I. Lambadaris, and J. Talim, "On modeling of fair rate calculation in Resilient Packet Rings," in *Proceedings of the 10th IEEE Symposium on Computers and Communications (ISCC 2005)*, June 27-30 2005.

[86] D. Schupke and A. Riedl, "Packet transfer delay comparison of a store-and-forward and a cut-through Resilient Packet Ring," in *Proceedings of the 2002 International Zurich Seminar on Broadband Communications, 2002. Access, Transmission, Networking*, February 19-21 2002, pp. 12–1–12–5.

[87] F. Davik, A. Kvalbein, and S. Gjessing, "Improvement of Resilient Packet Ring fairness," in *Proceedings of the 48th annual IEEE Global Telecommunications Conference (GLOBECOM 2005)*, St. Louis, Missouri, USA, November 28 – December 2 2005, to appear in.

[88] F. Davik and S. Gjessing, "Improvement of Resilient Packet Ring Fairness," Simula Research Laboratory, Tech. Rep. 2005-02, February 2005, Revised August 2005. [Online]. Available: http://www.simula.no

[89] F. Davik, A. Kvalbein, and S. Gjessing, "Congestion domain boundaries in Resilient Packet Rings," Simula Research Laboratory, Tech. Rep. 2005-03, February 2005, Revised August 2005. [Online]. Available: http://www.simula.no

[90] E. Knightly, L. Balzano, V. Gambiroza, Y. Liu, P. Yuan, S. Sheafor, and H. Zhang, "Achieving high performance with Darwin's fairness algorithm," March 2002, presentation at IEEE 802.17 Meeting. [Online]. Available: http://grouper.ieee.org/groups/802/17/documents/presentations/mar2002

[91] F. Alharbi and N. Ansari, "Low complexity distributed bandwidth allocation for Resilient Packet Ring networks," in *Proceedings of 2004 Workshop on High Performance Switching and Routing, 2004. HPSR*, April 18-21 2004, pp. 277–281.

[92] X. Zhou, D. Jin, and L. Zeng, "A novel fairness algorithm based on rate estimation in Resilient Packet Ring," in *Proceedings of the SPIE: Network Architectures, Management, and Applications*, vol. 5282, April 2004, pp. 134–142.

[93] X. Zhou, G. Shi, H. Fang, and L. Zeng, "Fairness algorithm analysis in Resilient Packet Ring," in *Proceedings of the 2003 International Conference on Communication Technology (ICCT 2003)*, vol. 1, April 9-11 2003, pp. 622–624.

[94] Y. F. Robichaud and C. Huang, "Improved fairness algorithm to prevent tail node induced oscillations in RPR," in *Proceedings of the 40th annual IEEE International Conference on Communications*, Seoul, Korea, May 16-20 2005.

[95] F. Davik, A. Kvalbein, and S. Gjessing, "Performance evaluation and improvement of non-stable Resilent Packet Ring behavior," in *Proceedings, Part II of the 4th International Conference on Networking (ICN'05)*, ser. LNCS 3421, Reunion Island, April 17-21 2005, pp. 551–563.

[96] ——, "Resilent Packet Ring low priority traffic latency," in *Proceedings of the 2005 International Conference on Communications in Computing: CIC 2005*, Las Vegas, Nevada, USA, June 27-30 2005, pp. 35–40.

[97] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for differentiated services," December 1998, IETF, RFC 2475.

[98] M. A. El-Gendy, A. Bose, and K. G. Shin, "Evolution of the Internet QoS and support for soft real-time applications," *Proceedings of the IEEE*, vol. 91, 2003.

[99] B. Davie, A. Charny, J. Bennett, K. Benson, J. L. Boudec, W. Courtney, S. Davari, V. Firoiua, and D. Stiliadis, "An Expedited Forwarding PHB (Per-Hop Behavior)," March 2002, IETF, RFC 3246.

[100] F. Davik and S. Gjessing, "Applying the DiffServ model to a Resilient Packet Ring network," Simula Research Laboratory, Technical Report 2005-01, February 2005, Revised August 2005. [Online]. Available: http://www.simula.no

[101] ——, "Applying the DiffServ model to a Resilient Packet Ring network," in *Proceedings of Networking 2005*, ser. LNCS 3462. University of Waterloo, Waterloo Ontario Canada: IFIP International Federation for Information Processing, May 2005, pp. 1461–1464.

[102] X. Wang, B. Huang, X. Yu, and F. Zhang, "Edge QoS study of RPR equipment," *Proceedings of the SPIE The International Society for Optical Engineering*, vol. 5281, no. 1, pp. 396–403, 2004.

Part II

Research Papers

Paper I

IEEE 802.17 Resilient Packet Ring Background and Overview

Fredrik Davik, Mete Yilmaz, Stein Gjessing and Necdet Uzun

Published: A shorter version of this paper was published in IEEE Communications Magazine, March 2004.

Author Contribution: In this paper, the first three authors (Davik, Yilmaz and Gjessing) contributed equally to the general discussions on the properties of the RPR technology. The contributions from the last author, Uzun, are more limited.

Davik ran and explained all the simulation results in the paper (except for the latency plot (Fig. 4)).

IEEE 802.17 Resilient Packet Ring Background and Overview

Fredrik Davik, Mete Yilmaz, Stein Gjessing, Necdet Uzun

Abstract — **The IEEE Working group P802.17 is standardizing a new ring topology network architecture, called the Resilient Packet Ring (RPR), to be used mainly in metropolitan and wide area networks. This paper presents a technology background, gives an overview, and explains many of the design choices the RPR working group faced during the development of the standard. Some major architectural features are illustrated and compared by showing performance evaluation results using the RPR simulator developed at Simula Research Laboratory using the OPNET Modeler simulation environment.**

Index Terms— **Communications, Networking, MAN, WAN, Ring networks, Spatial reuse, Fairness.**

Fredrik Davik is with Simula Research Laboratory and Ericsson Research
Mete Yilmaz is with Cisco Systems
Stein Gjessing is with Simula Research Laboratory and is a visiting scholar at Department of Computer Engineering, San Jose State University
Necdet Uzun is with Cisco Systems

A shorter version of this report is to appear in IEEE Communications Magazine March 2004.

Simula Research Laboratory, Technical Report 11-2003, December 2003

I. INTRODUCTION

The Resilient Packet Ring (RPR, IEEE 802.17) is the latest development in a series of ring based network protocols standardized by IEEE [6]. IEEE 802.5 Token Ring [7] and IEEE 1596 Scalable Coherent Interface (SCI) [8] are examples of other ring based IEEE standards. Packet ring based data networks were pioneered by the Cambridge Ring [10], followed by other important network architectures, notably MetaRing [2], FDDI [12], ATMR [13] and CRMA-II [14].

Rings are in general built using several point-to-point connections. When the connections between the stations are bidirectional, rings allow for resilience (a frame can reach its destination even in the presence of a link failure). A ring is also simpler to operate and administrate than a complex mesh or an irregular network.

Networks deployed by service providers in the MANs or WANs are often based on SONET/SDH rings. Many SONET rings consist of a dual-ring configuration in which one of the rings is used as the back-up ring that remains unused during normal operation and utilized only in the case of failure of the primary ring [1]. The static bandwidth allocation and network monitoring requirements increase the total cost of a SONET network. While Gigabit Ethernet does not require static allocation and provides cost advantages; it cannot provide desired features such as fairness and fast (<50ms) auto-restoration.

In order to provide efficient, carrier class packet transport, some companies started to develop proprietary ring technologies. For example, Cisco Systems developed the Dynamic Packet Transport (DPT) [20] technology based on the Spatial Reuse Protocol (SRP) [19], and Nortel Networks developed the OPTera Packet Edge technology [16].

In order to standardize these new initiatives, IEEE was approached. One of the goals of the new standard is to utilize the simplicity of ring networks and use the bandwidth of the dual-ring as efficiently as possible for high-speed data transmission in MANs and in WANs. Important goals also includes distribution of bandwidth fairly to all active stations while providing fast auto restoration. For a rapid and widespread deployment, the reuse of existing physical layers is another important goal. To achieve all of this, the IEEE working group P802.17 was formally started in March 2001 under the name Resilient Packet Ring.

Since RPR is being standardized in the IEEE 802 LAN/MAN (Ethernet) families of network protocols, it can inherently bridge to other IEEE 802 networks and mimic a broadcast medium. RPR implements a Medium Access Control (MAC) protocol, for access to the shared ring communication medium, which has a client interface similar to that of Ethernet's.

The rest of this paper is organized as follows: In section II and III respectively ring network basics and RPR station design is discussed. The so-called fairness algorithm is the topic of section IV, while sections V, VI and VII treat topology discovery, resilience and bridging. Finally frame formats are outlined in section VIII, and a conclusion is given. In order to demonstrate different operational modes, some performance figures are included and discussed. The scenarios have been executed on the RPR simulator model developed at Simula Research Laboratory and implemented in OPNET Modeler [15], according to the latest RPR draft standard as of November 2003 (v3.0).

II. RING NETWORK BASICS

To facilitate discussion of design choices that were made in the development phase of RPR, this section introduces some basic ring networking principles, not all implemented in RPR.

In unicast addressing (broadcast will be covered later), frames are added on to the ring by a sender station, that also decides on which of the two counter rotating rings (called ringlet 0 and ringlet 1 in RPR) the frame should travel to the receiving station. The destination address in the frame header might not be the exact address of the receiving station itself. However, the station should, based on the destination address, recognize that it is the receiving station for this frame. In this way the transmission on the ring (from sender to receiver) might be only one hop in a multi hop transmission from source to destination.

If a station does not recognize the destination address in the frame header, it transits the frame, i.e. the frame is forwarded to the next station on the ring. In RPR, the transit methods supported are cut through (the station starts to forward the frame before it is completely received) and store and forward.

To prevent frames, with a destination address recognized by no stations on the ring, to circulate forever, a time to live field (TTL) is decremented by all stations (as in RPR) or by one station (as in SCI) on the ring. Frames received with a TTL value of 0 is not passed on to downstream stations (is stripped from the ring).

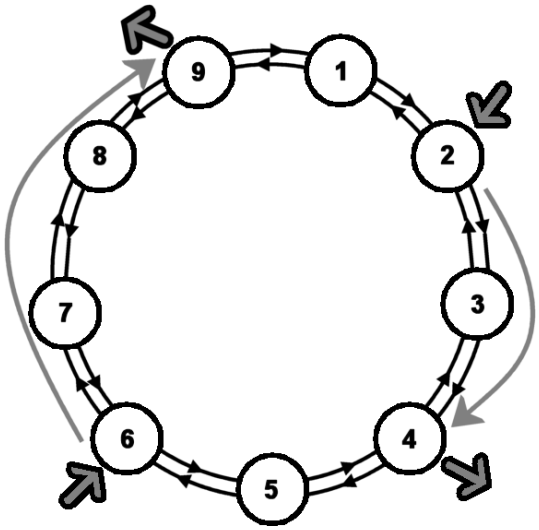


Figure 1. Destination Stripping and Spatial Reuse illustrated on the outer ringlet.

When a station recognizes that it is the receiver of a frame, it may copy the contents of the frame and let the frame traverse the ring back to the sender (like in the Token Ring), it may send back only an acknowledgement (if the station is able to receive the frame) or a

negative acknowledgement (if the station is unable to receive the frame) back to the sender (like in SCI), or it may remove the frame completely from the ring (like in RPR). When the receiving station removes the frame from the ring, the bandwidth otherwise consumed by this frame on the path back to the source, is available for use by other sending stations. This is generally known as spatial reuse.

Figure 1 shows an example scenario where spatial reuse is obtained on the outer ringlet; station 2 is transmitting to station 4 at the same time as station 6 is transmitting to station 9. Destination stripping with spatial reuse was previously exploited in systems like MetaRing [2], ATMR [13], CRMA-II [14] and SCI [8].

The ring access method is an important design choice. A token may circulate the ring, so that the station holding the token is the only station allowed to send (like in Token Ring). An alternative access method, called a "buffer insertion" ring, was developed as early as 1974 [5][17]. Every station on the ring has a buffer called an "insertion buffer" (called a "transit queue" in RPR, see Figure 2) in which frames transiting the station may be temporarily queued. The station must act according to three simple rules. The first principle is that, the station will not add packets to the ring as long as there are packets in the insertion buffer or packets in transit. Secondly, when there is no frame in transit, the station itself is allowed to add a frame. Thirdly, if a passing frame arrives at a station when it has started to add a frame, the frame in transit is temporarily (for as long as it takes to complete the sending of the added frame) queued in the insertion buffer.

Obviously these three simple principles need some improvement to make up a full, working protocol that distributes bandwidth fairly. This has been studied before [3][11][4][9], and how this is achieved in RPR will be revealed in section IV when the RPR fairness algorithm is discussed.

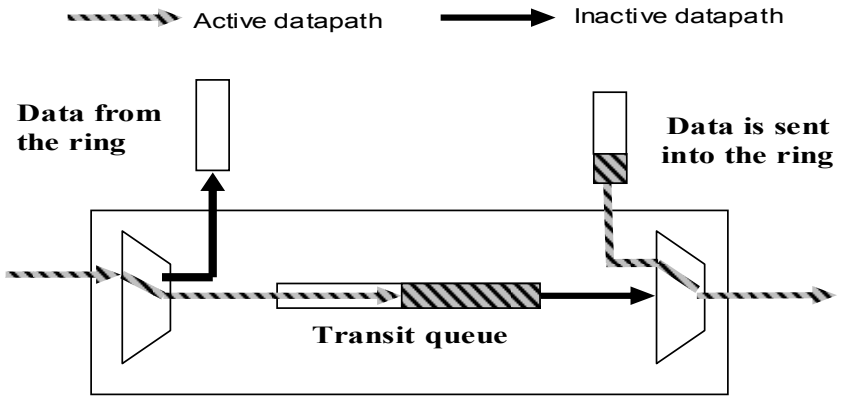


Figure 2. The "insertion buffer" or "transit queue" stores frames in transit, while the station itself adds a frame (here a stations attachment to only one ring is shown).

III. STATION DESIGN AND PACKET PRIORITY

The stations on the RPR ring implements a medium access control (MAC) protocol that regulates the stations access to the ring communication medium. All links around the ring is mandated to have the same capacity (called the full rate). The P802.17 working group has defined several physical layer interfaces (reconciliation sublayers) for Ethernet (called PacketPHYs) and SONET/SDH [6]. The MAC also implements access points that clients can call in order to send and receive frames and status information.

The RPR working group decided to implement a three level, class based, traffic priority scheme. The objectives of the class based scheme is to let class A be a low latency, low jitter class, class B be a class with predictable latency and jitter, and finally class C be a best effort transport class. It is worth to note that the RPR ring does not discard frames to resolve congestion. Hence when a frame has been added onto the ring, even if it is a class C frame, it will eventually arrive at its destination. The design decision behind this choice will be explained later

Class A traffic is divided into classes A0 and A1, and class B traffic is divided into class B-CIR (Committed Information Rate) and B-EIR (Excess Information Rate). The two traffic classes C and B-EIR are called Fairness Eligible (FE), for reasons that will become clear in section IV.

The bandwidth around the ring is pre-allocated in two ways. The first is called "reserved" and can only be used by class A0 traffic, and is equally reserved all around the ringlet. If stations are not using their pre-allocated A0 bandwidth, this bandwidth is wasted. In this way TDM-like traffic can be sent by RPR stations as A0 frames.

The other pre-allocated bandwidth is called “reclaimable”. A station that has class A1 or B-CIR traffic to send, pre-allocates “reclaimable” bandwidth for these types of traffic. If not in use, such bandwidth can be used by FE traffic. In addition, any bandwidth not pre-allocated is also used to send FE traffic. The distribution and use of unallocated and unused reclaimable bandwidth (FE bandwidth) is dynamically controlled by the fairness algorithm.

A station’s reservation of class A0 bandwidth is broadcasted on the ring using topology messages (topology messages will be discussed later). Having received such topology messages from all other stations on the ring, every station calculates how much bandwidth to reserve for class A0 traffic. The remaining bandwidth, called “unreserved rate” can be used for all other traffic classes. This in general means that the unreserved rate is the full rate minus the sum of all station’s A0 reservations. The standard claims that spatial reuse of A0 bandwidth is possible but does not specify how this can be obtained.

An RPR station implements several traffic shapers that limit and smooth the add- and transit traffic. There is one shaper for each of the traffic classes A0, A1, B-CIR as well as one for FE traffic and one for non-A0 traffic. The shapers for class A0, A1, B-CIR and non-A0 traffic are preconfigured, while the FE shaper is dynamically adjusted by the fairness algorithm. The non-A0 shaper (also called the Downstream shaper) limits the amount of non class A0 traffic transmitted, ensuring that reserved bandwidth is available all around the ring. The rate of this shaper is the full rate minus the bandwidth reserved for class A0 traffic (i.e. unreserved rate).

When a station tries to send more class B traffic than the B-CIR shaper allows, the rest of the class B traffic is sent as class B-EIR. Class B-EIR traffic has higher add-priority than class C traffic.

It is important that the preconfigured shapers are correctly set, in particular that the sum of the bandwidth allocated for A0, A1 and B-CIR traffic does not surpass the maximum bandwidth (the full rate).

It is not obvious that class A0 pre-allocated and reserved bandwidth may be spatially reused. For example stations 2 and 6 in figure 1 may both send 10Mbit/sec class A0 traffic to respectively stations 4 and 9. If this is the only class A0 traffic reserved, then ideally, the spatial reuse feature should allow us to reserve only 10Mbit/s all around the ring. However, when the constant "reserved bandwidth" is calculated, it is calculated as the sum of all A0 allocations. If the downstream shaper is set to unreserved rate (full rate minus reserved rate), no spatial reuse will be achieved for A0 traffic. If, on the other hand, the downstream shapers are set lower than the unreserved rate (because of known spatial reuse of A0 traffic), spatial reuse of A0 traffic is indeed achieved. RPR does not contain mechanisms to ensure the intended spatial reuse, and it may even not be mandated by the final standard to set the downstream shaper to any other value than the unreserved rate. Anyhow, if stations 2 and 6 can not be trusted to send to non-overlapping segments of the ring, a total of 20 Mbit/sec must be reserved around the ring for class A0 traffic. Misconfiguration of the downstream shaper may cause serious problems at run time.

Also class A1 and C-BIR traffic may be spatially reused, so that the total pre-allocated bandwidth on any link may be calculated taking spatial reuse into consideration, in the same way as explained above for class A0 traffic. But also for these classes of service, it is outside the scope of the RPR standard to enforce spatial reuse. Hence, also here it might be wise to assume that all stations send all their class A1 and B-CIR traffic all around the ring, and that the total pre allocated class A1 and B-CIR bandwidth is the sum of all station's allocations. The difference between allocation of A1 and B-CIR bandwidth with or without spatial reuse, only affects the calculation that is needed to ensure that the total ring capacity is not surpassed. Since unused class A1 and B-CIR bandwidth is reclaimable, this unused bandwidth may anyhow be used by the fairness algorithm to send FE-traffic (class B-EIR and class C traffic).

The minimum transit queue size is the maximum transfer unit that a station itself may add (because this is the maximum buffer size needed by the frames in transit while the station adds a new frame). Some flexibility for scheduling of frames from the add- and transit-path can be obtained by increasing the size of the transit queue. For example, a station may add a frame even if the transit queue is not completely empty. Also a larger queue may store lower priority transit frames while the station is adding high priority frames. The transit queue could have been specified as a priority queue, where frames with the highest priority are dequeued first. This was considered too complex and instead the working group decided that a station optionally may have two transit queues. Then high priority transit frames (class A) are queued in the Primary Transit Queue (PTQ), while class B and C frames are queued in the Secondary Transit Queue (STQ). Forwarding from the PTQ has priority over the STQ and most types of add traffic. Figure 3 shows one ring interface, with the three add- and two transit queues. The numbers in the circles indicate a crude priority on the output link. Regarding priority between add traffic and the STQ, as the STQ fills up, it will have increasingly higher priority (this is not a linear function, but based on thresholds). Since class A frames have priority over all other

traffic, a class A frame traveling thee ring will usually experience not much more than the propagation delay and some occasional transit delays waiting for outgoing packets to completely leave the station (RPR does not support pre-emption of packets).

When in transit, both class B and C frames are stored in the STQ, hence, once added to the ring, they experience delay values within the same range. The difference between class B and class C frames is the scheduling at the ingress. Class B-CIR (as well as class A1) add frames have priority over the class B and class C frames in the STQ (as long as the STQ is not completely full). The worst case delay for class B-CIR frames is the propagation delay, plus the maximum delay that both B and C frames can experience in passing through the (FIFO) STQs on their way around the ring to the receiver. Class B-EIR and class C frames may have to wait very long to get onto the ring, depending on how much bandwidth is consumed by A and B-CIR frames, and how many other stations are adding class B-EIR and class C traffic. Hence it is very hard to give any bound on the latency of class B-EIR and class C frames.

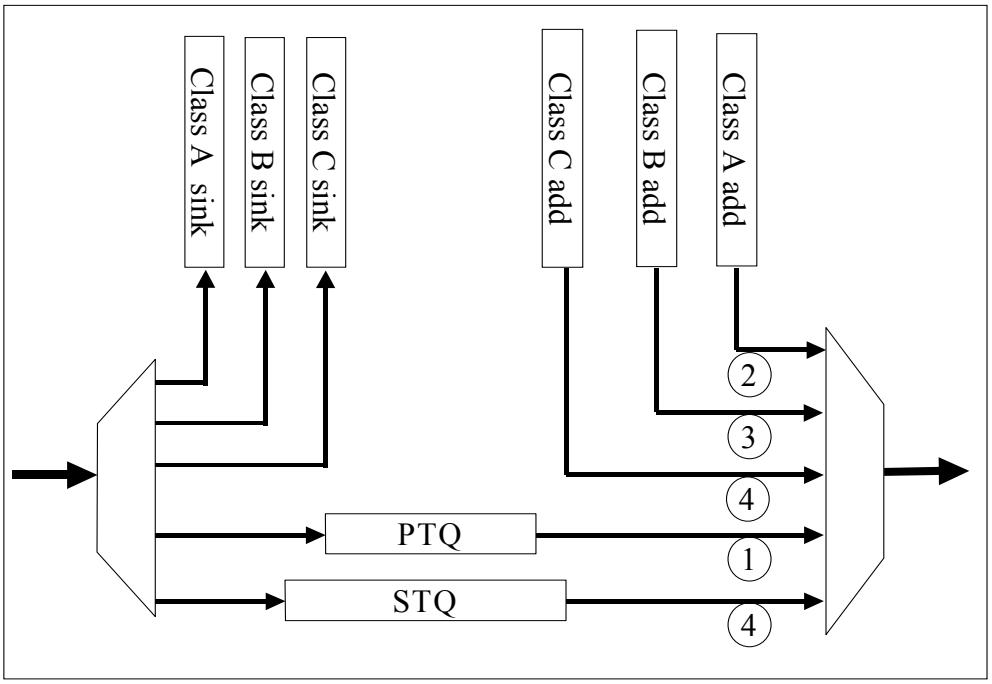


Figure 3. The attachment to one ring by a Dual Transit Queue Station. The numbers in the circles give a very crude indication of output link priority.

An RPR station may, however, have one transit queue only (the PTQ). In order for class A traffic to move quickly around the ring, the transit queues in all single transit queue stations should then be almost empty. This is achieved by letting transit traffic have priority over all add traffic, and by requiring all class A traffic to be reserved (class A0). Hence there will always be room for class A traffic and class B and C traffic are competing for the remaining bandwidth, just like in the two transit queue stations.

Figure 4 shows part of an example run where the latency of frames sent between two

given stations on an RPR ring is measured. The ring is overloaded with random background, class C, traffic. Latency is measured from the time a packet is ready to enter the ring (i.e. first in the ingress queue), until it arrives at the receiver. Notice how class A traffic keeps its low delay even when the ring is congested. Also notice how class B traffic still have low jitter under high load, while class C traffic experiences some very high delays. This example was run with two transit queue stations.

An RPR ring may consist of both one and two transit queue stations. The rules for adding and scheduling traffic are local to the station, and the fairness algorithm described below works for both station designs.

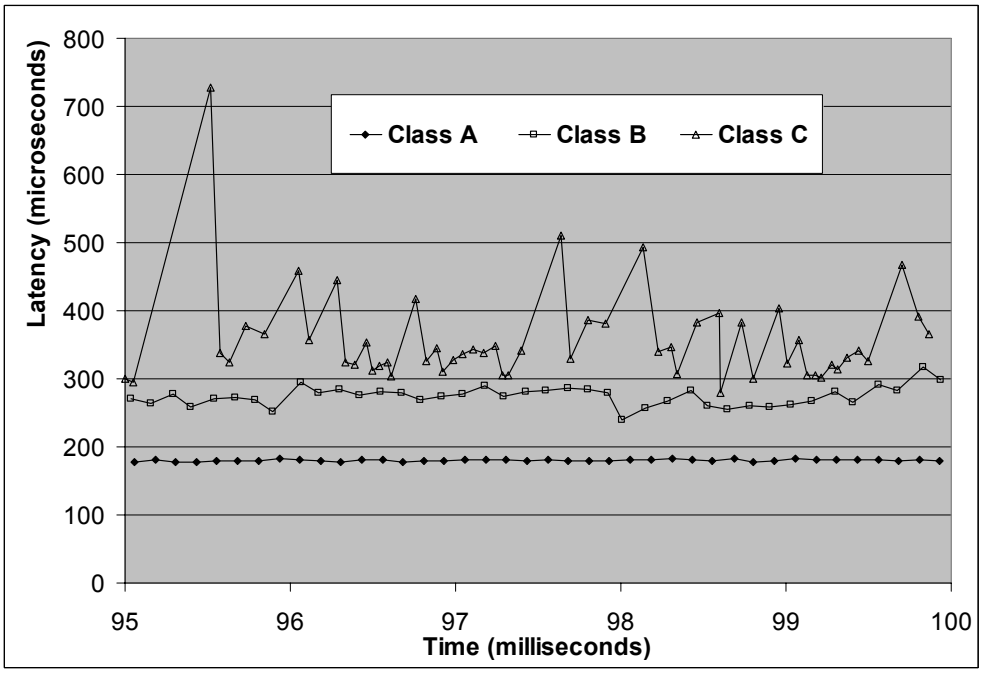


Figure 4. Frame latency from station 1 to station 7 on a 16 station overloaded ring. The propagation and minimum frame latency is 180 microseconds.

IV. THE RPR FAIRNESS ALGORITHM

In the basic "buffer insertion" access method, a station may only send a frame if the transit queue is empty. Hence it is very easy for a downstream station to be starved by upstream stations. In RPR, the resolution to the starvation problem is to enforce all stations to behave according to a specified "fairness" algorithm.

The objective of the fairness algorithm is to distribute unallocated and unused reclaimable bandwidth fairly among the contending stations and use this bandwidth to send class B-EIR and class C traffic, i.e. the fairness eligible (FE) traffic. Class A0 traffic is obviously not affected, since bandwidth is reserved for this class exclusively. Classes A1 and B-CIR are indirectly affected, as will be explained below.

When defining fair distribution of bandwidth, the P802.17 working group decided to enforce the principle that when the demand for bandwidth on a span of the ring is greater

than the supply, the available bandwidth should be fairly distributed between the contending source stations. This is the same principle as outlined by the so called RIAS fairness [4]. The working group also decided that a weight is assigned to each station so that a fair distribution of bandwidth need not be an equal one.

There are several ways to ensure fair allocation of bandwidth around the ring. One way is an algorithm that enquires around the ring how much bandwidth each of the stations need, and then notifies them afterwards how much they were allocated [18]. RPR takes another approach. When the bandwidth on the output link of a station is exhausted (the link is congested), the fairness algorithm starts working to distribute this bandwidth fairly. The most probable cause of congestion is the station itself and its immediate upstream neighbors. Hence by sending a so called fairness message upstream (on the opposite ring) the probable cause of the congestion is reached faster than by sending the fairness message downstream over the congested link. Figure 5 shows how the attachment to one ring asks the other attachment to queue and send a fairness message. In the sequel we focus on fairness on one ring. The fairness algorithm on the other ring works exactly the same way.

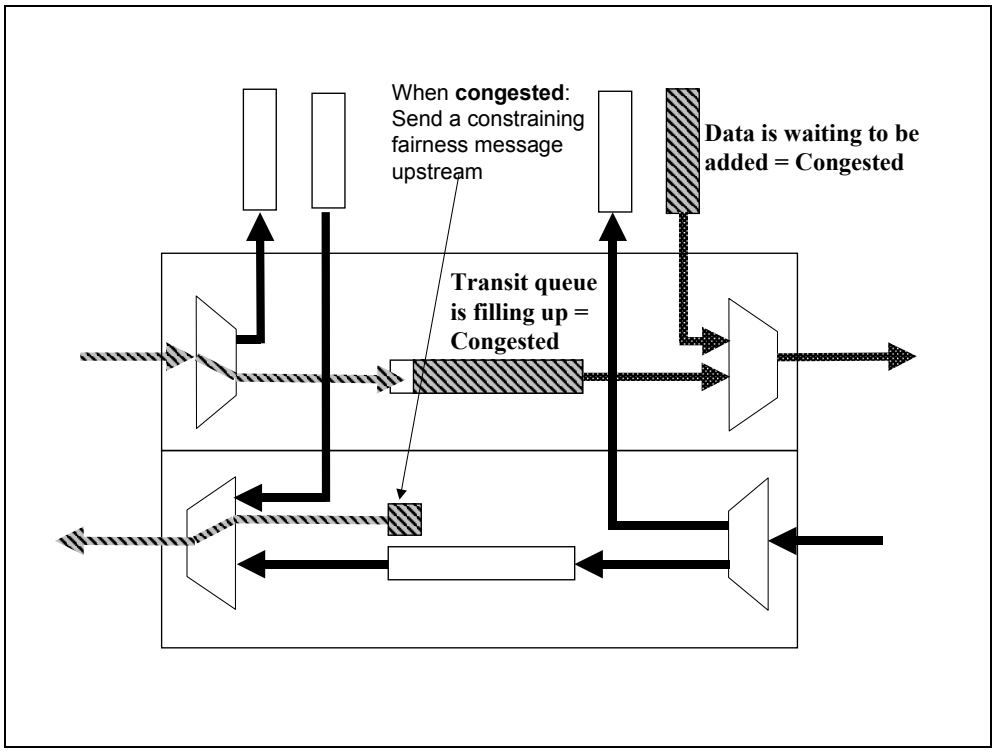


Figure 5. When a station becomes congested it sends a fairness message upstream.

That a link is congested (or that it is very close to being congested) may be identified in several ways by the attached upstream station: a) Frames that are to be added have to wait a long time before they are forwarded, b) the transit buffer is filling up (and hence transit frames have to wait a long time before they are sent on), and c) if the capacity of the

transmit link is (almost) fully used. The first method is used in a single transit queue design, the second method is used in the dual transit queue design, and the third method is used in both designs.

The congested station tries to make a first estimate at what the fair rate should be. After an estimate is calculated, this value is sent upstream in a fairness message. The station receiving the fairness message will decrease it's add rate to this value. If this station also transits more traffic than this value, the fairness message is transmitted further upstream, in order to tell more stations to decrease their add rate. In this way all stations upstream from the congestion point, that are contributing to the congestion, are notified and have to throttle their add traffic to the rate given in the fairness message originating from the congestion point. This segment of the ring (that receives a fairness message with the same value) is called a congestion domain.

There are two methods specified for fair rate estimation and adjustment: 1) The Aggressive method: 2) The Conservative method. The goal of the Aggressive fairness method is to quickly respond to changes in the traffic load. The Conservative option performs rate adjustments more restrictively. It makes an adjustment and waits to observe the effect before making a new decision. Since the station design with one transit queue needs a more restrictive rate control, the conservative method of rate estimation and adjustment may be a good match.

The main difference between conservative and aggressive fairness is the way the fair rate is initially estimated, and how it is adjusted towards the real fair rate. In the conservative mode, the congested station calculates the initial fair rate either by 1) dividing the available bandwidth between all upstream stations that are currently sending frames through this station or by 2) use its own current add rate. A timer is used to ensure that additional rate changes are made only when the congested station have had time to see how this new fair rate affects the congestion (i.e., gets better or worse). The period of this timer is referred to as the Fairness Round Trip Time (FRTT). FRTT is an estimate of the time it takes for a congested station to see the full effect of the fairness message it sent to upstream stations. FRTT consists of two parts: 1) the propagation delay for a class A frame when transmitted from the congestion domain head (i.e. the congested station) to the congestion domain tail (the station at the other end of the congestion domain) and back (LRTT – Loop Round Trip Time). 2) The difference between the propagation delay for a class C and a class A frame sent from the tail to the head (FDD – Fairness Differential Delay). LRTT needs to be computed on initialization of the ring and when the topology changes, while FDD is computed when a station becomes tail of a congestion domain and thereafter at configurable intervals. FDD reflects the congestion situation, i.e. the STQ fill levels on the transit path from head to tail. As the congestion domain changes, so does the FRTT. LRTT and FDD frames are special types of control frames.

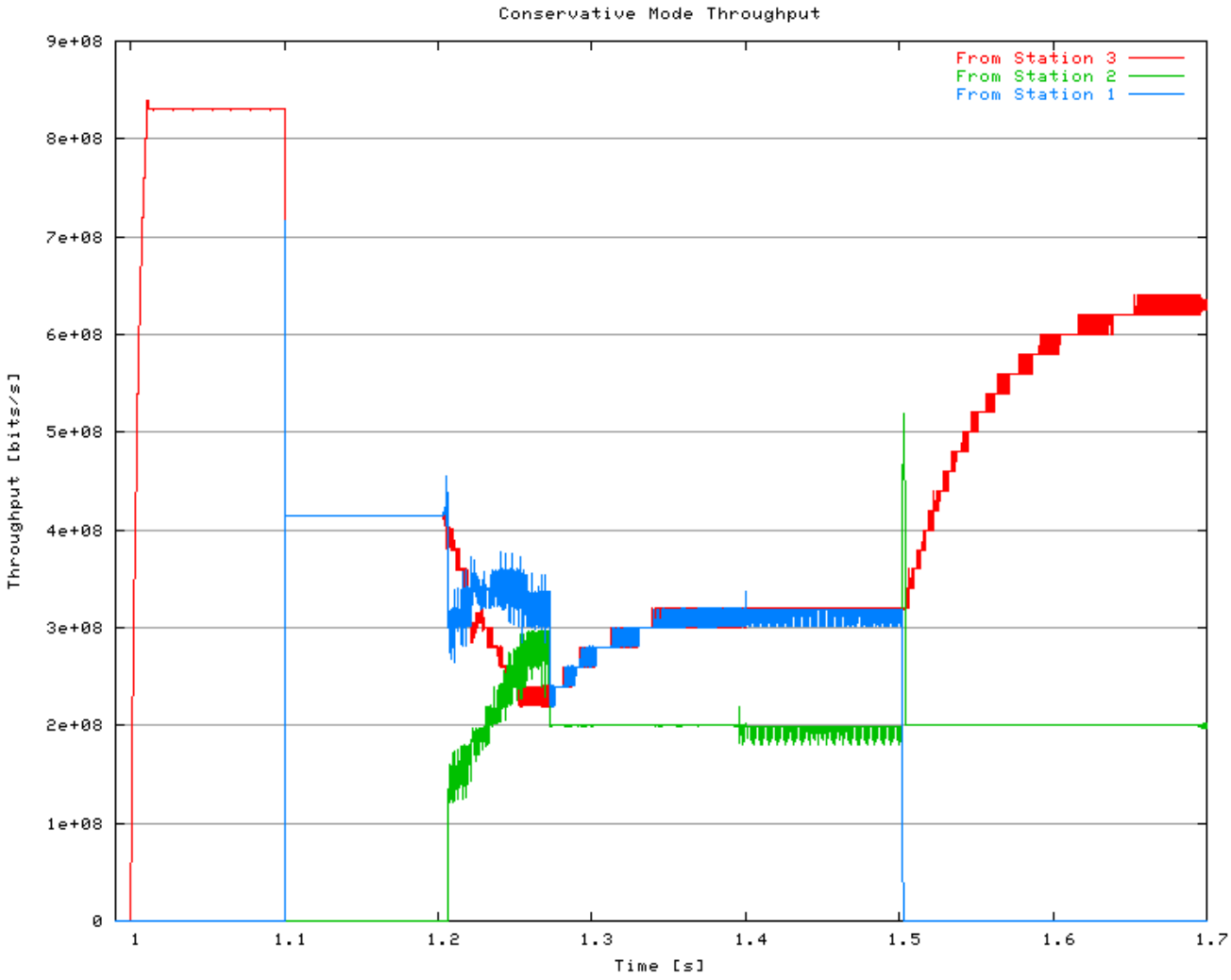


Figure 6. Dynamic traffic handled by the conservative fairness algorithm. Number of bits/sec. as received by station 4.

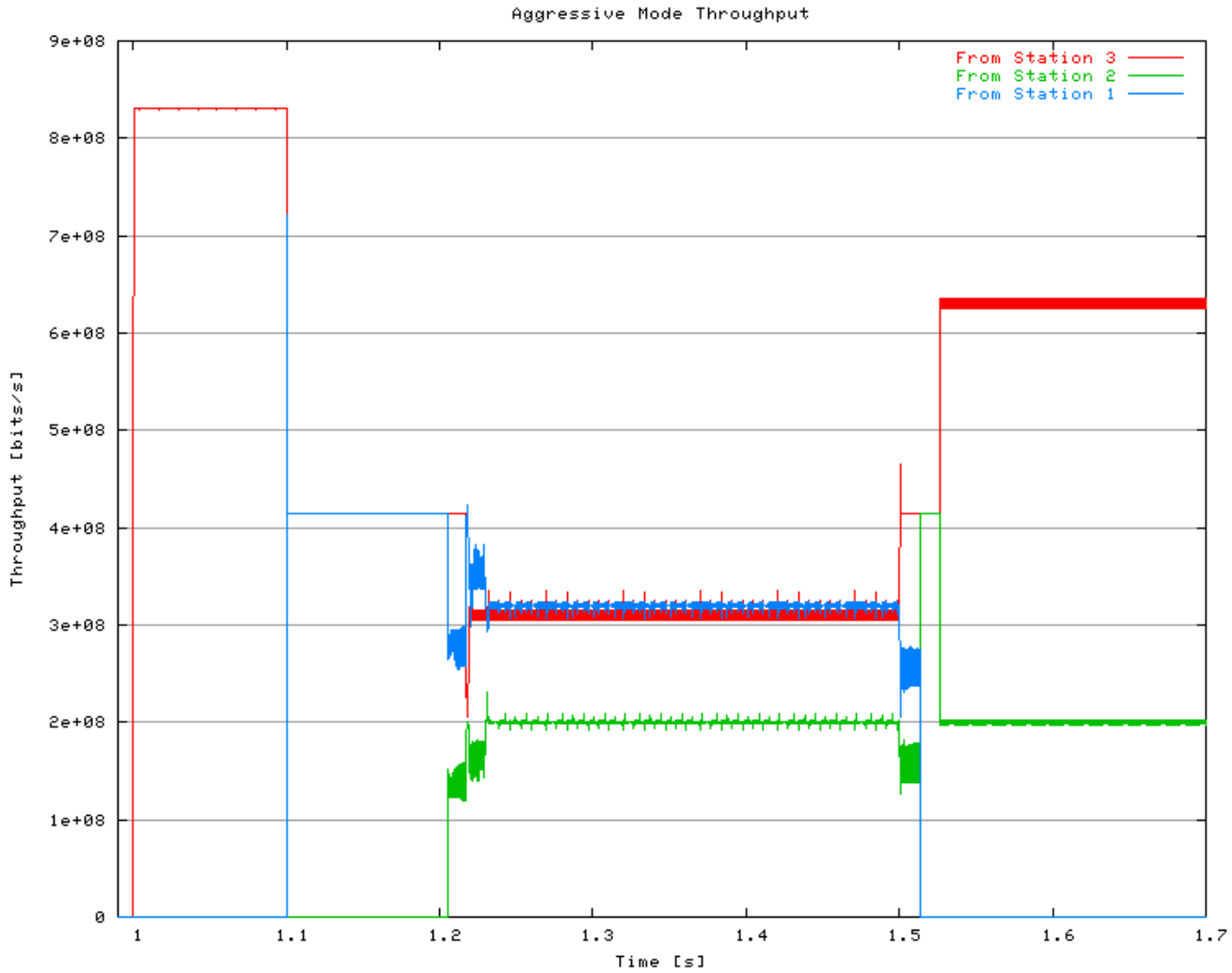


Figure 7. Dynamic traffic handled by the aggressive fairness algorithm. Number of bits/sec. as received by station 4.

In the Aggressive mode, the congested station makes a first estimate of the fair rate equal to the rate the station itself lately have been able to add to the ring. Since the station is congested, this means that it has been able to send very little traffic onto the ring recently. Hence this estimate is probably too low, but it is used as a starting point and a way to alleviate congestion. When congestion is indeed removed, the (previously congested) station will not send any more fairness messages upstream, or more correctly it will send fairness messages with a default fair value representing the full link rate (such frames are sent all the time with preset intervals as heart beats.) A station receiving a fairness message indicating no congestion (i.e., full link rate) will increase its add traffic (assuming the station's demand is greater than what it is currently adding). In this way (if the traffic pattern is stable) the same station will become congested again after a while, but this time the estimated fair rate will be closer to the real fair rate, and hence the upstream stations do not have to decrease their traffic rate as much as previously.

Figure 6 and Figure 7 shows how respectively the conservative and the aggressive fairness algorithm work for a given scenario. Both scenarios are simulated on a 16-station ring, with 50 km one Gbit/sec links and 500 Byte packets, of which each station uses 1 % for A0 traffic. Stations 1, 2 and 3 are sending to station 4. The traffic starts at time 1.0 sec., and initially only station 3 is sending. At time 1.05 sec. station 1 starts sending. Both of these flows are greedy class C flows, and both fairness methods are quick to share the bandwidth on the congested link (from station 3 to station 4) equally. At time 1.2 sec. station 2 starts sending a 200 Mbit/sec flow (also class C frames to station 4). We see that the aggressive method very quickly (but after some high oscillations) changes to the new fair distribution of bandwidth. The conservative method, however, waiting longer between every rate adjustment, uses more time to converge to the new fair division of add rates.

At time 1.5 sec., the traffic from station 1 stops. For both methods we see that some traffic from station 2 that has been queued, now are being released, and hence there are an added number of station 2 packets received at station 4. The aggressive method has some additional oscillations, but otherwise adjusts quickly the new traffic pattern. The conservative method adjusts with fewer oscillations, but more slowly.

The fairness algorithm affects class A1 and B-CIR traffic indirectly. When stations have less such traffic to send than they have pre-allocated, this bandwidth may be used to send FE traffic. When a station, at a later time, wants to reclaim this bandwidth (now it has more A1 or B-CIR traffic to send), this bandwidth is taken away from the FE traffic, and a link may now be (even worse) congested. Because class A1 and B-CIR add traffic have priority over traffic from the STQ, the STQ occupancy increases while the fairness algorithm tries to throttle upstream FE traffic senders. The STQ should be sized so that it does not overflow when this situation occurs (discussed further below).

In the single transit queue station design, the transit traffic has priority over all add traffic, hence no frame will ever be lost on the way around the ring. In a dual transit queue architecture, class A1 and B-CIR add traffic has priority over traffic transiting in the STQ. This may cause the STQ to fill up, and if the fairness algorithm does not work fast enough (is not able to stall upstream traffic fast enough), and the STQ is small, then the STQ may fill completely. If the STQ is full, and there is class A or B-CIR traffic to be added, there is a design choice between throwing away frames in the STQ, or stalling the class A and B-CIR add traffic. The latter choice would cause priority inversion. Because the STQ should never be completely full if the system is properly configured (the STQ is large enough) and the stations do not send excess traffic, the RPR working group decided that RPR will not throw frames away. A short priority inversion period is what will happen in the unlikely event that an STQ becomes full.

As noted above, the way to avoid priority inversion in the dual transit queue case, is to have a large STQ that is able to store class B-EIR and class C traffic while the station adds class A1 and B-CIR traffic and the fairness algorithm tries to alleviate congestion. The negative factors in having a large STQ is obviously that traffic transiting a very full STQ will experience severe delay. However, if the STQ's were not that large, the delay would be experienced in the ingress buffers instead. The fairness algorithm is triggered when the STQ becomes partially full. This threshold is named "stqLowThreshold" and the default value is 1/8 of the size of the STQ. By having a large STQ, some variation in traffic load

may be experienced and smoothed by the STQs even without the fairness algorithm kicking in.

In the fairness algorithm, as explained above, a station sees at most one congestion point. There is an optional fairness message in RPR called a multi choke fairness message. Each station on the ring puts its own congestion status (that tell how much its output link is used presently) in such a message and sends it to all the other stations on the ring. A receiving station may collect these messages and build a global image of the congestion situation on the ring, and schedule its add traffic accordingly. How this is done is however outside the scope of the standard.

v. TOPOLOGY DISCOVERY

Topology discovery determines connectivity and the ordering of the stations around the ring. This is accomplished by collecting information about the stations and the links, via the topology discovery protocol. The collected information is stored in the topology databases of each station.

At system initialization, all stations send control frames, called topology discovery messages, containing their own status, around the ring. Topology messages are always sent all the way around the ring, on both ringlets, with an initial TTL equal to 255 (the maximum number of stations). All other stations on the ring receive these frames. Upon reception of such a frame, the station has two pieces of information, namely 1) the distance, measured in hops, to the origination station (calculated from 255 – frame.ttl + 1) 2) the capabilities of the originating stations as described by the frame contents. Having received such frames from all other stations on the ring, each station has enough information to compute a complete topology image.

When a new station is inserted into a ring, or when a station detects a link failure, it will immediately transmit a topology discovery message. If any station receives a topology message inconsistent with its current topology image, it will also immediately transmit a new topology message (always containing only the stations own status). Hence the first station that notices a change starts a ripple effect, resulting in all stations transmitting their updated status information, and all stations rebuilding their topology image.

The topology database includes not only the ordering of the stations around the ring and the protection status of the stations (describing its connected links, with status signal fail, signal degrade, or idle), but also the attributes of the stations and the round trip times to all the other stations on the ring.

Once the topology information has become stable, meaning that the topology image does not change during a specified time period, a consistency check will be performed. For example the station will make sure that the information collected on one ringlet matches the other.

Even under stable and consistent conditions, stations will continue to periodically transmit topology discovery messages, in order to provide robustness to the correct operation of the ring.

When the client submits a frame to the MAC, without specifying which ringlet to be used, the MAC uses the topology database to find the shortest path. Information in the topology database is also used to calculate the Fairness Round Trip Time in the

conservative mode of the fairness algorithm.

VI. RESILIENCE

As described in the previous section, as soon as a station recognizes that one of its links or a neighbor station has failed, it sends out topology messages. When a station receives such a message telling that the ring is broken, it starts to send frames in the only viable direction to the receiver. This behavior, which is mandatory in RPR, is called steering.

The IEEE 802 family of networks have a default packet mode called “strict” in RPR. This means that packets should arrive in the same order as they are sent. To achieve this after a link or station failure, all stations stop adding packets and discard all transit frames until their new topology image is stable and consistent. Only then will stations start to steer packets onto the ring. Even on a 2000 km ring, it will take a maximum of 50 ms for this algorithm to converge, that is from the failure is observed by one station, until all stations have consistent topology databases and can steer new frames.

RPR optionally defines a packet mode/attribute called relaxed, meaning that it is tolerable that these packets arrive out of order. Such packets may be steered immediately after the failure has been detected and before the database is consistent. Relaxed frames will not be discarded from the transit queues either.

When a station detects that a link or its adjacent neighbor has failed, the station may optionally wrap the ring at the break point (called “wrapping”) and immediately send frames back in the other direction (on the other ringlet) instead of discarding them. Frames not marked as wrap eligible (via the *we* frame field) are always discarded at a wrap point.

VII. BRIDGING

The RPR standard specifies methods for networks interconnection by bridging to other network protocols in the IEEE 802 family. An application of this can be transport of Ethernet frames over RPR to provide resilience, class of service support and better utilization of the network.

Figure 8 shows an example, where an RPR ring is bridged to many Ethernet in the First Mile access networks. Station 1 may be the station that includes the interface between the 802-networks (the access and MAN network), and the backbone network.

RPR uses 48-bit source and destination MAC addresses in the same format as Ethernet (see section VIII). When an Ethernet frame is bridged into an RPR ring, the bridge inserts some information into the Ethernet frame in order to transform it into an RPR frame. Similarly this information will be removed if the frame moves from RPR (back) to Ethernet.

When participating in the spanning tree protocol, RPR is viewed as one broadcast enabled subnet, exactly like any other broadcast LAN. The ring structure is then not visible, and incurs no problem for the spanning tree protocol. The spanning tree protocol may not break the ring, but may disable the bridges connected to the RPR stations on the ring.

RPR implements broadcast by sending the frame all around the ring, or by sending the frame half way on both ringlets. In the latter case the TTL field is initially set to a value so that it becomes zero, and the packet is stripped, when it has traveled half the ring. Using broadcast, obviously, no spatial reuse is achieved.

Since RPR can bridge to any other Ethernet, for example Ethernet in the First Mile, we can envision Ethernets spanning all the way from the customer into the Metropolitan or even Wide Area Network. Whether such large and long ranging Ethernets will be feasible or practical in the future, is to be seen.

An extended frame format is also defined in the standard to transport Ethernet frames. In this format an RPR header encapsulates Ethernet frames.

Another way to connect RPR to other data networks is to implement IP or layer 3 routers on top of the MAC clients. In this way RPR behaves exactly like any other Ethernet connected to one or more IP routers. In Figure 8, station 1 may implement an IP-router on top of the MAC client interface. IP routers connected to RPR should in the future also take advantage of the class based packet priority scheme defined by RPR when they send Quality of Service constrained traffic over RPR.

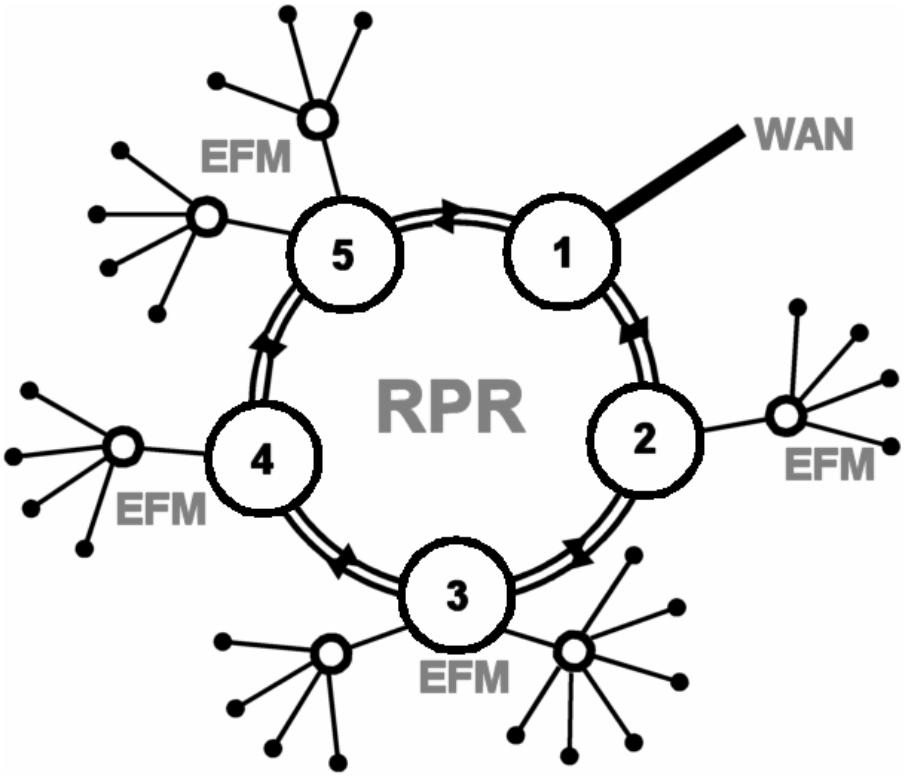


Figure 8. Example RPR-EFM network.

VIII. FRAME FORMAT

It may seem that the packet format of a protocol is a very simple component, but it allows easier understanding of the protocol and provides clues on how the protocol is implemented. Data, fairness, control and idle frames are the four different frame formats defined in the RPR standard. In the following subsections, the four types of RPR frames and their important fields are introduced.

A. Data Frames

Data frames have two formats, which are basic and extended. Extended frame format is aimed at transparent bridging applications allowing an easy way of egress processing and ingress encapsulation of other medium access control (MAC) frames. Using extended frame format also enables RPR rings to eliminate out of ordering and duplication of bridged packets. The Extended frame format is not described in this article.

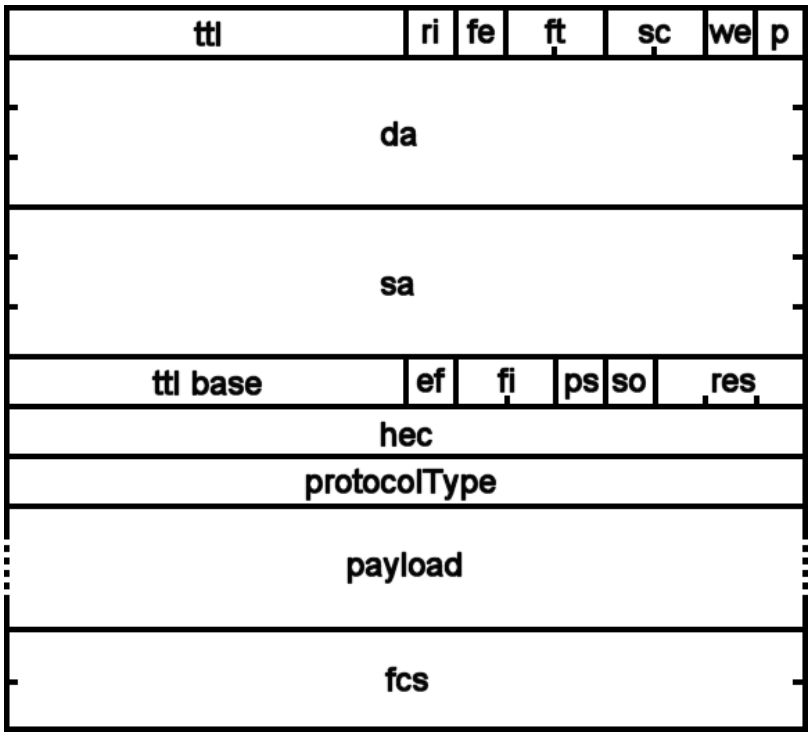


Figure 9. RPR basic data frame format.

Following is a short summary of the functionalities of RPR basic data frame fields:
ttl: The 8 bit "time to live" field is used for detecting packets that should be removed from the ring.
ri: The "ring identifier" bit defines the ring which the packet belongs to (i.e., which ring the packet was inserted into initially).
fe: The "fairness eligible" bit indicates that the packet has to abide by the rules of the fairness algorithm.
ft: The two-bit "frame type" defines the type of the frame.
sc: The two bit "service class" field indicates the class of service (A0, A1, B, C) that the frame is expected to receive.
we: The "wrap eligible" bit defines if the frame can be wrapped at a wrap node.

p: The "parity" bit is reserved for future use in data frames, since the header is being protected by the "*hec*" field.
da: The six-byte "destination address" field defines the destination of a frame.
sa: The six-byte "source address" field shows from which node the frame was sent.
ttl base: This is a fixed field which is set to the initial value of the "*ttl*" field when the packet was initially sourced into the ring. It is used for fast calculation of the number of hops that a packet has traveled.
ef: The "extended frame" bit is set when the frame is an extended frame format.
fi: The two bit "flooding indication" is used to identify if a frame was flooded or not. If the frame was flooded, it also indicates the type of flooding, if the frame is sent on a single ring or on both rings.
ps: The "passed sourced" bit is used as an indication that a frame has passed its source on the opposing ring after a wrap condition. This is useful for stripping packets from the ring under some error conditions.
so: The "strict order" bit enforces the correct order of receive under certain conditions, if there is a chance of receiving a frame with "so" bit set, the frame will be discarded at the receiving station.
res: This is a three-bit reserved field in the header for future expansion.
hec: The two byte "header error correction" field protects the initial 16 bytes of the header.

B. Fairness Frames

The 16-byte fairness frame is a compact control message. It is the communication mechanism of RPR fairness algorithm. The feedback data is the "fairRate" field. The "ffType" (fairness frame format type) field identifies the type of the fairness frame, which can be single-choke or multi-choke message type.

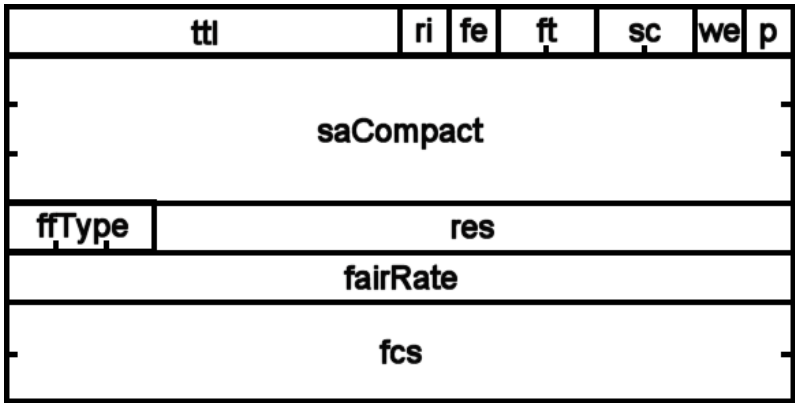


Figure 10. RPR fairness frame format.

C. Control Frames

A control frame has format similar to the data frame but is distinguished by a designated "ft" field value and the type of information carried is specified by its controlType fields. There are different types of control frames in RPR, for example, topology and protection protocol information and OAM (Operations Administration and Maintenance).

D. *Idle Frames*

Idles frames allow strict control of bandwidth allocation around the ring. The frame format is similar to the fairness frame. The “fairnessHeader” and “fairRate” fields are replaced by the four byte “idlePayload”.

IX. CONCLUSION

This paper has discussed and explained the RPR architecture. It has showed how RPR has taken features from earlier ring based protocols, and combined them into a novel and coherent architecture. Important parts that have been covered in this paper include the class based priority scheme, the station design and the fairness algorithm. Performance evaluations using the latest version of the draft standard demonstrate how the protocol behaves using different options. In particular we have demonstrated how the aggressive fairness method is very responsive to change, while the conservative method has a more dampened response under varying load.

RPR is a new MAC-layer technology that may span into the MANs and WANs. RPR can bridged to access networks like EFM, making it possible to perform layer 2 switching far into the backbone network if such large link layer networks turn out to be practical. RPR may also do switching in the backbone network, by letting an RPR ring implement virtual point-to-point links between the routers connected to the stations on the ring.

RPR may differentiate traffic, so when used to implement IP links, it is able to help the IP routers implement the QoS aware communication that is needed in a network that carry multimedia traffic.

REFERENCES

[1] ANSI T1.105.01-2000: Synchronous Optical Network (SONET)- Automatic Protection.

[2] I. Cidon, Y.Ofek, “MetaRing - A Full-Duplex Ring with Fairness and Spatial Reuse”, IEEE Trans on Communications, Vol. 41, No. 1, January 1993.

[3] I. Cidon, L. Georgiadis, R. Guerin, Y. Shavitt: Improved fairness algorithms for rings with spatial reuse. INFOCOM '94. Networking for Global Communications. IEEE, 1994

[4] V. Gambiroza, Y. Liu, P. Yuan, and E. Knightly, “High Performance Fair Bandwidth Allocation for Resilient Packet Rings.”, In Proceedings of the 15th ITC Specialist Seminar on Traffic Engineering and Traffic Management, Wurzburg, Germany, July 2002

[5] E.R. Hafner, Z. Nendal, M. Tschanz, “A Digital Loop Communication System.” , IEEE Transactions on Communications, Volume: 22, Issue: 6, June 1974.

[6] IEEE Draft P802.17, draft 3.0, Resilient Packet Ring, November 2003.

[7] IEEE Standard 802.5–1989, “IEEE standard for token ring”.

[8] IEEE Standard 1596–1990, “IEEE standard for a Scalable Coherent Interface (SCI)”

[9] I. Kessler, A. Krishna: On the cost of fairness in ring networks. IEEE/ACM Transactions on Networking, Vol. 1 No. 3, June 1993

[10] R.M. Needham, A.J. Herbert: The Cambridge Distributed Computing System. Addison-Wesley, London, 1982.

[11] D. Picker, R.D. Fellman: Enhancing SCI’s fairness protocol for increased throughput. IEEE Int. Conf. On Network Protocols. October, 1993.

[12] 1.F.E. Ross, “Overview of FDDI: The Fiber Distributed Data Interface”, IEEE J. on Selected Areas in Communications, Vol. 7, No. 7, September 1989.

[13] ISO/IECJTC1SC6 N7873, “Specification of the ATMR Protocol (V. 2.0)”, January 1993.
[14] W.W. Lemppenau, H.R.van As, H.R.Schindler, “ Prototyping a 2.4 Gbit/s CRMA-II Dual-Ring ATM LAN and MAN”, Proceedings of the 6th IEEE Workshop on Local and Metropolitan Area Networks, 1993.
[15] OPNET Modeler. http://www.opnet.com.
[16] Nortel Networks OPTera Technology, http://www.nortelnetworks.com/products/family/optera.html
[17] Cecil C. Reames and Ming T. Liu, “A Loop Network for Simultaneous Transmission of Variable-length Messages”,.ACM SIGARCH Computer Architecture News , Proceedings of the 2nd annual symposium on Computer architecture, Volume 3 Issue 4, December 1974.Cecil C. Reames and Ming T. Liu, “A Loop Network for Simultaneous Transmission of Variable-length Messages”,.ACM SIGARCH Computer Architecture News , Proceedings of the 2nd annual symposium on Computer architecture, Volume 3 Issue 4, December 1974.
[18] . J.H. Schuringa, G. Remsak, H.R. van As, A. Lila, “Cyclic Queueing Multiple Access (CQMA) for RPR Networks”, 7th European Conference on Networks, & Optical Communications (NOC2002), Darmstadt, Germany, pages 285-292, June 18-21, 2002.
[19] D. Tsiang, G. Suwala, “The Cisco SRP MAC Layer Protocol”, IETF Networking Group, RFC 2892, Aug. 2000
[20] Cisco DPT Technology, http://www.cisco.com/warp/public/779/servpro/solutions/optical/dpt.html

Paper II

Analytical Model of Aggressive Fairness Mode

Fredrik Davik, Amund Kvalbein and Stein Gjessing

Published: In Proceedings of the 40th annual IEEE International Conference on Communications (ICC 2005).

Evaluation: 2,150 papers were submitted to this conference, of which 692 were accepted, divided over nine symposia. Each paper received three or more reviews.

Author Contribution: The problem-idea as well as the derivation of an analytical expression and the running of simulations and associated explanations are all done by Davik.

The contribution of Kvalbein and Gjessing are mainly related to ensure a clear and precise presentation of the main ideas and results obtained.

An Analytical Bound for Convergence of the Resilient Packet Ring Aggressive Mode Fairness Algorithm

Fredrik Davik
Simula Research Laboratory
University of Oslo
Ericsson Research Norway
Email: bjornfd@simula.no

Amund Kvalbein
Simula Research Laboratory
Email: amundk@simula.no

Stein Gjessing
Simula Research Laboratory
Email: steing@simula.no

***Abstract*— Resilient Packet Ring (RPR) is a new standard, designated IEEE standard number 802.17, for MAN and WAN dual ring topologies. RPR uses the buffer insertion principle as a basis for its medium access control protocol. In this paper, we analyze parts of the aggressive mode of the RPR fairness protocol. We look at a congested node, and utilize control systems theory to analyze the stability of the associated fairness algorithm. In particular, we discuss how the settings of the two important parameters ageCoef and lpCoef influence the stability of an RPR-network. At the end of the paper we present simulated scenarios in order to illustrate our results.**

Keywords: Resilient Packet Ring, Fairness, Control Systems, Simulations

I. Introduction

In a ring network, when nodes transmit more data than the ring bandwidth can sustain, we have a resource allocation problem. The solution to the problem is twofold: i) a fair distribution of the bandwidth must be defined, and ii) a policy for this fair distribution of bandwidth must be implemented and enforced.

A recent addition to the protocol family for ring topologies is the IEEE standard for Resilient Packet Rings (RPR), IEEE standard 802.17 [1]. RPR uses the RIAS principle for fair allocation of bandwidth [2]. This means that when a link is congested, the available bandwidth should be fairly (according to the RIAS definition) distributed between all nodes that transmit data over this link.

RPR belongs to the class of ring networks based on the buffer insertion principle [3], [4]. Legacy ring technologies based on the same principle include SCI [5], CRMA-II [6], [7] and MetaRing [8]. Other classes of ring technologies are slotted medium access control protocols (Cambridge Ring [9], ATMR [10]) and token based protocols (IEEE 802.5 Token Ring [11] and FDDI [12]).

RPR's policy for fair division of ring bandwidth is enforced by the fairness algorithm. There are two modes of operation for this algorithm. These two modes are respectively the *conservative* mode of operation, discussed in [13], and the *aggressive* mode of operation, discussed in [13], [14]. The *conservative* mode of operation, uses a form of rate control, where the congested node (i.e. the node immediately upstream of a congested link) issues a rate change command (back-pressure message, also called a fairness message) and then waits to the see the resulting effect. When the estimated waiting period has elapsed, if the resulting effect is not the RIAS fair division of bandwidth, a new rate value is calculated and distributed to upstream nodes. The estimation of the waiting period is based on periodic measurements, and is denoted the *Fairness Round Trip Time* (*FRTT*).

For the *aggressive* mode of operation, a waiting period is not estimated. Instead the congested node periodically sends fairness messages upstream, containing the congested station's estimate of the fair rate. This estimate is calculated based on measurements of the congested node's own send rate and statically configured parameters. These statically configured parameters will typically be set based on heuristic guidelines or expert knowledge/simulation results. Obvious weaknesses with this approach is the risk of mis-configurations that may result in instabilities and/or underutilization of the network.

In this paper, we show analytically how these parameters should be set in a system. By this, we reduce the risk of network instability as well as easing the task of configuring a Resilient Packet Ring network.

In section II we give a short introduction to the RPR fairness algorithm and the vocabulary used in the rest of the paper. In section III, our main analysis of the *aggressive* fairness algorithm is developed. Then, in section IV, we will present and discuss simulation scenarios and associated simulation results we use to verify our analytical results. Finally in sections V and VI, we refer to related work, conclude and give some directions to further work.

II. The RPR fairness algorithm

RPR defines three data packet service classes: high priority (class A), medium priority (class B) and low priority (class C). The high and a configured portion of the medium-priority

traffic is sent using reserved bandwidth[1], while the rest of the medium and the low-priority traffic uses the remaining (unreserved) bandwidth. In a congested situation, the unreserved bandwidth is divided between the contending nodes by the fairness algorithm. Traffic sent using the unreserved bandwidth is referred to as *fairness eligible*, since the fair send rate for this traffic is governed by the fairness algorithm.

An RPR node may contain two separate insertion buffers, known in RPR as transit queues. With such a design, transit traffic of class A is assigned to the Primary Transit Queue (PTQ), while transit traffic of classes B and C use the Secondary Transit Queue (STQ). In this paper we assume that all nodes have two transit queues. The high-level architecture of an RPR node is shown in Fig. 1.

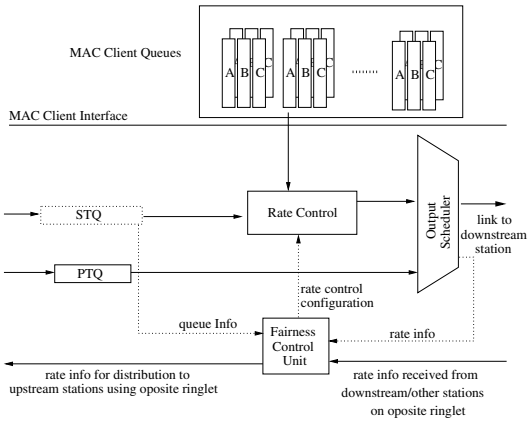


Fig. 1. Generic RPR node design, showing a node's attachment to the ring for transferral of data in the east direction and transferral of fairness messages (rate info) in the west direction. The solid lines indicates the flow of data through the node. The dotted lines indicates the exchange of control/configuration information between node internal function blocks.

When the demand for bandwidth over a link is higher than the available capacity, the link becomes congested. We also say that the node upstream from this link is congested. In a congested node, the STQ is filling up, because the node itself adds traffic at the same time as class B or C traffic is transiting the node. As long as the STQ is only partially filled, the bandwidth on the out-link is equally divided between the local add traffic and the transit traffic, and the node is still in the *uncongested* state. However, when the STQ-occupancy exceeds a threshold, denoted *low*, the node is by definition *congested* [1], and actions (described in section III) will be taken to alleviate congestion. If the STQ continues to fill and the STQ-occupancy exceeds a *high*-threshold, the scheduling rules of RPR causes the local (fairness eligible) add traffic to stop.

A congested node constitutes the *head* of a so called *congestion domain*. The node furthest away, and upstream from the congested node, that contributes to the congestion, is defined as *tail* of the congestion domain. Hence a *congestion domain* contains all nodes that contribute to the congestion of a certain link, including any nodes in between, even if these nodes are not contributing to the congestion. Fig. 2 shows an example of a *congestion domain*.

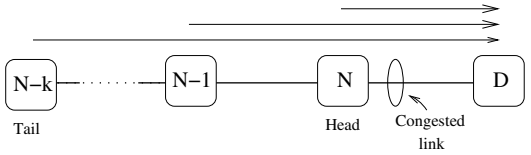


Fig. 2. Nodes N, N-1,..., N-k all send traffic to node D. Thus node N becomes the most congested node in the segment. Node N then becomes head of the *congestion domain* while node N-k, which is the node furthest away, contributing to the congestion, becomes the tail of the congestion domain.

It is the responsibility of the *head* to alleviate congestion by dividing the available bandwidth of the congested link between all contending nodes, so that each gets its fair share of the capacity of the congested link. The *head* does this by calculating a so called *fair rate*, and advertises this rate upstream to all nodes in the *congestion domain*. No node, having received this rate message, is then allowed to send at a rate higher than the (received) fair rate, over the congested link. As mentioned above, there are two versions of this fairness algorithm. In this paper we analyze the so called *aggressive* version.

III. Aggressive Mode Rate Control Configuration

In general, in feedback control systems not using predictive methods, a rule of thumb for design of the control system is that the time constant for the controller part of the system, should not be faster than the time constant of the system itself. Thus, given the time constant of the controller, this introduces a constraint with respect to which systems can be safely controlled. In an RPR-network, the time constant of the system (a *congestion domain* with its associated nodes) will change dynamically, depending on the load situation on the ring. The time-constant of the system consists of a fixed part and a dynamic part.

The fixed part is the propagation delay from the *head* of the *congestion domain* to the *tail* and back. Also, the per node processing of fairness messages can be considered fixed. However, the processing delay, is so small that it has no significance for the scenarios considered in this paper.

The dynamic part consists of two sub-parts: i) the queuing delay, experienced by a fairness message, when propagating from the *head* to the *tail*, and ii) the time required to empty the transit queues in the path between *tail* and the *head* before the rate change at the *tail* is observable at the *head.*

In this section, we show how the setting of two parameters, named $ageCoef$ and $lpCoef$, relates to the steady-state and transient-response behavior of the rate control algorithm for a *congestion domain head.*

On an RPR-network, the *head* of a *congestion domain* calculates the fair division of send rates for traffic traversing

[1]Actually, some of the bandwidth called reserved in this paper may dynamically be allocated to fairness eligible traffic. This simplification, however, does not influence the results presented.

the congested link. The calculation is done at periodic intervals (every aging interval). According to the standard, the aging interval is 100μs (for line rates $\geq$ 622Mbit/s). In the RPR standard, the calculation of the rate is specified as two cascaded low-pass filters, also referred to as rate-counters. The first filter, takes the weighted sum of locally added traffic, denoted $x(n)$, during the current aging-interval, n, and the previous filter output-value, $addRate(n-1)$. The resulting sum is the new output-value of the filter, $addRate(n)$. This value is then fed into the 2^{nd} low-pass filter, which takes the weighted sum of its input, $addRate(n)$, and the previous output-value of the filter, $lpAddRate(n-1)$. The resulting sum is the new output-value of the filter, $lpAddRate(n)$.

The calculated value, $lpAddRate(n)$, becomes the *head*'s new estimate of the fair rate, and is distributed in the form of a rate change command, in RPR known as a *fairness message*, to the upstream neighbors. The filtering process described above can be formulated in terms of a discrete-time 2^{nd}-order low-pass filter, with a sampling period which equals the aging interval. The filtering processes is described formally below.

Let the amount of locally added traffic during aging interval n be denoted $x(n)$, and let:

$$p_1 = \frac{ageCoef-1}{ageCoef}, \text{ where} \quad ageCoef \in \{1,2,4,8,16\} \tag{1}$$

Then, from the standard, we have:

$$addRate(n) = addRate(n-1) \cdot p_1 + x(n) \tag{2}$$

$$lpAddRate(n) = \frac{1}{lpCoef}\left(lpAddRate(n-1) \cdot (lpCoef-1) + addRate(n)\right) \tag{3}$$

Where: $lpCoef \in \{16, 32, 64, 128, 256, 512\}$

This can be modeled as a 2^{nd} order digital low-pass filter as shown in Fig. 3. In the figure, the box with the marking z^{-1} denotes that the value on the output of the box is delayed one sampling period (i.e. one aging interval) as compared to the value on the input of the box. The filter input and output-values, denoted respectively $X(z)$ and $Y(z)$ is the Z-domain representations of the discrete time-domain signals $x(n)$ and $y(n)$, where $y(n) = lpAddRate(n)$.

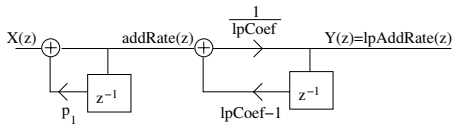


Fig. 3. Block Diagram of the two Cascaded First-Order Low-Pass Filters Yielding the Fair Rate Based on the Congestion Head's Own Send Rate $x(n)$

Let:

$$p_2 = \frac{lpCoef-1}{lpCoef} \tag{4}$$

Then the transfer function, $H(z)$, of the 2^{nd} order low-pass filter shown in Fig. 3 can be written as shown in (5) below.

$$H(z) = \frac{Y(z)}{X(z)} = \frac{1}{lpCoef} \cdot \frac{z^2}{(z-p_1)(z-p_2)} \tag{5}$$

Given this model of the *aggressive* fairness algorithm's rate control mechanism, using discrete control systems' analysis techniques, we can deduce some of its basic properties. For the purpose of this paper, the most important system properties are the transient response and stability. In this paper, we use a simplistic approach, where we from the properties of the rate control part of the system, i.e. the 2^{nd} order low-pass filter, deduce some properties, and then we utilize the findings to determine some boundary operating conditions for a system. Specifically, we determine the time-constant of the filter when applied to a step input and use this time-constant to determine the boundary conditions of the system.

If a node, transiting traffic from upstream nodes, has sent no traffic on the ring lately, the value of its $lpAddRate$ rate-counter will be equal or close to zero. Assume that the node is currently uncongested. Also assume that both the aggregate demand of upstream nodes and the local demand for bandwidth, over the link connected to the node's downstream neighbor, exceed half of the available bandwidth. Then the scheduling rules of RPR will result in equal division of bandwidth between local and transit traffic. With the sum of demands being greater than the available bandwidth, this will result in the secondary transit queue (STQ) starting to fill. As long as the STQ-occupancy stays below the high threshold, the local add-rate, denoted $x(n)$, will be constant and equal to half of the available bandwidth. Once the STQ-occupancy exceeds the high threshold, the scheduling rules of RPR causes the local add-rate, $x(n)$, to be 0, to avoid overflow of the STQ. The value of $x(n)$ will remain 0 until the STQ-occupancy falls below the *high* threshold.

The scheduling rules described above, are summarized in (6) below:

$$x(n) = \begin{cases} bandwidth & \text{, STQ-occupancy} = 0 \\ \frac{bandwidth}{2} & \text{, } 0 < \text{STQ-occupancy} < \textit{high} \\ 0 & \text{, otherwise} \end{cases} \tag{6}$$

These scheduling rules, controlling the value of the node's local add-rate, $x(n)$, together with the configuration of the 2^{nd} order low-pass filter shown in Fig. 3, control the behavior of the $lpAddRate(n)$ rate-counter in the *congestion domain head*. Now, let us consider the scenario where a node's $lpAddRate = 0$ and the STQ-occupancy is close or equal to 0. The node is currently transiting data from upstream nodes at the available bandwidth of the link connected to the downstream neighbor. Next, the node starts sending local (add) traffic over the same link, which at some point in the near future will make the node *head* of a *congestion domain*. From the time where the node starts transmitting and onwards, the behavior of its $lpAddRate$ rate-counter is characterized by a set of cycles. A cycle consist of two consecutive periods. In the first period, $x(n) = \frac{bandwidth}{2}$, and thus $lpAddRate(n)$ monotonically increases until the STQ-occupancy exceeds the *high* threshold. At that point, the second period starts. In the second period, $x(n) = 0$, and thus $lpAddRate(n)$ starts monotonically decreasing. The second period ends when STQ-occupancy falls below the *high* threshold. Thus the cycle is

concluded.

The behavior of the rate control algorithm for a *congestion domain head*, resembles that of the step response of a second order continuous-time system. In (7) below, $G(s)$ is the Laplace transform of the general 2^{nd}-order continuous-time system function, $g(t)$:

$$G(s) = \mathcal{L}(g(t)) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{7}$$

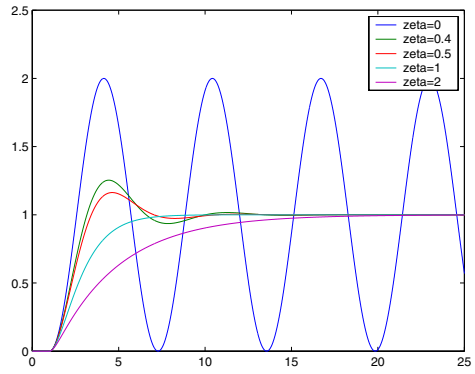


Fig. 4. Unit step response for 2nd order continuous-time system.

For an unstable second order continuous-time system, the system output value, in response to a step input, represented in Fig. 4 above, with the plot where $\zeta = 0$, oscillates between the max and min values determined by the system's inherent boundaries, with a frequency as determined by the system's natural frequency, ω_n. For a corresponding stable system, represented by the plots in Fig. 4 where $\zeta \in \{0.4, 0.5, 1, 2\}$, the system output in response to the same input, converges to the same steady-state value. The speed of convergence and the presence and magnitude of oscillations during convergence however, depends on the setting of ζ.

Given the cyclic behavior of $lpAddRate(n)$, the stability of the *aggressive* mode fairness algorithm, can be studied in terms of the step-response of the rate-control (fairness) algorithm executing in the head-node. To determine whether an RPR-network will stabilize or not, it is sufficient to analyze the first period of a cycle (described above) that will result from a given configuration of the $ageCoef$ and $lpCoef$ parameters. The analysis consists of determining the time-constant, τ, for a given configuration. Once this time-constant has been decided, this represents an upper bound for the time-constant of a system to be safely controlled using this configuration. Below, an analytical expression for the value of τ is derived.

As discussed above, it is safe to assume that the input to the system, $x(n)$, during the first half of the initial cycle, can be described in terms of a step function:

$$x(n) = \begin{cases} 0, & n < 0 \\ \Delta, & \text{otherwise, where } \Delta = \frac{bandwidth}{2} \end{cases} \tag{8}$$

Then, the corresponding Z-transform, $X(z)$, of the input signal, $x(n)$, is:

$$X(z) = Z\{x(n)\} = \frac{\Delta \cdot z}{z - 1} \tag{9}$$

If we evaluate the output of the system, $Y(z)$, in response to the input, $X(z)$, we have (from (5) and (9)) the relationship:

$$Y(z) = \frac{\Delta}{lpCoef} \cdot \frac{z^3}{(z-1)(z-p_1)(z-p_2)} \tag{10}$$

If we perform a partial fraction expansion on (10) above before performing the inverse Z-transform on the signal $Y(z)$, we get the (discrete) time domain representation of the output signal:

$$y(n) = Z^{-1}\{Y(z)\} = \tag{11}$$
$$\frac{\Delta}{lpCoef} \cdot \left(\frac{1}{(1-p_1)(1-p_2)} + \frac{p_1^{n+2}}{(p_1-1)(p_1-p_2)} + \frac{p_2^{n+2}}{(p_2-1)(p_2-p_1)} \right)$$

This function resembles an exponential ramp function, consisting of a constant part and two parallel first order-filter functions with a steady-state value:

$$y_{ss} = \lim_{n\to\infty}(y_n) = \frac{\Delta}{lpCoef \cdot (1-p_1)(1-p_2)} = ageCoef \cdot \Delta \tag{12}$$

As the ratio $\frac{p_2}{p_1}$, from the configuration of parameters $ageCoef$ and $lpCoef$, becomes larger than 1, the transient behavior, and thus the function's time-constant, is determined more by the last fraction, denoted $y_2(n)$, where:

$$y_2(n) = \frac{\Delta}{lpCoef} \cdot \frac{p_2^{n+2}}{(p_2-1)(p_2-p_1)} \tag{13}$$

Thus in these cases, it is sufficient to study $y_2(n)$ to determine the transient response of the 2^{nd} order filter. $y_2(n)$ together with the first fraction from (11), represents a first order filter system. Given that the input to the filter, $x(n)$, is constant during the observation interval, the value of $y_2(n)$, evaluated at discrete points $n \in \{0, 1, 2, ..., N\}$, will match exactly the corresponding points on a continuous-time filter with the same output-function as shown in (13). In the case of the continuous-time filter, however, the output-values is defined for all values of $t \geq 0$ (not only at discrete times, as is the case of $y(n)$). To find the time-constant of the digital filter, we analyze the filter in terms of output-function for an equivalent continuous-time filter-function (we replace the discrete variable n in (11) with the continuous variable t).

For a first order continuous-time (filter) system, the system time-constant, τ, in response to a step function applied to the system input, is defined as shown in (14) [15].

$$\frac{y_{2_{ss}}}{\tau} = \frac{d}{dt}\left(y_2(t)\right)|_{t=0} \tag{14}$$
$$= \frac{d}{dt}\left(\frac{\Delta}{lpCoef} \cdot \frac{p_2^{t+2}}{(p_2-1)(p_2-p_1)} \right)\Bigg|_{t=0}$$
$$= \frac{\Delta}{lpCoef} \cdot \frac{ln(p_2) \cdot p_2^2}{(p_2-1)(p_2-p_1)}$$

$$\Rightarrow \tau = \left(\frac{y_{2_{ss}} \cdot lpCoef(p_2-1)(p_2-p_1)}{\Delta \cdot ln(p_2) \cdot p_2^2} \right) \tag{15}$$

In (14), $y_{2_{ss}}$ represents the steady-state value of y_2 as $t \to \infty$. In the case of an exponential increasing function with a non-zero start value ($y_2(0) \neq 0$), the $y_{2_{ss}}$ factor in the numerator on the left hand side can be replaced with the expression $y_{2_{ss}} - y_2(0)$. Evaluating y_2 at 0 and ∞ gives:

$$y_{2_{ss}} = lim_{t\to\infty}(y_2(t)) = 0 \quad (16)$$

$$y_2(0) = \frac{\Delta}{lpCoef} \cdot \frac{p_2^2}{(p_2-1)(p_2-p_1)} \quad (17)$$

Substituting $(y_{2_{ss}} - y_2(0))$ for $y_{2_{22}}$ in (14), we get:

$$\tau = \frac{-1}{ln(p_2)} = \frac{-1}{ln(\frac{lpCoef-1}{lpCoef})} \approx lpCoef \quad (18)$$

The final simplification is obtained when using a power-series expansion of the $ln(\frac{lpCoef-1}{lpCoef})$ expression in the denominator.

Note that in (18) above, the time-constant is specified in units of t, which for the discrete filter corresponds to the number of sampling-periods. As noted at the start of the chapter, the sampling period equals one aging interval.

As an example, given a system with $ageCoef = 4$ and $lpCoef = 64$, the resulting value of τ from (18) above becomes $\tau \approx 64$ [aging intervals] $= 6.4ms$. This means that in a scenario with a node connected upstream of a congested link, having the configuration shown above, the maximum system time-constant (as discussed at the start of this chapter) equals 6.4ms. Failure to operate within these boundary conditions may result in an unstable system, i.e. the system fails to converge to the fair division of send rates over the congested link.

In an RPR-network, when the time-constant of the rate-controlling (fairness) algorithm executing in the *head* node is too fast compared to the system's time-constant, the system will be unstable. The symptoms of an unstable RPR-network, like that shown in Fig. 6 b), resembles those of a 2^{nd} order continuous-time system shown in Fig. 4 above with $\zeta = 0$. In the case of an unstable RPR-network, the output of the contending stations will oscillate between a max and a min value, where the max value is limited to the value of the available bandwidth and the min value is limited to 0. When the time-constant of the rate controlling (fairness) algorithm equals or exceeds the system's time-constant, the system will be stable. When this is the case, the throughput of the contending stations converges to the (RIAS) fair division of bandwidth. This behavior is shown in Fig. 6 a). Depending on the ζ "equivalent" setting of the RPR-network, the speed of convergence and presence and magnitude of oscillations is decided.

In this chapter, we have discussed how the setting of the two parameters, named $ageCoef$ and $lpCoef$, relate to the steady-state and transient-response behavior of the fairness algorithm for a *congestion domain head*, executing the *aggressive* mode of the RPR fairness algorithm. In the next chapter, we will illustrate the findings using our RPR simulator model implemented within the OPNET Modeler framework [16].

IV. Simulation Results

In this section, we describe the simulation scenario used to illustrate and evaluate the analytical results described above.

For the scenario used, often referred to as a "hot"-receiver scenario, we aim to show the relation between the link length (propagation delay) and the system settling time. With system settling time, we mean the time used from congestion occurs, until a fair division of send rates has been established by the fairness algorithm. For an unstable configuration, the settling time will be infinite, as the fairness algorithm never converges to the fair division of send rates.

Our simulation scenario is illustrated in Fig. 5. Node 4 is the "hot" receiver, receiving traffic from nodes 0 - 3. These nodes all try to send at their maximum allowed rate, making the link between nodes 3 and 4 the most congested link. We use a fixed packet size of 500B, and the STQ buffer thresholds *full*, *high* and *low* are set to: 254400, 63625 and 31812 bytes.

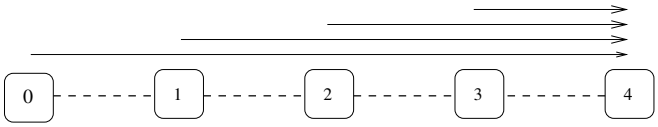
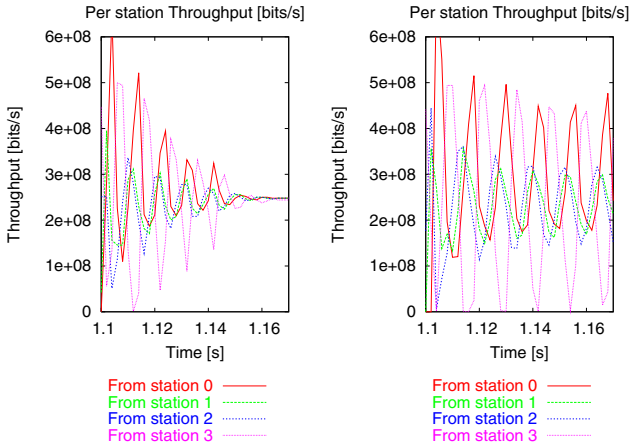


Fig. 5. In this hot receiver scenario, nodes 0, 1, 2 and 3 send at their maximum allowed rate to the "hot" receiver – node 4.

A. Fairness Convergence as a Function of Link Delay

In this simulation experiment, we want to illustrate the similarity between the system response of an RPR-network and the step-response of a second-order continuous time system with various values for the ζ variable shown in (7) above. We show two sets of output-curves, one (Fig. 6 a)) where the aggregate link-delay (system time-constant) is within the stability bounds of the RPR-network and thus, the RPR-network converges to the fair division of link bandwidth. The other set (Fig. 6 b)) shows the throughput-curves for a scenario where the system time-constant exceeds the stability bounds of the RPR-network. Correspondingly, for this scenario, the RPR-network does not converge to the fair division of link bandwidth. For both sets, $lpCoef = 64$ and $ageCoef = 4$.

As noted above, Fig. 6 a) shows a scenario where the configuration of the $lpCoef$ parameter is set to a value causing



a) link-delay is $500\mu s$ b) link-delay is $750\mu s$

Fig. 6. Throughput of traffic received by station 4 in hot receiver scenario.

the fairness algorithm to converge to the fair division of send rates. The link-delay in this scenario is 500μs, thus the aggregate link-propagation delay (from node 3 to node 0 and back) is 3ms. The corresponding analytical system time-constant is given by (18) and gives a bound of 64 aging intervals (i.e. 6.4 ms). When we compare the throughput convergence of the RPR-network with the convergence of the continuous-time system shown in Fig. 4 we can observe the similarity. For a continuous-time system, with a value of $\zeta < 1$, the convergence to the steady-state value is characterized by exponential (decreasing) oscillations. The lower the value of ζ, the higher the oscillations will be. Finally, when ζ becomes zero, the system will never converge and the oscillations are bounded by the physical boundaries of the system.

Similarly, for our RPR-network, if the time-constant, τ, of the rate-controller, is large compared to the system time-constant, the convergence towards fair rate will proceed slowly and surely without oscillations. If τ is in the (lower) proximity of the time-constant of the RPR-network (as the one showed in Fig. 6 a)), the RPR-network converges to the fair division of rates with exponential decreasing oscillations. Finally, as illustrated in Fig. 6 b), when the link delay is increased to 750μs, the value of τ clearly becomes too low compared to the system time-constant. As shown, for this configuration, the throughput of the contending nodes experience oscillations that are approximately constant in amplitude and the system does not converge to a steady-state (fair rate) value.

Given the analytical results presented in section III, the observant reader may find the behavior plotted in Fig. 6 b) unexpected. The analytical results indicate that given a configuration where $lpCoef$ is set to 64, the rate-controller should be able to handle networks where the system-delay is bounded upwards to 6.4ms. In the case of 750μs link-delays, the aggregate propagation delay from node 3 to node 0 and back is 4.5ms. Additionally fairness messages may incur more than 100μs processing delay per node traversed, when propagating from the *head* to the *tail*, resulting in a delay close to 5ms. Finally, queueing and scheduling delays experienced in both the upstream path (experienced by rate messages sent from *head* to *tail*) and the downstream path adds to the total delay. Thus in total, the system time-constant becomes too large compared to the value of τ.

B. Fairness Convergence as a Function of Total Link Delay

By varying the link length of each link, we can control the aggregate propagation delay from the *head* to the *tail* and back in the *congestion domain*. For each of the allowed values of $lpCoef$ (shown in (3)), while keeping the value of $ageCoef$ constant at the RPR default setting of 4, we run a sequence of simulations, increasing the link-delay for each iteration. The obtained throughput-data are subsequently processed to find the time it takes for the fairness algorithm to reach a stable state. Based on [15], we define a state as stable when no observed values deviate more than 5% from the mean value over a 50 ms time interval. Formally, we define our system to be stable at time t_0, if all values sampled in the interval $\langle t_0, t_0 + t_s \rangle$ ($t_s = 50ms$) are within $\pm 5\%$ of the mean value in the same interval:

$$\begin{aligned} max(X(t)) &< \overline{X(t)} * 1.05, \quad t_0 < t < t_0 + t_s \\ min(X(t)) &> \overline{X(t)} * 0.95, \quad t_0 < t < t_0 + t_s \end{aligned} \tag{19}$$

We use the "hot"-receiver scenario described above, and measure the time it takes from a link (the link between nodes 3 and 4 in Fig. 5 above) becomes congested, to the system has stabilized at the fair division of sending rate for each source.

For each allowed value of the *lpCoef* parameter, we plot the convergence-time required to reach the stable state as defined in (19) above. The variable in the plot is the aggregate of link propagation-delays between the *head* and *tail* node and back. The results are shown in Fig. 7 below. The abscissa value when the stabilization time becomes infinite is the point where, for the corresponding system time-constant and $lpCoef$ setting, the system no longer satisfies the stability criterion of (19) above.

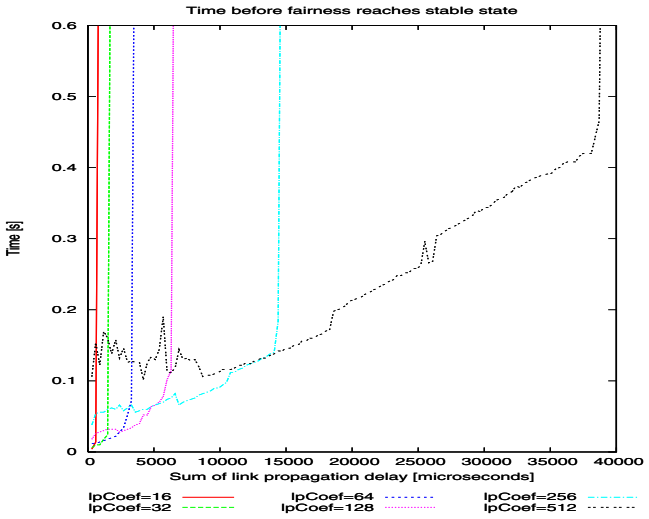


Fig. 7. The time it takes before the fairness algorithm stabilizes at the fair rate for each node sending over a congested link for varying link lengths and *lpCoef* settings.

As seen in the figure, the point where the system (for the various $lpCoef$) settings no longer converges are lower than those of the analytical expression in (18) ($lpCoef$=16: 1.6ms, $lpCoef$=32: 3.2ms, $lpCoef$=64: 6.4ms, $lpCoef$=128: 12.8ms, $lpCoef$=256: 25.6ms, $lpCoef$=512: 51.2ms). As discussed for the previous simulation experiment, we believe that the major cause of this is the scheduling and queuing delays on the path between *head* and *tail*.

V. Related Work

Analytical performance evaluation of a complex system like an RPR-network is extremely hard to perform precisely. Several papers are published for buffer insertion rings presenting analytical performance evaluation based on queueing theory

[17], [18], [19]. Similarly, we can find analytical studies of RPR based on queueing theory [20]. Other relevant work in the area of buffer insertion rings is the work presented in [21]. In this paper, Scott et al. studied the performance of the buffer insertion ring SCI analytically, but did not include the Fairness protocol. For RPR in general, several papers have been published, studying different performance aspects. In [22], Huang et al. present a thorough analysis of ring access delays for RPR-networks with nodes that contain a single transit queue. In [2], Gambiroza et al. focus on the operation of the RPR fairness algorithm and their alternative proposal, DVSR, and its ability, for some given load scenarios to converge at the fair division of rates according to their RIAS fairness reference model. More general fairness in insertion buffer rings have been studied by several [23], [24], [25], [26].

VI. Conclusion

When the demand for bandwidth in a Resilient Packet Ring is higher than the supply, the fairness algorithm running in the most congested node (the *head* of a *congestion domain*) is responsible for calculating the fair rate that all nodes sending over the congested link, must adhere to. In this paper we have developed a discrete control system model of RPR's *aggressive* mode fairness algorithm rate control mechanism running in *congestion domain* heads. We are not aware of any other contribution that evaluate the performance of RPR-networks using control systems theoretic approaches. We have discussed the model and deduced some of its basic properties. We have argued that the time-constant, τ, of the rate-controller must not be larger than the time-constant of the system it controls, which consists of all nodes contributing to the congestion (the *congestion domain*). Based on this we have shown that the setting of some important system parameters, $ageCoef$ and $lpCoef$, can be analytically estimated based on the size of the controlled system (the size of the *congestion domain*). This new insight makes the configuration of an RPR-network sounder, and less prone to configuration errors. Finally, we have supported our analytical findings by simulations that indicate that our analytical model yields a reasonable upper bound for the convergence of the Resilient Packet Ring *aggressive* mode fairness algorithm.

In future work, it would be interesting to extend our control system model to include all the nodes that contribute to the congestion. It would also be interesting to model the fairness algorithm even more precisely, in order to find even closer correspondence between the model and the simulated results.

VII. Acknowledgements

We want to thank Nils Damm Christophersen for providing valuable insights to the analytical section of this paper.

References

[1] IEEE Computer Society, "IEEE Std 802.17-2004," September 24 2004.

[2] V. Gambiroza, P. Yuan, L. Balzano, Y. Liu, S. Sheafor, and E. Knightly, "Design, analysis, and implementation of DVSR: a fair high-performance protocol for packet rings," *IEEE/ACM Trans. Networking*, vol. 12, no. 1, pp. 85–102, 2004.

[3] E. Hafner, Z. Nendal, and M. Tschanz, "A Digital Loop Communication System," *IEEE Trans. Commun.*, vol. 22, no. 6, pp. 877 – 881, June 1974.

[4] C. C. Reames and M. T. Liu, "A Loop Network for Simultaneous Transmission of Variable-length Messages," in *Proceedings of the 2nd Annual Symposium on Computer Architecture*, vol. 3, December 1974.

[5] "IEEE Standard 1596-1990," IEEE Standard for Scalable Coherent Interface (SCI).

[6] H. R. van As, W. W. Lemppenau, P. Zafiropulo, and E. Zurfluh, "CRMA-II: A Gbit/s MAC Protocol for Ring and Bus Networks with Immediate Access Capability," in *Proceedings of the Ninth Annual European Fibre Optic and Local Area Networks Conference (EFOC/LAN'91)*, London, June 1991, pp. 262 – 277.

[7] W. W. Lemppenau, H. R. van As, and H. R. Schindler, "Prototyping a 2.4 Gbit/s CRMA-II dual-ring ATM LAN and MAN," in *Proceedings of the 6th IEEE Workshop on Local and Metropolitan Area Networks*, 1993, pp. 17 – 18.

[8] I. Cidon and Y. Ofek, "MetaRing - A Full Duplex Ring with Fairness and Spatial Reuse," *IEEE Trans. Commun.*, vol. 41, no. 1, pp. 110 – 120, January 1993.

[9] R. Needham and A. Herbert, *The Cambridge Distributed Computing System*. London: Addison-Wesley, 1982.

[10] "ISO/IECJTC1SC6 N7873," January 1993, specification of the ATMR Protocol (V.2.0).

[11] "IEEE Standard 802.5-1989," IEEE Standard for Token Ring.

[12] F. E. Ross, "An Overview of FDDI: the Fiber Distributed Data Interface," *IEEE J. Select. Areas Commun.*, vol. 7, no. 7, pp. 1043 – 1051, September 1989.

[13] F. Davik, M. Yilmaz, S. Gjessing, and N. Uzun, "IEEE 802.17 Resilient Packet Ring Tutorial," *IEEE Commun. Mag.*, vol. 42, no. 3, pp. 112–118, March 2004.

[14] F. Davik and S. Gjessing, "The Stability of the Resilient Packet Ring Aggressive Fairness Algorithm," in *Proceedings of the 13th IEEE Workshop on Local and Metropolitan Area Networks (LANMAN2004)*, Mill Valley, CA, April 25-28 2004, pp. 17–22.

[15] C. L. Phillips and R. D. Harbor, *Feedback Control Systems*, 2nd ed. Prentice-Hall, 1991.

[16] "OPNET Modeler. http://www.opnet.com." [Online]. Available: http://www.opnet.com/

[17] M. T. Liu, G. Babic, and R. Pardo, "Traffic analysis of the distributed loop computer network (DLCN)," in *Proc. Nat. Telecommun. Conf.*, December 1997.

[18] A. Thomasian and H. Kanakia, "Performance Study of loop networks using buffer insertion," *Comput. Networks*, vol. 3, pp. 419–425, December 1979.

[19] W. Bux and M. Schlatter, "An Approximate Method for the Performace Analysis of Buffer Insertion Rings," *IEEE Trans. Commun.*, vol. 1, January 1983.

[20] P. Yue, Z. Liu, and J. Liu, "High performance fair bandwidth allocation algorithm for resilient packet ring," in *Proceedings of the 17th International Conference on Advanced Information Networking and Applications*, March 27-29 2003, pp. 415 – 420.

[21] S. L. Scott, J. R. Goodman, and M. K. Vernon, "Analysis of the SCI Ring," University of Wisconsin, Madison, Tech. Rep. 1055, November 1991.

[22] C. Huang, H. Peng, F. Yuan, and J. Hawkins, "A steady state bound for resilient packet rings," in *Global Telecommunications Conference, (GLOBECOM '03)*, vol. 7. IEEE, December 2003, pp. 4054–4058.

[23] I. Cidon, L. Georgiadis, R. Guerin, and Y. Shavitt, "Improved fairness algorithms for rings with spatial reuse," *IEEE/ACM Trans. Networking*, vol. 5, no. 2, pp. 190–204, 1997.

[24] I. Kessler and A. Krishna, "On the cost of fairness in ring networks," *IEEE/ACM Trans. Networking*, vol. 1, no. 3, pp. 306–313, 1993.

[25] D. Picker and R. Fellman, "Enhancing SCI's fairness protocol for increased throughput," in *IEEE Int. Conf. On Network Protocols*, October 1993.

[26] J. Schuringa, G. Remsak, and H. R. van As, "Cyclic Queuing Multiple Access (CQMA) for RPR Networks," in *Proceedings of the 7th European Conference on Networks & Optical Communications (NOC2002)*, Darmstadt, Germany, June 2002, pp. 285 – 292.

Paper III

Improvement of Resilient Packet Ring Fairness

Fredrik Davik and Stein Gjessing

Published: The most important results of this paper (technical report) have have been accepted for publication in Proceedings of the 48th annual IEEE Global Telecommunications Conference (GLOBECOM 2005). The accepted paper also includes the most important findings from paper IV.

In the combined conference version, the main problems addressed by Paper III and Paper IV are discussed, along with proposed solutions and a short presentation of their performance.

Evaluation: We do not have the conference statistics from 2005. In 2004 however, 2086 papers were submitted to the conference, of which 792 were accepted, divided over 7 symposia. The conference version was reviewed by four reviewers.

Author Contribution: All proposed modifications and their associated discussions and simulations are done by Davik.

The contribution from Gjessing is mainly related to the assurance of a clear and precise presentation of the main ideas and results.

Improvement of Resilient Packet Ring Fairness

February 2005
Revised August 2005

Fredrik Davik
Simula Research Laboratory
University of Oslo
Ericsson Research Norway
Email: bjornfd@simula.no

Stein Gjessing
Simula Research Laboratory
Email: steing@simula.no

Abstract— **Resilient Packet Ring (RPR) is a recent networking standard developed by the IEEE LAN/MAN working group. RPR is an insertion buffer, dual ring technology, utilizing a back pressure based fairness algorithm to distribute bandwidth when congestion occurs. The fairness algorithm has two modes of operation, called respectively the *aggressive* and the *conservative* fairness modes. For some scenarios, the *aggressive* fairness mode suffers from severe performance deficiencies.**

In this paper, we propose two novel contributions. The first is a measurement method which enables a node to determine its operating context. The second contribution is a fair rate calculation method, termed the *moderate* fairness mode, which resolves the *aggressive* mode performance deficiencies while retaining several other properties provided by the *aggressive* mode fairness.

We compare the performance of the *moderate* fairness mode to that of the *aggressive* and the *conservative* modes by simulations, and find that for some scenarios the *moderate* mode outperforms the *aggressive* and the *conservative* modes. For some other scenarios, the convergence time of the *moderate* mode is somewhat longer than that of the *aggressive* mode.

Keywords: Resilient Packet Ring, Fairness, Performance evaluation, Simulations, Next generation protocol design and evaluation, Communications modeling, Next Generation Networks Principles, High-speed Networks.

I. Introduction and motivation

Resilient Packet Ring (RPR) is a new networking standard developed by the IEEE 802 LAN/MAN Standards Committee, assigned standard number IEEE 802.17-2004 [1], [2]. Although RPR was developed by the LAN/MAN committee, it is designed mainly to be a standard for metropolitan and wide area networks.

RPR is a ring topology network. By the use of two rings (also called ringlets), resilience is ensured; if one link fails, any two nodes connected to the ring still have a viable communication path between them. When a node wants to send a packet to another node on the ring, it adds (sends) the packet onto one of the two ringlets. For bandwidth efficiency, the ringlet that gives the shortest path is used by default, but a sender can override this (on a per packet basis) if it for some reason has a ringlet preference. When the packet travels on the ring, it *transits* all nodes between the sender and the receiver. When it reaches the destination, the packet is removed (stripped) from the ring. Hence the bandwidth that would otherwise be consumed by the packet on its way back to the sender (as is the case in the Token Ring [3]), can be used by other communications. Such destination stripping of packets leads to what is commonly known as *spatial reuse*.

RPR uses *insertion buffer(s)* for collision avoidance [4], [5]. When a packet in transit arrives at a node that is currently adding a packet to the ring, the transiting packet is temporarily stored in an insertion buffer, called a *transit queue* in RPR. In order to get some flexibility in the scheduling of link bandwidth resources between add- and transit traffic, the transit queues may be in the order of hundreds of kilobytes large.

In a buffer insertion ring like RPR, a *fairness algorithm* is needed in order to divide the bandwidth fairly[1] between contending nodes, when congestion occurs [6], [7].

The RPR fairness algorithm runs in one of two modes, termed respectively the *conservative* and the *aggressive* modes. The *aggressive* mode of operation is simpler (i.e. requires less code and configuration) than the *conservative* mode, and is used by e.g. Cisco Systems. In this paper, the main focus is on some of the performance deficiencies found in the *aggressive* fairness mode. We do however evaluate the performance of the *conservative* mode fairness for some selected scenarios.

The feedback control system nature of the RPR fairness algorithm makes the amount of add traffic from each sending node oscillate during the transient phase where the feedback control system tries to adjust to a new load [8]. Several papers have reported that in some cases the oscillations decreases and (under a stable traffic pattern) converges to a fair distribution of add rates, while under other conditions, the algorithm diverges, and oscillations continues [8]–[11].

The main contribution of this paper is twofold. First, we propose a novel context determination function, to enable a congested node to determine whether it is utilizing its fair share of bandwidth or not. Secondly, we propose an alternative fair rate calculation mode, termed the *moderate* fairness mode, with the goal of improving on the properties of the *aggressive*

[1] RPR nodes may have different weights, so a fair division does not have to be an equal one. In this paper, however, we assume all nodes have the same weight.

mode fairness. The proposed fairness mode, is designed to fit within the framework given by the current RPR standard.

To evaluate the properties of the three fairness modes, we use performance evaluation by simulations. We have implemented the RPR standard within the the J-Sim and OPNET modeler discrete event simulator frameworks [12], [13]. For this paper however, we use our OPNET implementation.

The rest of this paper is organized as follows: In section II, we present a brief introduction to the RPR fairness algorithm. Then, in section III, we present a class of scenarios where a so-called *modest* head node in a congestion domain induces permanent oscillations in the throughput experienced by the head's upstream neighbors [14]. Then in section IV, we discuss some possible solutions to these problems. Then in sections V and VI, we present our novel contributions handling these problems. In section VII, we evaluate our contributions by simulations and compare the performance to that of the original RPR fairness modes. Finally in sections VIII, IX and X, we present related work, conclude and present possible directions for further works.

II. The RPR Fairness Algorithm

When several sending nodes try to send over a congested link concurrently, the objective of the RPR fairness algorithm is to divide the available bandwidth fairly between the contending nodes. RPR has three traffic classes: high, medium and low priority. Bandwidth for high and medium traffic is pre-allocated, so the fairness algorithm distributes bandwidth to low priority traffic only. In this paper all data traffic is low priority.

The fairness algorithm is a closed-loop control system [8]. The goal of the fairness algorithm is to arrive at the "Ring Ingress Aggregated with Spatial Reuse" (RIAS) fair division of rates over the congested link [9]. For a congested link, over which each active node has an infinite demand and all nodes have equal weights, the RIAS fair bandwidth will be the capacity of the link divided by the number of active nodes. Fairness for ring segments having active nodes with different weights is not covered by the RIAS reference model. In this paper, all nodes have equal weights, thus we can use the RIAS reference model.

The control system encompasses all nodes that send over the same congested link, known in RPR as a *congestion domain* [15]. The node directly upstream of the most congested link is called the *head* of the congestion domain. The node in the congestion domain that is furthest away from the head is called the *tail* of the congestion domain. Later in this paper we are going to use a scenario depicted in figure 1. Here nodes 0, 1, 2 and 3 all send traffic to node 4. When these nodes in total want to send more than the available bandwidth, the most congested link will be the link immediately downstream of node 3. Thus, the congestion domain will consist of the 4 nodes from node 0 to node 3. Node 0 will be the tail of the domain, node 3 the head.

When a node's total transit and add traffic amounts to more than the full bandwidth, the transit queue of the node with a congested out-link will fill up[2]. When the transit queue occupancy is above a threshold called *low*, the node enters a state called *congested* and if it is has not observed that there are downstream nodes that are more congested, it becomes head of a congestion domain. As head, it starts sending fairness messages, which is the feedback mechanism of the control system. These feedback messages instruct the upstream nodes to restrict their add rate to that of the head's fair rate estimate.

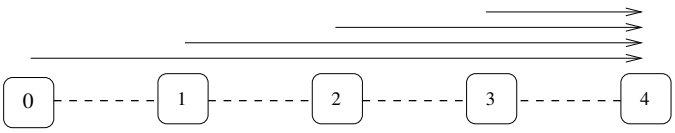


Figure 1: A congestion domain consisting of 4 active nodes: node 0, 1, 2 and 3. The output link of node 3 is the most congested link in the domain, thus node 3 is the head-*, while node 0 is the* tail *of the congestion domain.*

When a fairness message is received by upstream nodes in a congestion domain, these nodes restrict their add rate to the value of the encapsulated fair rate estimate. The head estimates and advertises new fair rates estimates periodically (every aging interval (termed $agingInterval$)). The default duration of an $agingInterval$ is 100 μs.

The time it takes from the head advertises a new fair rate estimate, until it sees the effect of this action, is the time it takes for a fairness message to reach an upstream node, and then the time it takes for the packets from this node, generated in accordance with the newly received fair rate estimate, to reach the head. Hence, in general there is a considerable feedback latency in the control system. This latency, which can be considered the time-constant of the congestion domain, combined with the configuration of the algorithm in the head for calculating fair rate estimates, decides the stability of the RPR fairness algorithm.

The RPR standard has defined two algorithms for the calculation of the fair rate estimates. In this paper, although the main focus is on the improvement of deficiencies found in the *aggressive* fairness mode, in section II-A, we provide a brief description of the *conservative* fairness mode. For a more detailed overview of the *conservative* fairness mode, refer to [2], [16]. Following the brief overview of the *conservative* fairness mode, we provide a brief overview of the *aggressive* fairness mode in section II-B. For a detailed overview of the *aggressive* fairness mode, refer to [8].

A. Conservative Fairness Mode

In the *conservative* fairness mode, when using the node design with two transit queues, the goal of the fairness algorithm is to maximize the throughput of the congested link, while at the same time keeping the occupancy of the

[2]RPR nodes may have one or two transit queues. In the case of a node with two transit queues, the highest priority traffic will use one transit queue, while the two lower priority classes will use the other transit queue. In the RPR model used in this paper there are two transit queues, but since all data traffic will be of the lowest priority, the high priority transit queue will be empty.

transit queue for low and medium priority traffic (termed the Secondary Transit Queue (*STQ*)) between two thresholds termed *low* and *medium*. The layout of the *STQ* with associated queue thresholds and their default level relative to another, higher threshold, termed *high*, is shown in Fig. 2.

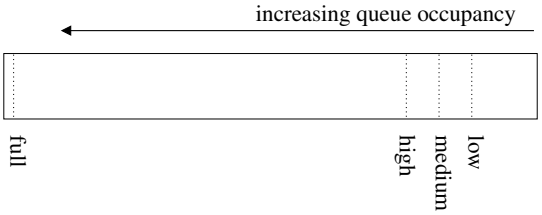


Figure 2: Secondary Transit Queue and associated queue thresholds. Note that in the figure, the relative location of the thresholds w.r.t. the high threshold is according to the default settings specified by the RPR standard. This layout applies to the aggressive *fairness mode as well. In the* aggressive *fairness mode however, the* medium *threshold is not used.*

In this fairness mode, to allow for a more modestly sized *STQ* than is used for the *aggressive* fairness mode, several techniques are used: i) once the *STQ* occupancy exceeds the *medium* threshold, the actions taken to reduce the queue occupancy are more *aggressive* (fair rate estimate reductions are done faster) than the corresponding actions taken to increase the queue occupancy once the occupancy falls below the *low* threshold; ii) to prevent that the fair rate estimate is adjusted faster (or slower) than the effect can be observed, the time-constant of the congestion domain is measured and used to set a timer termed Fairness Round Trip Time (FRTT).

Thus, once the fair rate estimate has been adjusted, the FRTT timer is reset and no additional rate adjustments are performed before the FRTT timer expires.

To provide a means of congestion recovery, should the *STQ* occupancy exceed the *high* threshold, the fair rate estimate is reduced every $agingInterval$, regardless of the status of the FRTT timer.

B. Aggressive Fairness Mode

In the *aggressive* fairness mode, once a node utilizing the two-transit buffer design has become congested, it continues to add traffic until the transit queue occupancy reaches the *high* threshold (shown in Fig. 2). At this point, the head stops its add traffic, until the upstream nodes have reduced their send rate so much that the head's transit queue occupancy decreases below the high threshold.

We have previously shown that for the *aggressive* fairness mode, the value used by the head as its fair rate estimate, is the head's own add rate run through a 2nd order low-pass filter. Below, we will give the reader a short summary of the properties of this filter. A detailed analysis of the *aggressive* fairness mode with associated stability properties can be found in [8].

The low-pass filter is shown in Fig. 3. In the figure, the box with the marking z^{-1} denotes that the value on the output of the box is delayed one sampling period (i.e. one $agingInterval$) as compared to the value on the input of the box. The filter input and output-values, denoted respectively $X(z)$ and $Y(z)$ are the Z-domain representations of the discrete time-domain signals $x(n)$ and $y(n)$, where $x(n)$ is the add rate of the head and $y(n)$ is the low-pass filtered version, $lpAddRate(n)$.

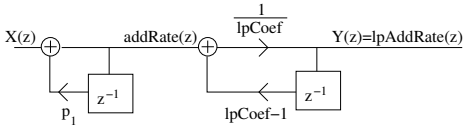


Figure 3: Block diagram of the second-order low-pass filters yielding the fair rate estimate based on the congestion head's own send rate $x(n)$

In the figure below, the value of the constants p_1 and p_2 comes from the RPR configuration parameters *ageCoef* and *lpCoef* as shown below:

$$p_1 = \frac{ageCoef - 1}{ageCoef},\ ageCoef \in \{1, 2, 4, 8, 16\} \tag{1}$$

$$p_2 = \frac{lpCoef - 1}{lpCoef},\ lpCoef \in \{16, 32, 64, 128, 256, 512\} \tag{2}$$

This gives a transfer function, $H(z)$, of the filter as shown in (3) below.

$$H(z) = \frac{Y(z)}{X(z)} = \frac{1}{lpCoef} \cdot \frac{z^2}{(z - p_1)(z - p_2)} \tag{3}$$

Under the assumption that the input to the filter, $x(n)$ is a step-function, the output of the filter consists of a constant part and two first-order filter functions. Further, when $lpCoef > ageCoef$, the filter time-constant is dominated by the second filter stage. In this case, the time-constant, τ, of the filter, can be approximated by the expression:

$$\tau \approx \frac{-1}{ln(p_2)} = \frac{-1}{ln(\frac{lpCoef-1}{lpCoef})} \approx lpCoef\ [agingIntervals] \tag{4}$$

It has been shown that the *aggressive* mode fairness algorithm does not always reach a stable state [8]. In general, the calculated fair rate estimate always varies (oscillates) initially in response to transient load conditions. If the (new) traffic load is stable, these oscillations should ideally decay as the fairness algorithm converges to the new fair division of sending rates. For some scenarios however, even under (new) stable load conditions, the *aggressive* mode fairness algorithm does not converge, and the rate at which each different node is allowed to send, continues to oscillate.

In the next section, we will present and discuss a class of scenarios, for which the *aggressive* mode fairness algorithm does not converge.

III. Unbalanced Traffic Scenario

As described above, congestion occurs when several senders at the same time want to transmit more data than the link capacity (bandwidth) over the same link can sustain (all links in RPR have the same capacity). Some senders may be greedy, i.e. they want to send as much as possible. Other senders send at a limited rate. For modest senders, i.e. senders sending less than their fair share, RPR should not impose any rate restrictions. For nodes having more to send than their fair share, RPR should restrict their sending rate to their fair share.

In the case where the head of a congestion domain is a modest sender, utilizing the **aggressive** fairness mode, this induces permanent oscillations in the throughput of its upstream greedy senders. An example of this, using 1Gbit/s links, is shown in figure 4 below.

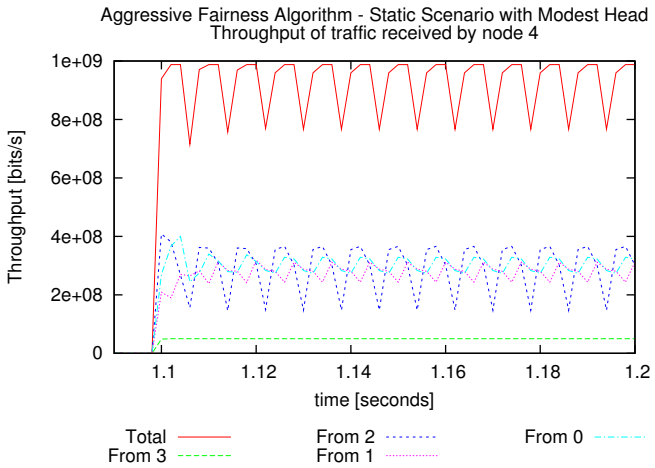


Figure 4: Modest congestion head node. The figure plots throughput per sender node measured at node 4.

As seen in the figure, the induced oscillations in the throughput of the individual nodes upstream of the head causes the aggregate throughput of the congested link to oscillate. This leads to a reduction in link utilization of the congested link. For illustrative purposes, we have checked the actual numbers of this scenario. When the congestion domain head is configured as modest sender, adding traffic at 5% of the link rate, the link-utilization converges toward a value of 92.8%. For the same scenario, if we configure the head as a greedy sender, the link utilization for the congested link converges to a value of 99.8%. I.e., for a 1Gbit/s link-speed, we incur a throughput loss of 70 Mbit/s. For scenarios where the head node is a modest sender, the actual throughput loss depends on several factors, notably the node configurations and the size of the congestion domain.

Another effect of the throughput oscillations, is that the occupancy of the *STQ* in the head will vary more, thus leading to greater delay variations for medium and low-priority traffic.

The origin of this oscillatory behavior comes from the method used by the *aggressive* fairness mode to control the send rate of its upstream neighbors. In the event of a congestion situation, the head of the congestion domain uses its own add rate as its fair rate estimate, deciding how much its upstream neighbors should be allowed to send over the congested link. As we already know, the (modest) head adds traffic over the congested link at a rate that is below the fair rate. Thus as long as the *STQ* occupancy of the head exceeds the *low* threshold, the rate values encapsulated in the fairness messages will be the head's own modest add rate. The effect of these fairness messages, when received by the upstream neighbors, is that for a period o f time, the aggregate of traffic crossing the congested link will be less than the capacity of the same link. As noted above, this results in a reduction in the link utilization of the congested link.

Before we proceed to the description of our contributions, in the next section, we present the goals we seek to achieve.

IV. Improving on Stability and Throughput Performance for Modest Head Scenarios

As discussed and shown in section III, the presence of a modest head in a congestion domain where the head utilizes the *aggressive* fairness mode, leads to throughput loss and consequently reduced link utilization for the most congested link in the congestion domain. Now the question becomes – is there anything that can be done to improve the performance of the *aggressive* fairness mode in this type of scenarios?

One alternative would be to implement an alternative fairness algorithm where this type of problem is not an issue [9], [10], [17].

A second alternative is to modify the current RPR *aggressive* fairness mode in order to improve its performance in the presence of a modest head [11]. An inherent constraint for this alternative is that the proposed modification must fit within the given RPR standard framework w/o yielding undesirable side-effects, such as performance degradations for other traffic scenarios.

A third alternative would be to design an alternative fairness mode, that fits within the RPR framework, resolves the above performance deficiencies and can be used regardless of the sending behavior of the congestion head.

Our proposed solution belongs to the third alternative above, but can also be used as outlined for the second alternative (i.e. as an alternative to the *aggressive* fairness mode, when the congestion head has a modest sending behavior).

In the context of the second alternative, it is clear that there must be a way for a congestion head to determine its operating context, to know whether it is operating as a modest head or not (i.e. to decide which fairness "mode of operation" to use).

Based on the above discussion, we are ready to introduce a set of Design Objectives (DOs) which we seek to meet when designing our contributions.

DO 1: Remove the oscillatory behavior in the presence of a head with a modest sending behavior.

DO 2: Retain the behavior of the original algorithm in the presence of a head with a demand greater than or equal to its fair share.

DO 3: Minimize the changes (state information, code and processing requirements) to the current algorithm.

DO 4: Fit our modifications into the framework given by the RPR standard.

DO 5: Allow for interoperability of nodes in a congestion domain regardless of the fairness mode used.

In the next section (section V), to allow for the fulfillment of the above Design Objectives in the context of alternative 2 outlined above, we propose a context determination method. Then in section VI, we describe our *moderate* fairness mode.

V. Monitoring of Congestion Head's Bandwidth Utilization

We can envision at least two strategies for providing the context determination method described in the previous section.

One way is to compare the head's own add rate, *lpAddRate*, to the total available bandwidth divided by the number of active nodes (the fair rate for a congestion domain with all greedy senders having equal weights). If the head is sending at a rate close to or above this ratio, one can assume that the head's own add rate is a good rate estimate that can be used for distribution to its upstream neighbors. If the head is sending at a rate below this ratio, we can assume that the head's capacity demand is below the head's fair share and that the head's own add rate should not be used as a fair rate estimate.

This is the method used by Zhou et al. in [11].

There are several problems with this method. **Firstly**, this requires knowledge about the number of nodes sending traffic over the congestion point. The calculation of the number of nodes sending traffic over the congestion point is not mandatory for nodes running the *aggressive* fairness mode. **Secondly**, and more seriously; the use of a rate threshold for switching between two different rate calculation methods may lead to instabilities. This is especially true when the rate threshold is set to the fair rate. As the rate algorithm converges towards the fair rate, it is expected that the add rate of the congestion head will oscillate around the fair rate [8]. Thus this will cause the algorithm to constantly switch between two different rate calculation methods. As seen in their paper, the oscillations are reduced by a factor 2 as compared to the original RPR fairness algorithm, but the magnitude of the oscillations are still 50% of the line rate. The throughput improvement is 13.4%, leading to a link-utilization of ≈87%.

Thirdly, below, we will argue that the basic underlying assumption, that the head is adding traffic at a rate $R_{head} \geq \frac{C}{|\mathcal{X}|}$ (where C is the link capacity and $\mathcal{X}$ is the set of nodes sending traffic over the congested link), is not an indication of the correctness of the head's fair rate estimate.

Let us assume that the above assumption was true, and that we have a scenario, with the set $\mathcal{X}$ of active senders, sending traffic over the congestion point.

With a head adding traffic at a rate of $\frac{C}{|\mathcal{X}|}$, then the head's fair rate estimate ($R_{fair\ estimate}$) would be given by (5) below.

$$R_{fair\ estimate} = \frac{C}{|\mathcal{X}|} \tag{5}$$

Further, let us assume that the congestion domain has a set of nodes $\mathcal{Y}$, where each node $i \in \mathcal{Y}$, is sending traffic at a rate R_i, lower than the rate of the head (R_{head}), while the remaining nodes, $|\mathcal{X}| - |\mathcal{Y}| - 1$, send as much as possible. We will now show the error in the fair rate estimate, given by (5) and the theoretical fair rate, shown in (6) below.

$$R_{fair} = \frac{C - \sum_{i \in \mathcal{Y}} R_i - \frac{C}{|\mathcal{X}|}}{|\mathcal{X}| - |\mathcal{Y}| - 1} \tag{6}$$

Thus the error, E, in the fair rate estimate is given by (7) below.

$$E = R_{fair} - R_{fair\ estimate} = \frac{C - \sum_{i \in \mathcal{Y}} R_i - \frac{C}{|\mathcal{X}|}}{|\mathcal{X}| - |\mathcal{Y}| - 1} - \frac{C}{|\mathcal{X}|} \tag{7}$$

As shown in (8) below, we can make some general observations for this function.

$$\sum_{i \in \mathcal{Y}} R_i \to 0 \text{ and } |\mathcal{Y}| \to (|\mathcal{X}| - 2) \Rightarrow E \to C \cdot \frac{|\mathcal{X}| - 2}{|\mathcal{X}|} \tag{8}$$

I.e. as the demand of the nodes in the set $\mathcal{Y}$ decreases and the number of nodes in the set approaches the remaining number of nodes in the congestion domain except for one (the worst-case), the error approaches $C \cdot \frac{|\mathcal{X}|-2}{|\mathcal{X}|}$. Thus as the size of the congestion domain ($|\mathcal{X}|$) increases, $E \to C$. I.e. the error of the fair rate estimate approaches the link-rate.

In conclusion, it is clear that the use of the node's own add rate as a fair rate estimate results in large (fair rate estimation) errors for some scenarios. Furthermore, it is clear that it is not possible for the head to reason about the correctness of the use of its add rate as a fair rate estimate, purely based on its knowledge of the link-rate, its own add rate and the number of active senders over the congested link

A second, and novel strategy, for monitoring whether the congestion head is utilizing its fair share of the bandwidth, is to measure the fraction of time when the client is allowed to send traffic but chooses not to do so. By this, a congested node is able to intelligently reason about the usability of its own add rate as a fair rate estimate. Given the design of the RPR MAC layer, this can be easily accomplished. By monitoring the signal (denoted *sendC* in the standard) sent from the MAC layer to the MAC client we can measure the aggregate of the fraction of the time that i) this signal indicates that traffic can be transmitted on the ring, regardless of the destination

address and ii) the MAC layer does not receive a packet from the MAC client. This aggregate represents the link capacity the local node chooses not to use for local traffic. For a node with a traffic demand equal to or greater than its fair share, this aggregate will be zero (as it always have something to transmit when it is allowed to).

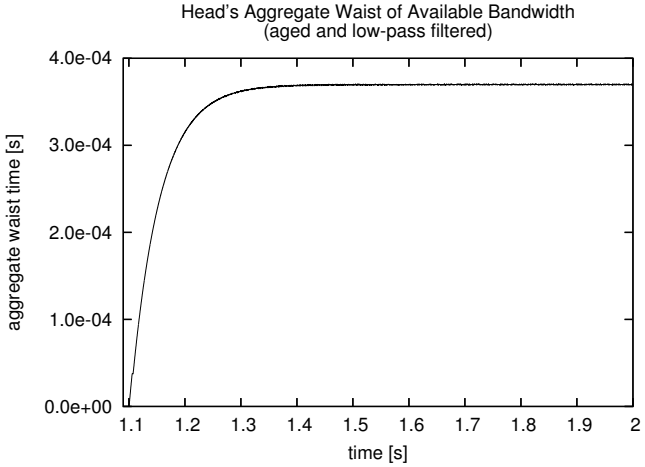


Figure 5: Modest congestion head node. The figure plots the aged and low-pass filtered measured aggregate of the time the head node chooses not to utilize the available bandwidth for transmission of local traffic.

For the scenario shown in figure 1, with a modest head, we have measured this (aged and low-pass filtered) aggregate. This is shown in figure 5. By use of the same aging and low-pass filtering as is used on other measured variables in the RPR standard, we get a value which will converge towards $ageCoef \cdot agingInterval = 400\ [\mu s]$ as the amount of added traffic by the head approaches 0 (a very modest head). For the specific scenario shown here, we see that the long term average waist (denoted *lpWaistTime*) of the head is $\approx 370\ [\mu s]$. Thus clearly, the value of $lpWaistTime > 0$ and by inspecting the value of *lpWaistTime*, the head will know that it is operating in a modest head context and that it should not use its own add rate as its fair rate estimate.

Given this context determination function, we are ready to discuss the problem of calculating a fair rate estimate for distribution by the head to its upstream neighbors.

VI. Alternative Fair rate estimate calculation

Given that the head of a congestion domain is not utilizing fully the bandwidth available for local traffic, we know that the use of its local add rate as a fair rate estimate, results in permanent throughput oscillations for its upstream active neighbors. This also leads to a reduction in link utilization of the congested link.

From previous work, we know that the way rate adjustments are performed are critical for the stability of an RPR ring [8]. If rate adjustments are made too fast, not allowing enough time to observe the effect of previously issued fair rate estimates, the fairness algorithm will not converge to the fair rate. Similarly, if rate adjustments are made too slowly, the algorithm will converge, but may result in unfair division of bandwidth and/or reduced link-utilization during the transient period as well as prolong the duration of the transient period.

If the new rate estimation method differs greatly from the method currently used, this may alter the convergence/stability properties of the fairness method significantly. Thus warranting a complete study of all aspects of the stability and fairness properties of the proposed algorithm.

As stated in the introduction, the goal of the control system is to provide a fair sharing of the bandwidth of the congested link. In the case of a modest head, this means a fair sharing of the available bandwidth minus the fraction used by the head. We know that the theoretical fair rate value to be distributed to the upstream neighbors reside somewhere in the region $\langle lpAddRate, bandwidth - lpAddRate \rangle$ (where $lpAddRate$ is the add rate of the congestion domain head). The exact value depends on the number of upstream active nodes, their weights and their demand. In the case of all greedy senders having equal weights, the fair rate value to distribute to upstream node would be $\frac{bandwidth - lpAddRate}{active\ senders - 1}$. The head node is regarded as an active sender, but does not have any additional bandwidth demands, thus we subtract 1 from the active sender count.

Further, let us assume that an additional requirement to the fairness algorithm is to maintain the *STQ* occupancy at a relatively constant level above 0 and below the *high* threshold. By doing this, we obtain three things: i) we keep the transit delay (for low and medium priority traffic) through the congested node at a relatively fixed level; ii) we ensure that the node always have traffic to send once the output-link becomes available, thus avoiding under-utilization of the link; and iii) we ensure that local traffic gets its fair share (or a fraction of it in the case of a modest head) as the RPR scheduling algorithm ensures the scheduling of both local (within the estimated fair rate bound) and transit traffic while the *STQ* occupancy is below the *high* threshold.

Now, what can be done to achieve this? For the simplicity of the discussion, we assume that the demand of the upstream nodes is stable and that they always have more to send than the capacity of the congested link allows. When the head of the congestion domain calculates a fair rate estimate to be distributed to its upstream neighbors, we know that at some future time, the resulting aggregate traffic from the upstream nodes is either too high (the *STQ* occupancy will increase), too low (the *STQ* occupancy will decrease) or correct[3] (the *STQ* occupancy remains constant).

To achieve a fair division of bandwidth, we monitor the **occupancy** of the *STQ*, just like in the original algorithm. In addition to this, we also monitor the **growth-direction** of the *STQ* occupancy.

[3] A *STQ* occupancy that remains constant at 0 or at the *high*-threshold is not an indication of a correct rate-estimate. But these are special cases that requires special handling.

A. Operation of the Moderate fairness mode

The *aggressive* fairness mode declares a node as *congested* when the *STQ* occupancy exceeds the *low* threshold. From this point onwards, as long as the *STQ* occupancy remains above the *low* threshold, the node sends back-pressure messages to its upstream neighbors. When the *STQ* occupancy falls below the *low* threshold again, the node reenter the *uncongested* state and the node signals to its upstream neighbor(s), that they are allowed to increase their add rate. This rate increase is done according to their own configuration settings.

In the *moderate* fairness mode, a node (that was previously not congested) becomes congested when the *STQ*-occupancy exceeds the *low*-threshold. Once a node has become congested however, the transition back to the uncongested state is done first once i) the *STQ*-occupancy has fallen below the *low*-threshold, and ii) the fair rate estimate equals the maximum allowed rate. The slight modification associated with the transition from the congested back to the uncongested state is to avoid loosing control over the sending behavior of upstream nodes, during transient periods where the *STQ* occupancy falls below the *low* threshold.

In the absence of a locally available and usable fair rate estimate (remember that for a modest head scenario, the use of the node's own add rate, $lpAddRate$, prevents the *aggressive* fairness mode to stabilize), we introduce a locally maintained **rate estimate**. Let us denote this estimate *mRate* (moderate Rate estimate). We also maintain aged and low-pass filtered versions of this variable, denoted respectively *ageMRate* and *lpMRate*. The value of ***lpMRate* is the fair rate estimate**, which is distributed to the upstream neighbors in the fairness messages.

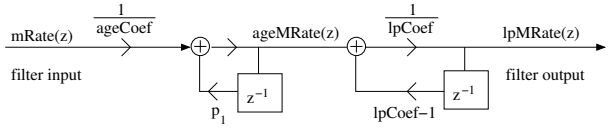


Figure 6: Second order low-pass filter as applied to the rate estimate $mRate$. The output of the filter, $lpMRate$, is the fair rate estimate, distributed to upstream neighbors by a congestion head.

We use the same[4] two-stage second-order low-pass filter construct, used by both the *aggressive* and the *conservative* modes as shown in Fig. 6. In section II, we showed that for the values used for the configuration settings $ageCoef$ and $lpCoef$ ($p_1 = \frac{ageCoef-1}{ageCoef}$), this filter can be approximated by a first-order low-pass filter with a time-constant, $\tau \approx lpCoef \cdot agingInterval\ [s]$. In our *moderate* mode, the input to this filter is the variable, *mRate*, while the output of the filter, $lpMRate$, is the fair rate estimate.

The basic idea of our fairness mode is simple. If the *STQ* occupancy of the congested node increases, the fair rate estimate is too high and must be decreased. Correspondingly, if the *STQ* occupancy of the congested node decreases, the fair rate estimate is too low and must be decreased.

[4]With the exception of a divisor $ageCoef$ applied at the input.

To aid the convergence process, we use a rate interval, limited by a maximum and a minimum value, denoted respectively $mRateMin$ and $mRateMax$. The purpose of this rate interval is to ensure that, during increasing *STQ* occupancy, the use of the minimum rate value as the fair rate estimate, guarantees (at some future time) a decreasing *STQ* occupancy. Correspondingly, the use of the maximum rate value as the fair rate estimate, guarantees (at some future time) an increasing *STQ* occupancy. The challenge then, is to maintain and use this interval, so that the *STQ* occupancy can be kept at a relatively constant level (i.e. at the *low* threshold) and at the same time avoid that the *STQ* occupancy falls to 0 or exceeds the *high* threshold. Note that the size and location of the rate interval, as given by the values of $mRateMin$ and $mRateMax$, will change during the convergence process as well as in the case of a change in the applied load.

Upon transition from the uncongested to the congested state, we initialize $mRateMin$ to the value of the node's own send rate, $lpAddRate$. The value of $mRateMax$ is set to half of the maximum allowed rate. We also set the fair rate estimate, *lpMRate*, to $mRateMin$, while the low-pass filter input (see Fig. 6), *mRate*, is set to *mRateMax*.

Below follows a description of one iteration of the cyclic convergence process, starting as we have an increasing *STQ* occupancy at the head, and the *STQ* occupancy is below the *low* threshold.

When the *STQ* occupancy increases, once the *STQ* occupancy exceeds the *low* threshold, we decrease the fair rate estimate towards the minimum value of the rate interval ($mRateMin$) by setting the input of the low-pass filter to $mRateMin$. Once the fair rate estimate has been sufficiently reduced, the *STQ* occupancy will start to decrease. At this point for the given load, we know that in the future, to ensure a decreasing *STQ* occupancy, the fair rate estimate need not be set lower than its current value. Thus, we set $mRateMin$ to the current fair rate estimate.

Next, to oppose the decreasing *STQ* occupancy, we increase the fair rate estimate towards the maximum value of the rate interval ($mRateMax$).

Depending on the *STQ* occupancy, this is done in one of two ways. If the *STQ* occupancy is above the *low* threshold, to avoid increasing the rate estimate too much and too fast, we set the input of the low-pass filter to $\frac{mRateMin+mRateMax}{2}$. If this is not enough, and the *STQ* occupancy falls below *low* threshold, the input of the low-pass filter is set to $mRateMax$.

Once the fair rate estimate has been sufficiently increased, the *STQ* occupancy will, as a result, start to increase. At this point for the given load, we know that in the future, to ensure an increasing *STQ* occupancy, the fair rate estimate need not be set higher than its current value. Thus, we set $mRateMax$ to the current fair rate estimate. By this, the cycle is concluded and we are back to the starting point. For each iteration of this cycle, the size of the rate interval is improved (reduced).

The actual increase/decrease behavior follows an exponential ramp function, given by the properties of the second-order

low-pass filter shown in Fig. 6. During the periods[5] of increasing *STQ* occupancy, the filter-input is set to $mRateMin$, thus ensuring a monotonic decrease of the fair-rate estimate towards $mRateMin$. Correspondingly, during the periods of decreasing *STQ* occupancy, the filter-input is set to $mRateMax$, thus ensuring a monotonic increasing of the fair-rate estimate towards $mRateMax$.

The exponential ramp function ensures that the time, used to adjust the fair-rate estimate between the max/min values, remains constant, regardless of the size of the rate-interval. Thus, during the convergence process, the narrower the rate interval gets around the RIAS fair rate, the slower the fair rate estimate is adjusted. This way, the variations in throughput during steady-state are minimized.

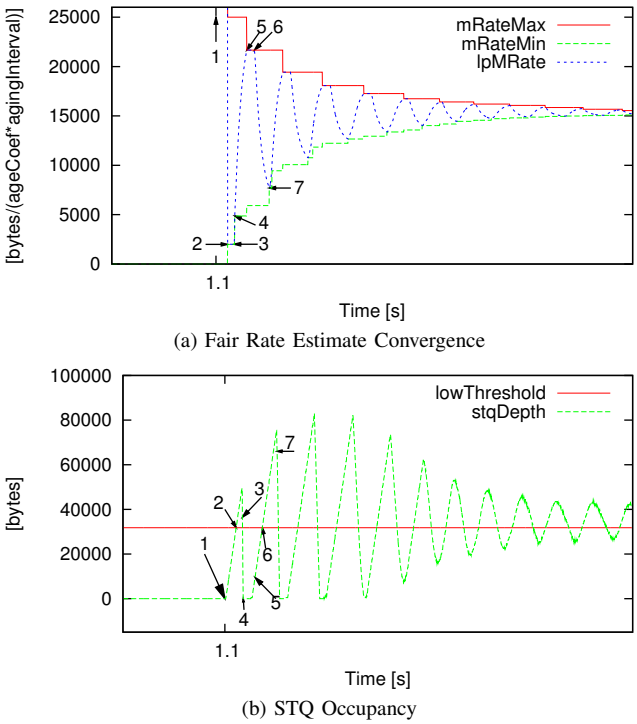
For the simple scenario shown in Fig.1, we will show the convergence process of our *moderate* fairness mode. In the scenario, we assume a stable demand by the individual nodes. The example illustrates, without loss of generality, the convergence for the transition between a no-load situation, and a max-load situation (i.e. a worst-case situation). For a dynamic scenario, the load-change is typically much smaller. Thus in this case, the task of the fairness mode is to shift the established rate interval higher or lower, so that the new fair rate is included in the interval. This is done by expanding the rate interval on one side before continuing the convergence cycle.

Below, the convergence towards the fair rate is illustrated for the scenario shown in Fig. 1, using 410μs (82km) link-delays and a value of 32 for the lpCoef parameter. The convergence of the fair rate estimate, *lpMRate*, is shown in Fig. 7a while the corresponding *STQ* occupancy for the *head* during the same period is shown in Fig. 7b. Significant points in the plots and the discussion have been marked with labels.

Let us assume there are currently no active nodes on the ring. At time t_1 (the point labelled 1), nodes 0, 1, 2 and 3 all start to transmit traffic to node 4. This will cause the *STQ* occupancy of node 3 to start to fill.

At the point labelled 2, node 3 becomes congested, thus we perform the initial variable assignments as described above. From this point, the value of *lpMRate* will increase towards *mRateMax*. The effect on the *STQ* occupancy is that at first, it will have a transient initial increase, before the effect of the reduced rate value is observable at the head (at point 3). As long as the *STQ* occupancy is above the *low* threshold and decreasing, to oppose the decreasing *STQ* occupancy, we set the input to the low-pass filter to $\frac{mRateMax+mRateMin}{2}$. If this is not sufficient to oppose the negative *STQ* growth. Once the *STQ* occupancy falls below the *low* threshold, we set the input to the low-pass filter to *mRateMax*. The result is seen at point 5, where the increased output-value of the low-pass filter results in an increase in the *STQ* occupancy. At this point, the value of *mRateMax* is replaced by the current output value from the low-pass filter, *lpMRate*. Thus effectively reducing

[5]The description in this paragraph is a slightly simplified version of that provided above, and is thus not entirely correct.



(a) Fair Rate Estimate Convergence

(b) STQ Occupancy

Figure 7: Illustration of fair rate estimate and STQ occupancy during rate convergence process, using the simulated scenario with a value of 32 for lpCoef and 410μs link-delays (82 km links).

the interval of rates, $\langle mRateMin, mRateMax\rangle$, by improving (decreasing) the maximum value. From this point, until the *STQ* occupancy exceeds the *low* threshold, we keep the input to the low-pass filter constant at *mRateMax*, thus keeping the rate value sent to upstream nodes constant. At point 6, the *STQ* occupancy once again exceeds the *low* threshold, and to prevent *STQ* overflow, we set the input to the low-pass filter to *mRateMin*. This results in a decrease in the output-value of low-pass filter. The effect of this is seen at point 7, where the growth-direction of the *STQ* occupancy once again becomes negative. At this point, we have a new estimate for the minimum rate-value, *mRateMin*, thus effectively reducing the interval of rates, $\langle mRateMin, mRateMax\rangle$, by improving (increasing) the minimum value. At point 7, the current rate adjustment cycle is concluded and the next cycle starts. By this, we are back to the starting point. At point 7, the only difference is that when entering the cycle starting at 7, the interval of rates have been (significantly) reduced compared to the size of the interval when entering the previous cycle (at 3).

The reader may have noticed that an explanation of the point labelled 4 has been omitted. At this point, the *STQ* has become empty (because the value of *mRateMin* is currently too low). To rectify this, and to minimize underutilization of the link, the value of *mRateMin* is corrected slightly based on its current value and the value of *mRateMax*. A similar corrective action is performed if the *STQ* occupancy reaches

the *high* threshold. A second corrective action we perform to avoid underutilization of the congested link, as long as it stays within rate restrictions (if such exist) received from downstream nodes, is to allow the congested node (the head) to add traffic to the output-link as long as the transit queue is empty.

Another observation the reader may have done, is that the points (i.e. 3, 5 and 7) where the *STQ* growth-direction changes is not exactly at the local minimum/maximum points for *STQ* occupancy. This is to avoid making any rate adjustments based on accidental variations in *STQ* occupancy (i.e. measurement "noise").

B. Comparison of the Moderate and the Aggressive Fairness Modes

The *moderate* fairness mode may seem overly complex compared to the *aggressive* fairness mode. Added complexity however, is difficult to overcome. For a greedy head, running the *aggressive* fairness mode, whenever the fair rate estimate is too low, resulting in decreasing *STQ* occupancy, the *head* may quickly (almost immediately) compensate for this using locally added traffic. And vice versa, when the fair rate estimate is too high, local traffic can be held back. This makes it possible to quickly establish self-adjusting rate intervals where the lower rate limit is established once the *STQ* occupancy exceeds the *high* threshold and the higher rate limit is established once the *STQ* occupancy falls below the *high* threshold. The rate limits are not explicitly stored in separate variables, but the value of the local *addRate* rate counter, which is the output signal from the first low-pass filter for the *aggressive* fairness mode, is quickly (in a matter of a few aging intervals) updated to the new rate limit for both the increasing and decreasing rate periods.

The *moderate* fairness mode can in some sense, be considered a generalized version of the *aggressive* fairness mode. In the case of *moderate* fairness, we do not (and cannot) rely purely on the capability of the head to compensate for bad fair rate estimates by the increase or throttling of local traffic. Thus we establish and maintain explicit rate intervals to aid the convergence of the fair rate calculation process. Further, we cannot use the node's measurements of locally added traffic as the fair rate estimate, as the correlation between the node's own add rate and the congestion state is rather weak. Thus we have to introduce a separate rate variable (much the same way as is done for the *conservative* fairness mode). Once a congestion state is established, we regulate the value of the rate estimate within the established rate interval, based on the *STQ* occupancy level and growth direction.

As discussed in section III and shown in Fig. 4, the establishment of a rate interval based on measurements of local traffic only, results in large oscillations in throughput and reduced utilization of the congested link. This is because the rate interval which is established by measurements of local traffic is incorrect – it is not centered around the fair rate. In an interval of rates, where the fair rate should be the center, the center of the interval is effectively placed at a negative offset, where the offset is decided by the amount of traffic transmitted by the *head*.

For the *moderate* fairness mode, once a fair rate estimate has been calculated, we do not see the the full effect of it before a system time-constant later. Further, the establishment and maintenance of maximum and minimum rate values can only be done by observing local maxima and minima points of the *STQ* occupancy. Thus when using the *moderate* fairness mode, the establishment of a rate interval takes more time than for the *aggressive* mode.

However, as we shall see in section VII, the performance of the *moderate* fairness mode, when used in the presence of a greedy head, is comparable to that of the *aggressive* fairness mode.

VII. Performance Evaluation

In this section, we present results for a set of carefully selected simulation scenarios. The goal of this selection is to evaluate and compare the performance of our proposed *moderate* fairness mode to that of the *aggressive* and the *conservative* fairness modes for some fundamental load-patterns. Having done this, it is should be possible to predict the behavior of our *moderate* fairness mode for many other load-patterns.

Thus in section VII-A, we start by comparing the behavior of the *moderate* fairness mode to that of the *aggressive* and *conservative* fairness modes when the head has a greedy sending behavior. Next, in section VII-B we compare the performance of the *moderate* fairness mode to that of the *aggressive* and the *conservative* fairness modes when the head has a modest sending behavior. In section VII-C, we demonstrate the ability of the *moderate* fairness mode to adapt to a dynamic load scenario. Finally, in section VII-D, we have run an extensive series of simulations, where we for each test scenario and each allowed value of the *lpCoef parameter*, test the ability of the *aggressive* and our *moderate* fairness modes to converge to the fair division of rates as the size of the tested network is increased.

In our paper "Congestion Domain Boundaries in Resilient Packet Rings" [15], we analyzed a problem, where the the tail of a congestion domain stops the propagation of fairness messages received from a downstream head. The negative side-effect of this behavior is that a node upstream of the tail may send excessive amounts of traffic, thus preventing the convergence of the fairness algorithm. Thus we incur both unfairness as well as non-convergence of the fairness algorithm. In the same paper, we proposed and evaluated a modification, termed the *tail-fix*, that solved these problems by a conditional propagation of the fair rate estimate beyond the tail of a congestion domain. In the performance evaluation experiment that follows, we make use of the *tail-fix* when evaluating the performance of our proposed *moderate* fairness mode.

A. Convergence when head has a greedy sending behavior

In this experiment, we compare the performance of the *aggressive*, the *moderate* and the *conservative* fairness modes

in the presence of a greedy head node. Further, in the first part of the experiment, we use the context determination function introduced in section V to allow for the selection between the *aggressive* and the *moderate* rate calculation modes, in accordance with a node's sending behavior and congestion status.

We use the same scenario as before (depicted in Fig. 1), but for this test, all nodes send as much traffic as possible. The per node configuration is shown in Table I.

Parameter Name	**Value**
Line Rate	1 [Gbit/s]
Packet size	500 [B] (fixed)
All nodes (in congestion domain) sending behavior	GREEDY
STQ Thresholds - low - high	 31812 [bytes] 120000 [bytes]
rampUpCoef	64
rampDnCoef	64
ageCoef	4
lpCoef	64
link-delay	410 [μs]
Start of traffic	1.1s

Table I: Configuration for comparison of the three fairness modes in the presence of a greedy head.

First, we let the context determination function (i.e. the value of the *lpWaistTime* variable), decide whether the head should use the *aggressive* or the *moderate* fairness modes. In the case of a greedy head. The value of the *lpWaistTime* variable remains at 0, thus the *aggressive* fairness mode is used. Next, we force the use of the *moderate* fairness mode, regardless of the value of the *lpWaistTime* variable. Finally, we force the use of the *conservative* fairness mode, regardless of the value of the *lpWaistTime* variable.

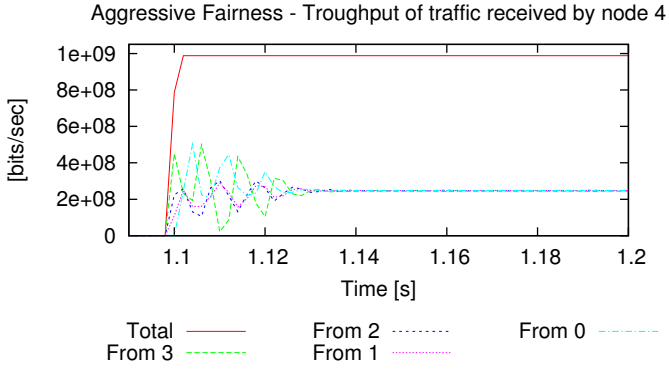


Figure 8: Aggressive *fairness mode with a greedy head. This type of scenario is optimal for the* aggressive *fairness mode, providing faster convergence than the two other fairness modes. At time 1.028s throughput convergence is achieved.*

We start by comparing the throughput convergence of the three fairness modes. From the measurements performed, we calculate the time it takes for the throughput of all flows to stabilize at a level within $\pm 5\%$ of the steady-state (fair) value. For the *aggressive* fairness mode throughput convergence, shown in Fig. 8, we found as expected that its associated

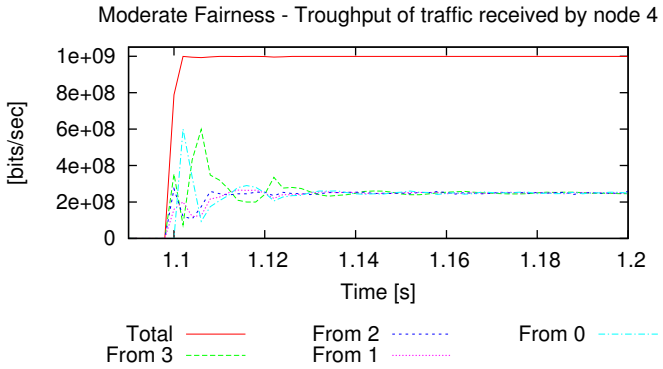


Figure 9: Moderate *fairness mode with a greedy head. The convergence time is 8ms (steady-state is obtained at 1.136s) longer than for the* aggressive *fairness mode for this scenario. Full link-utilization is achieved as fast as for the* aggressive *fairness mode.*

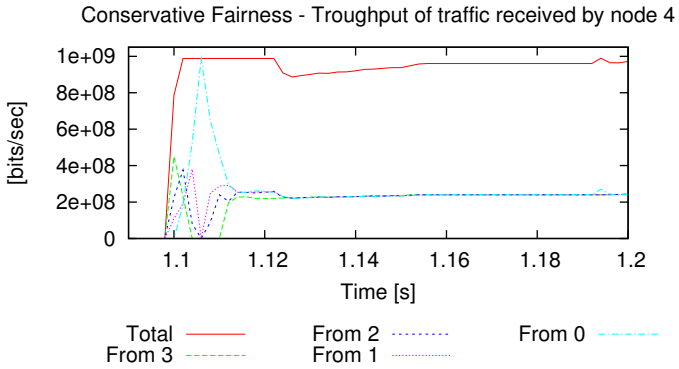


Figure 10: Conservative *fairness mode with a greedy head. The convergence time for the* conservative *fairness mode is much longer than for the other two fairness modes. Additionally, once steady state is achieved (shown in Fig. 11), we incur some reduced link-utilization because of the periodic event of an empty* STQ *buffer in the head.*

convergence time of 28 ms is the shortest (stable state achieved at time 1.128s) of the three fairness modes. Full link-utilization is achieved almost immediately, as the head will utilize any available bandwidth over the congested link, not in use by transit traffic.

The throughput convergence for the *moderate* fairness mode for the same scenario and using the same configuration is shown in Fig. 9. The convergence time here is somewhat higher. Steady state is achieved at time 1.136s, after 36ms. Full link-utilization however, is achieved immediately (once the sum of transit and add traffic is sufficient to fill the output link).

For the *conservative* fairness mode, the throughput convergence for this scenario is much slower. As seen from Figures 9 and 10, at the time where the *moderate* fairness mode has reached steady-state (1.136s), the *conservative* fairness mode has only reached 91% link-utilization. As seen in Fig. 11, full throughput for the *conservative* fairness mode is not achieved until 1.222s. Furthermore, as seen from Fig. 11, the *conservative* fairness mode suffers from sustained oscillations in total throughput, leading to a minor reduction in link-

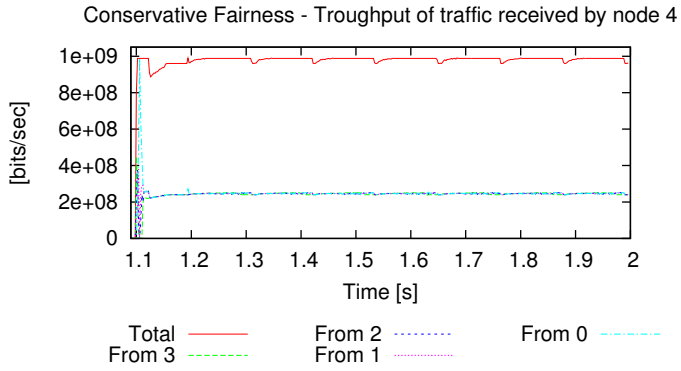


Figure 11: Conservative *fairness mode with a greedy head. The plot shows the reduced link-utilization (reduction in aggregate/total throughput) caused by the periodic under-run of the head's* STQ *buffer.*

utilization.

In addition to the throughput convergence and link-utilization characteristics of the three fairness modes, we can make another observation on the throughput performance of the three fairness modes. In the *aggressive* fairness mode, when the head is greedy, the *STQ* occupancy oscillates around the *high* threshold. The RPR scheduling rules block local low-priority traffic whenever the *STQ* occupancy reaches this threshold. This causes the throughput of the head to fall somewhat below the throughput of its upstream greedy neighbors. For the *moderate* and *conservative* modes, where the *STQ* occupancy oscillates around the *low* threshold (*moderate* mode) or between empty and the *medium* threshold (*conservative* mode), this is not the case. In this case, the head will achieve its (full) fair share of the available bandwidth. This effect can be seen when comparing figures 12, 13 and 14.

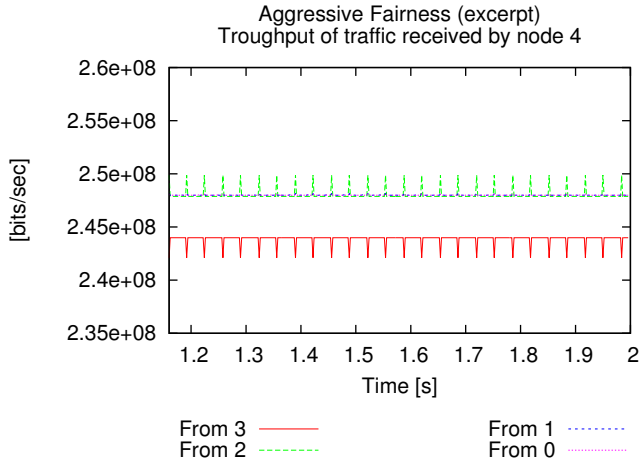


Figure 12: Excerpt of aggressive *fairness mode throughput with a greedy head. As seen in the figure, the traffic from the head (node 3) does not get its full fair share of bandwidth over the congested link. This is caused by the blocking of local traffic once the* STQ *occupancy exceeds the* high *threshold.*

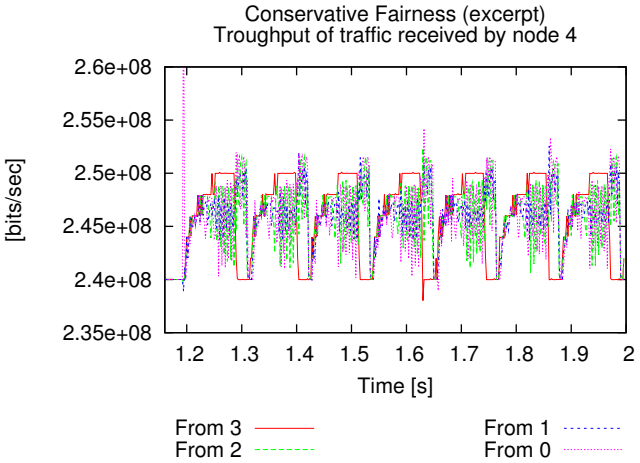


Figure 13: Excerpt of conservative *fairness mode throughput with a greedy head. As seen in the figure, the throughput of traffic from the head is not penalized. This is because in steady-state, the* STQ *occupancy oscillates varies between 0 and the* medium *threshold.*

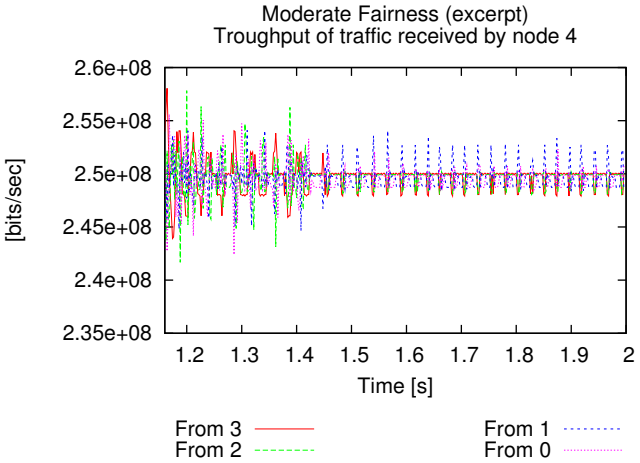


Figure 14: Excerpt of moderate *fairness mode throughput with a greedy head. As seen in the figure, the throughput of traffic from the head is not penalized. This is because in steady-state, the* STQ *occupancy oscillates around the* low *threshold. The variations in throughput may seem large, but are within* ± 2*[packets/averaging interval] of the mean (the fair rate). The averaging period is 2ms.*

B. Throughput Convergence for an Unbalanced Traffic Scenario

In this test, we use the configuration shown in Table II. Notice that the head is configured as a modest sender, sending CBR traffic at 5% of the line-rate.

Parameter Name	**Value**
Line Rate	1 [Gbit/s]
Packet size	500 [B] (fixed)
Head's Sending Behavior	CBR traffic at 5% of the line rate
All other active nodes (in congestion domain) sending behavior	GREEDY
STQ Thresholds - low - high	 31812 [bytes] 120000 [bytes]
rampUpCoef	64
rampDnCoef	64
ageCoef	4
lpCoef	64
link-delay	410 $[\mu s]$
Start of traffic	1.1s

Table II: Configuration for comparison of the three fairness modes in the presence of a modest head.

For this configuration, as expected (discussed in section III), the throughput convergence of the *aggressive* fairness mode (shown in Fig. 15) does not converge to the fair division of sending rates. Furthermore, the magnitude of the induced oscillations results in reduced link-utilization (i.e. the total throughput falls below the maximum obtainable).

The throughput convergence for the *conservative* fairness mode is shown in Fig. 16. As seen from the figure, it takes approximately 72ms just to reach a level of 90% of the available throughput on the congested link. Furthermore, the *conservative* mode of operation causes the *STQ* occupancy to fall to 0 at periodic intervals (just as for the *aggressive* mode). For the *conservative* fairness mode, this leads to a reduction in link-utilization for the congested link. For this particular scenario, the average reduction in link-utilization is $\approx 0.5\%$. Given the small magnitude of the *conservative* fairness mode's throughput-loss during steady-state, this can be neglected for most practical cases.

Finally, in Fig. 17, we show the throughput convergence of our *moderate* fairness mode. As seen from the figure, the convergence time is 52ms. During the transient phase, as the rate control algorithm is working on improving the size and position of the rate interval $\langle mRateMin, mRateMax \rangle$ (discussed in section VI-A), we have some brief periods where the *STQ* becomes empty, thus resulting in a temporary reduction in total throughput. Because of the low demand of the head, there are no local packets available for transmission every time the *STQ* becomes empty. Once steady-state is obtained however, the *STQ* does not become empty, and we do no longer incur any reductions in the link utilization. Furthermore, as seen from the figure, we have a fair sharing of the capacity of the congested link.

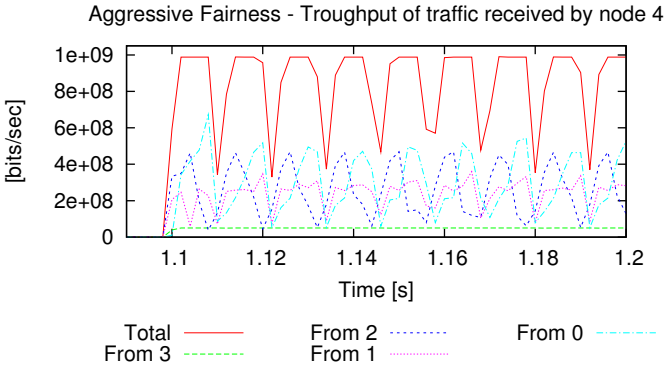


Figure 15: Aggressive *mode throughput convergence when head is modest. As seen, the* aggressive *fairness mode does not converge for this scenario. Additionally, the link utilization (as seen in the reduction on total/aggregate throughput) is reduced.*

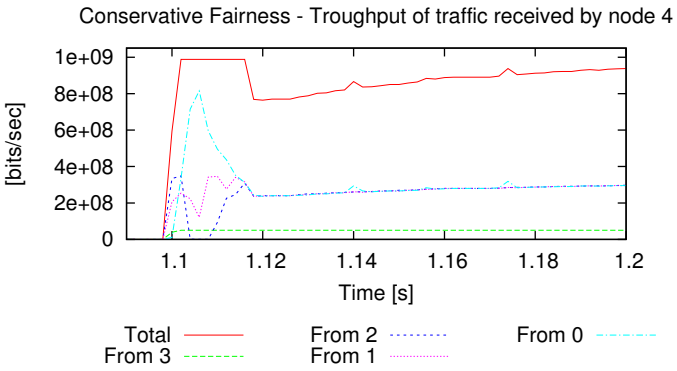


Figure 16: Conservative *mode throughput convergence when head is modest. As seen, the* conservative *fairness mode converges slowly. At time 1.172s, the total throughput has only reached 90% of full link-utilization.*

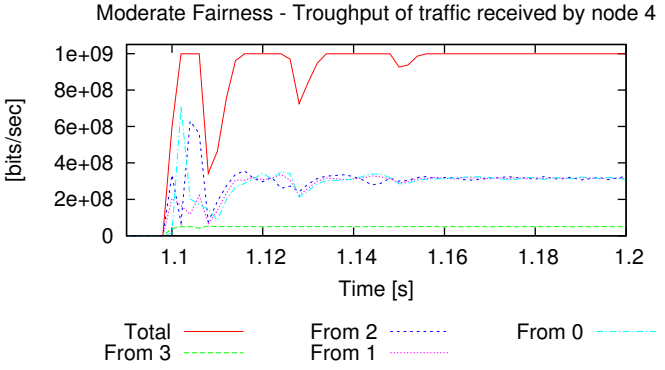


Figure 17: Moderate *mode throughput convergence when head is modest. The* moderate *fairness mode converges within 52ms and after this point, there is no loss in link-utilization.*

C. Convergence for a Dynamic Traffic Scenario

In this test, we use the scenario shown in Fig. 18. The per node configuration is shown in Table III. The purpose of this test is to show that our *moderate* fairness mode is able to adapt to changing load conditions. Thus during the simulation run, we start by, at time 1.4s, decreasing the load, before we, at time 1.6s, increase the load.

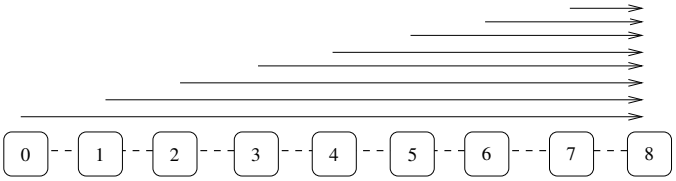


Figure 18: A congestion domain consisting of various active nodes. All active nodes send to node 8. Thus the link between nodes 7 and 8 is the most congested link in the domain.

Parameter Name	**Value**
Line Rate	1 [Gbit/s]
Packet size	500 [B] (fixed)
Head's Sending Behavior	CBR traffic at 5% of the line rate
All other active nodes (in congestion domain) sending behavior	GREEDY
STQ Thresholds - low - high	 31812 [bytes] 120000 [bytes]
rampUpCoef	64
rampDnCoef	64
ageCoef	4
lpCoef	128
link-delay	410 [μs]
Start of traffic	Per node settings specified below

Table III: Configuration for dynamic traffic scenario.

The resulting throughput for the various nodes is shown in Fig. 19. As seen from the figure, our algorithm converges nicely as the number of active senders increases as well as when the number of active senders decreases.

When the load decreases, the rate interval specified by the fairness mode variables $\langle mRateMin, mRateMax \rangle$ (discussed in section VI-A) is located too low. Thus, as a result of this and the modest demand of the head, the *STQ* will become empty during the transient phase, where the fairness mode works to shift the position of the rate interval higher. Thus at time 1.4s, the link-utilization is temporarily reduced (seen by the dip in total throughput), before reaching its maximum value again, following the convergence of the fairness algorithm.

Similarly, when the load is increased at time 1.6s, the rate interval specified by the fairness mode variables $\langle mRateMin, mRateMax \rangle$ is located too high. Thus, as a result the *STQ* occupancy will exceed the *high* threshold. Thus, local traffic from the head, will be blocked during portions of the transient phase, where the fairness mode works to shift the position of the rate interval lower. We also notice that following this, the STQ occupancy falls briefly to 0 (seen by the short reduction in aggregate throughput). Notice that at this

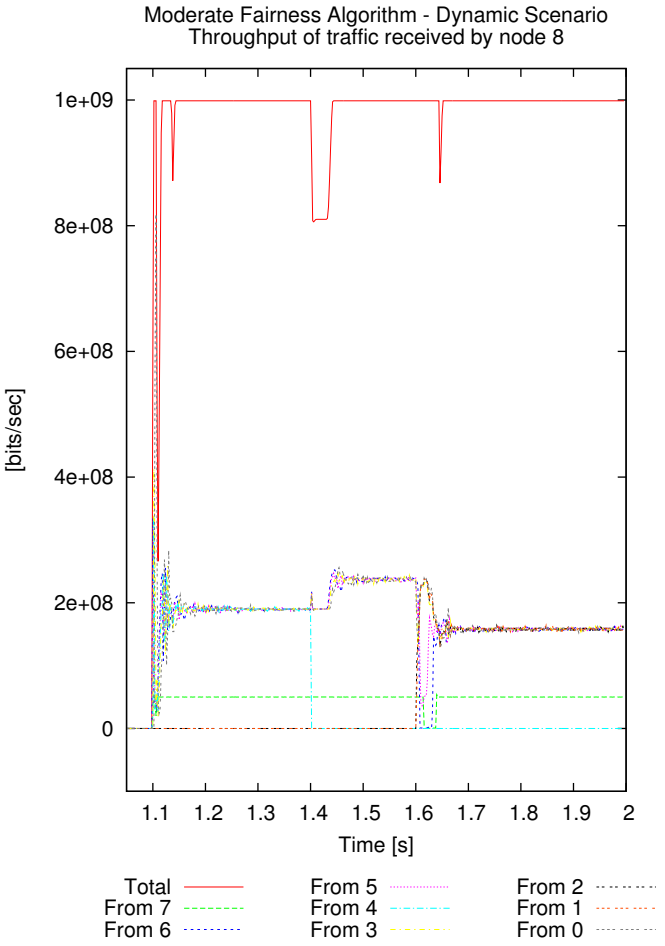


Figure 19: The figure shows the convergence for the moderate *fairness mode for a dynamic traffic scenario. In this scenario, nodes 0, 3, 4, 5, 6 and 7 transmit traffic from time 1.1s. At time 1.4s, node 4 stops transmitting traffic. Finally, at time 1.6s, nodes 1 and 2 start to transmit traffic. All nodes operate as greedy senders, expect for node 7, the head, which transmits traffic at 5% of the link-rate.*

time (1.6s), although the load change is larger (2 additional nodes start sending) than at the previous time (1.4s) (1 node stopped transmitting), the convergence time is approximately equal to that of the previous load change.

D. Convergence Time as a Function of Aggregate Propagation Delay

In this section, we want to compare the performance of our *moderate* fairness mode to that of the *aggressive* fairness mode as the size of the network (congestion domain) to be controlled increases. We use the scenario shown in Fig. 1. In this experiment, for each allowed value of the *lpCoef* parameter, we run a series of simulations. For each simulation run, we increase the network size. That is, we increase the per link propagation delay, thus increasing the time-constant, τ_{system}, of the congestion domain. After the simulation has been executed, we post-process the throughput data obtained, to determine the convergence time of the fairness mode used. A non-converging simulation-execution is represented by an infinite value of the convergence time.

Parameter Name	**Value**
Line Rate ($unreservedRate$)	1 [Gbit/s]
Packet size	500 [B] (fixed)
Upstream Nodes Sending Behavior	GREEDY
STQ Thresholds - low - high	 30000 [bytes], fixed size STQ 120000 [bytes], fixed size STQ otherwise given by (11) and (12)
rampUpCoef	64
rampDnCoef	64
ageCoef	4
lpCoef	$\{16, 32, 64, 128, 256, 512\}$
link-delay	free variable (all links are of equal length)
Start of traffic	1.1s

Table IV: Configuration for scenario where we investigate the relation between convergence of fairness mode and the aggregate propagation delay of the congestion domain.

The experiment is repeated for various load-configurations of the head (greedy and modest) for the two fairness modes as well as using a fixed *STQ* size and a *STQ* size that is a function of the aggregate propagation delay.

The configuration of our experiment is summarized in Table IV and Fig. 20. The results are shown in Fig. 21.

For fast convergence of the fairness mode, let us assume that the region between the *STQ* thresholds *low* and *high* must be able to buffer data for a period given by the system time constant – τ_{system}. For simplicity, we express τ_{system} in terms of the round-trip propagation delay of the congestion domain to be controlled as shown in (9) (disregarding transit path and transmission delays).

$$\tau_{system} \approx (|congestionDomain| - 1) \cdot 2 \cdot t_{lp} \tag{9}$$

In the formula, $|congestionDomain|$ is the size (number of nodes) in the congestion domain and t_{lp} is the length (propagation delay) of a single link in the network (all links are of equal length).

Thus for a period of duration τ_{system}, the head should be able to buffer an amount of transit traffic equal to the maximum amount of local traffic added during the same period. When receiving transit data at the maximum allowed rate, *unreservedRate*, as long at the *STQ* occupancy stays below the *high* threshold, the add rate of the head is limited upwards by the RPR scheduling rules to $addRate \leq \frac{unreservedRate}{2}$.

Thus for the configuration of the *STQ* size and associated thresholds, we get the formula as follows:

$$(high - low) \geq \frac{unreservedRate}{2} \cdot \tau_{system}\ [bits] \tag{10}$$

Furthermore, if we use the RPR standard's default value of the *low* threshold and convert to units of bytes:

$$low = \frac{high}{4} \tag{11}$$

We get:

$$\begin{aligned} high - \frac{high}{4} &\geq \frac{unreservedRate}{16} \cdot \tau_{system}\ [bytes] \\ \Rightarrow high &\geq \frac{unreservedRate}{12} \cdot \tau_{system}\ [bytes] \end{aligned} \tag{12}$$

For the *aggressive* fairness mode, we know that the fairness mode does not converge when the head has a modest sending behavior (represented by the cross in Fig. 21), thus for this mode, we present results for a greedy head sending behavior only.

If we consider the results for the *aggressive* fairness mode, we find that larger size of the *STQ* does not translate directly[6] to the stable operation of larger networks. This is illustrated by Fig. 21 a) and b). Fig. 21 a) shows the convergence result for the scenario, where the size (and associated thresholds) of the *STQ* is a function of the round-trip propagation delay (i.e. τ_{system}) of the congestion domain.

Fig. 21 b) on the other hand, shows the convergence result for the scenario, where the size (and associated thresholds) of the *STQ* is kept fixed according to the settings shown in Table IV. When we compare the two figures, we see that the experiment where we use a fixed *STQ* size, the fairness algorithm converges for networks of a larger size than for the experiment where the *STQ* size is a function of τ_{system}.

The reason for this is that an increased size of the *STQ* buffers (i.e. when the *STQ* size is a function of τ_{system}) contributes to increasing the queueing delay in the transit path of the congestion domain. This leads to an earlier onset (i.e. for networks of smaller size) of the problem described in [15], where node 1 will periodically assume the role as tail of the congestion domain. During the periods where node 1 operates as tail of the congestion domain, node 0 is allowed to transmit traffic at excessive rates, prohibiting convergence of the fairness algorithm.

[6]This does not mean the the *STQ* buffer can be arbitrarily small.

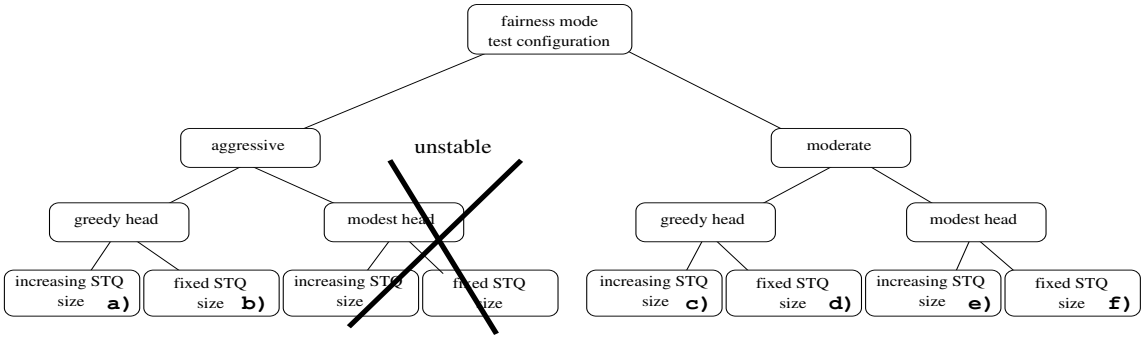


Figure 20: Test configuration overview for section "Convergence Time as a Function of Aggregate Propagation Delay". Notice the letters a-f in the lower right-hand corner of the leaf-nodes in the tree. These point to the corresponding sub-figure number in Fig. 21 on the next page, showing the simulation results for this experiment. The part of the tree with a cross over, corresponds to so called unbalanced traffic scenarios discussed in section III. For these scenarios, the aggressive *fairness mode does not converge, so we do not show any results for these tests.*

If we consider the *moderate* fairness mode (Fig. 21 c-f), we see that it is of importance that the *STQ* and associated thresholds are set large enough. The reason for this, is that the observation[7] of local optima (i.e. changes in growth direction) of the *STQ* occupancy, should be done without the *STQ* becoming empty or exceeding the *high* threshold. While this is OK in the initial phase (it may actually speed up the convergence in many cases) of the convergence process, the process will not converge if the *STQ* thresholds are set so that these thresholds are encountered at every cycle of the convergence process. Equation (12) provides a conservative estimate of the required threshold setting of the *STQ*. The fixed value of the *STQ* $high$ threshold used for experiments d) and f) translates to a limitation of $120 \cdot 10^3 \geq \frac{10^9}{12} \cdot \tau_{system} \Rightarrow \tau_{system} \leq 1.44ms$ for the system time-constant. For the *moderate* fairness mode and a head sending at 5% of the line-rate (Fig. 21 f)), we see that this *STQ* setting provides adequate performance for a network of a size ≤ 10 times this. At this point (i.e. when $\tau_{system} \approx 14000$) however, for $lpCoef = 512$, we see that although the fairness mode converges, it takes considerateby more time than if we increase the sizing of the *STQ* (shown in Fig. 21 e)).

Another property worth noting for the *moderate* fairness mode, is that convergence may take longer if the head is modest than if the head is greedy. This is in large part due to the fact that when the head is greedy, the head will be able to observe the resulting effect of a rate change much faster than when the head is modest. In particular, a change in *STQ* occupancy in response to the head's change in locally transmitted traffic may be observable almost immediately.

Finally, if we compare the performance of the *aggressive* fairness mode to that of the *moderate* fairness mode, we see that convergence for the *moderate* fairness mode takes somewhat longer for the greedy head scenarios. However, provided sufficiently sized and configured *STQ* buffers, the *moderate* fairness mode provides stable operation for a larger range of network sizes, as well as for unbalanced traffic (i.e. modest head) scenarios. While the operation of the *aggressive* mode fairness mode can be modified to support networks of larger size (e.g. by using our modification proposed in [15]), we are not aware of any modifications providing stable operation in unbalanced traffic scenarios.

In concluding this section, it appears that the *moderate* fairness mode provides a stability property where, the convergence of the fairness mode is a function of the network size and setting of the lpCoef parameter. We have previously demonstrated this property for the *aggressive* fairness mode [8].

[7]Remember that it is the finding of the local optima of the *STQ* occupancy that enables the fairness mode to improve the size and position of the dynamic rate range used to limit the adjustment of the *moderate* mode fair rate estimate, $lpMRate$.

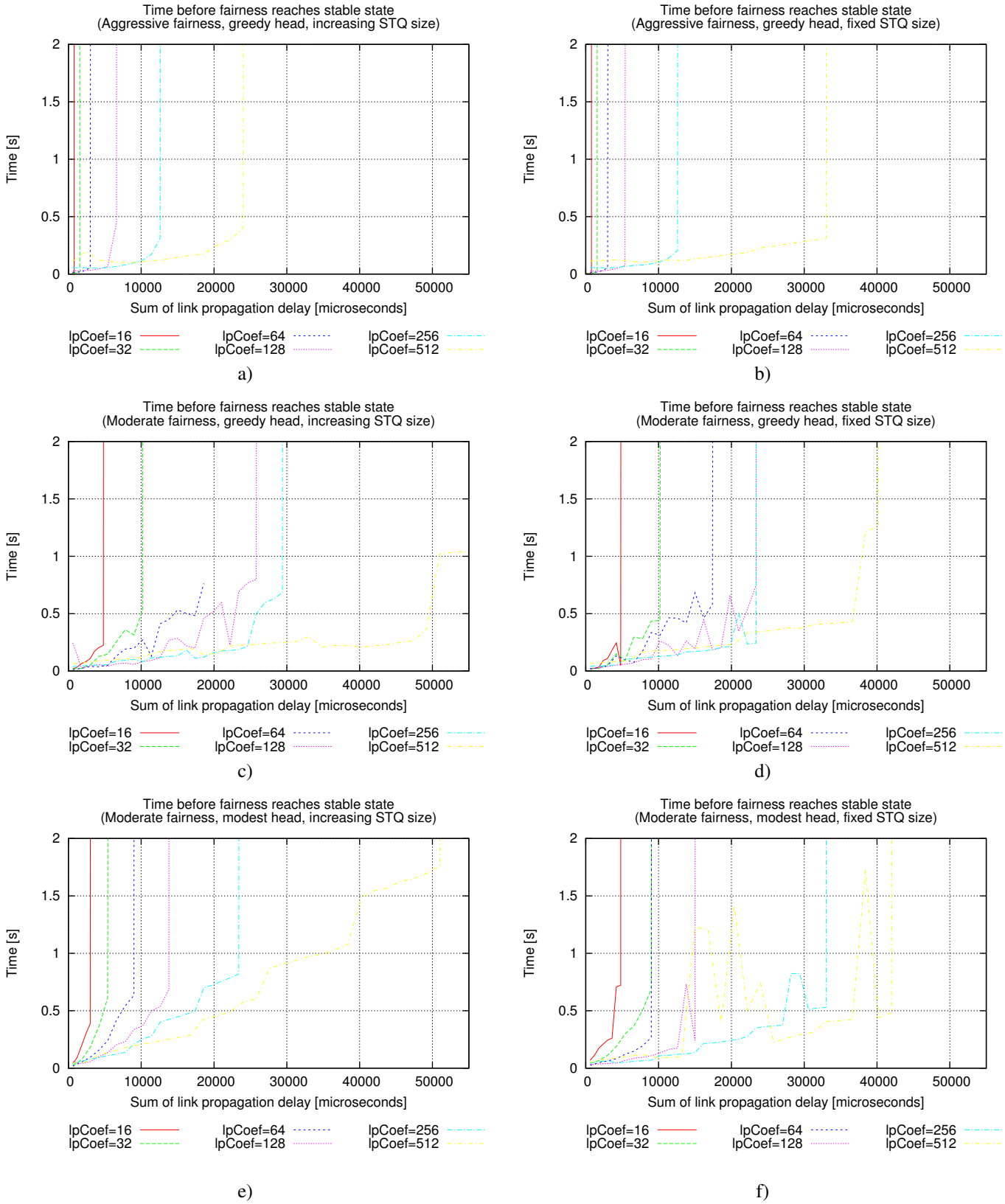


Figure 21: Throughput convergence for RPR Aggressive *and* Moderate *fairness modes.*

VIII. Related Work

Other groups having looked into the problems of sustained large oscillations and resulting loss of link-utilization for unbalanced traffic scenarios, have mostly solved the problem by proposing alternate fairness algorithms that would have to replace the current RPR *aggressive* mode fairness algorithm [9], [10]. Some work have presented results that seems to fit within the framework given by the RPR standard [11]. In that work however, the effect of the proposed modifications is to reduce the symptoms (oscillations) rather than removing them.

IX. Conclusion

In this paper we have proposed two novel contributions. The first contribution is a context determination function, to enable the head of congestion domain to decide whether it is using its fair share of bandwidth over the congested link. The method is based on local information only and does not require knowledge on the number of nodes sending traffic over the congestion point. The second contribution is a novel fairness mode, which we have called the *moderate* fairness mode. The *moderate* fairness mode can be regarded as a generalized version of the *aggressive* fairness mode, where we by use of an explicit, self-adjusting rate interval and the RPR 2-stage low-pass filter construct, control the convergence of the (*moderate*) fair rate calculation process. By use of this fairness mode, the problems reported for so called unbalanced load-scenarios is avoided and we do not incur the large throughput oscillations and resulting reduced link-utilization.

For the tested scenarios, our *moderate* fairness mode outperforms[8] the *conservative* fairness mode in terms of convergence time and link-utilization. For the *moderate* fairness mode, regardless of the sending behavior of the head, the average steady-state *STQ* occupancy in the head equals the *low* threshold. Thus providing a relatively low transit delay for all traffic classes. This is clearly better than for the *aggressive* fairness mode, where for a greedy head, the average *STQ* occupancy equals the *high* threshold.

In the *moderate* fairness mode however, the head cannot compensate for errors in its fair rate estimates purely by the increase or throttling of local traffic. Thus, for some scenarios, the convergence time may be somewhat longer than that of the *aggressive* fairness mode.

As seen in sections VII-A-VII-D, our *moderate* fairness mode converges to the fair division of sending rates, as well as maintaining full link-utilization for a broad range of tested scenarios, thus we claim conformance to DO1 and DO2.

There is no clear definition of a "minimized set of changes", neither is it clear that it makes sense to try to formulate such a definition. Thus conformance to DO3 can be argued. We have however, by use of existing statistics data, introduction of a set of new state variables and a new state machine (for the fair rate calculation) introduced a new fairness mode. Thus, we claim conformance to DO3 and DO4.

[8]Once in steady-state, the link-utilization of the *conservative* mode is only slightly lower than that of the *moderate* fairness mode.

Furthermore, in section VII-A, we have demonstrated that the use of the *moderate* fairness mode in the head, interoperates well with the use of the *aggressive* fairness mode in upstream nodes. We have however not tested interoperability with the *conservative* fairness mode, thus we can only claim partial conformance to DO5.

In conclusion, we will claim that our Design Objectives (DOs) presented in section IV have been partially fulfilled.

X. Further Work

In further work, it would be interesting to test our *moderate* fairness mode for an even broader set of test scenarios.

References

[1] IEEE Computer Society, "IEEE Std 802.17-2004," September 24 2004.

[2] F. Davik, M. Yilmaz, S. Gjessing, and N. Uzun, "IEEE 802.17 Resilient Packet Ring tutorial," *IEEE Commun. Mag.*, vol. 42, no. 3, pp. 112–118, March 2004.

[3] "Token ring access method and physical layer specifications," September 29 1989, IEEE Std 802.5-1989.

[4] E. Hafner, Z. Nendal, and M. Tschanz, "A digital loop communication system," *IEEE Trans. Commun.*, vol. 22, no. 6, pp. 877–881, June 1974.

[5] C. C. Reames and M. T. Liu, "A loop network for simultaneous transmission of variable-length messages," in *Proceedings of the 2nd Annual Symposium on Computer Architecture*, vol. 3, December 1974.

[6] H. van As, W. Lemppenau, H. Schindler, and P. Zafiropulo, "CRMA-II: A MAC protocol for ring-based Gb/s LANs and MANs," *Computer Networks and ISDN Systems*, vol. 26, no. 6-8, pp. 831–840, March 1994.

[7] I. Cidon and Y. Ofek, "MetaRing - a full duplex ring with fairness and spatial reuse," *IEEE Trans. Commun.*, vol. 41, no. 1, pp. 110–120, January 1993.

[8] F. Davik, A. Kvalbein, and S. Gjessing, "An analytical bound for convergence of the Resilient Packet Ring Aggressive mode fairness algorithm," in *Proceedings of the 40th annual IEEE International Conference on Communications*, Seoul, Korea, May 16-20 2005.

[9] V. Gambiroza, P. Yuan, L. Balzano, Y. Liu, S. Sheafor, and E. Knightly, "Design, analysis, and implementation of DVSR: a fair high-performance protocol for packet rings," *IEEE/ACM Trans. Networking*, vol. 12, no. 1, pp. 85–102, 2004.

[10] F. Alharbi and N. Ansari, "Low complexity distributed bandwidth allocation for Resilient Packet Ring networks," in *Proceedings of 2004 Workshop on High Performance Switching and Routing, 2004. HPSR*, April 18-21 2004, pp. 277–281.

[11] X. Zhou, G. Shi, H. Fang, and L. Zeng, "Fairness algorithm analysis in Resilient Packet Ring," in *Proceedings of the 2003 International Conference on Communication Technology (ICCT 2003)*, vol. 1, April 9-11 2003, pp. 622–624.

[12] H. Tyan, "Design, realization and evaluation of a component-based compositional software architecture for network simulation," Ph.D. dissertation, Ohio State University, 2002.

[13] OPNET Technologies, Inc, "OPNET Modeler." [Online]. Available: http://www.opnet.com/

[14] E. Knightly, L. Balzano, V. Gambiroza, Y. Liu, P. Yuan, S. Sheafor, and H. Zhang, "Achieving high performance with Darwin's fairness algorithm," July 2002, presentation at IEEE 802.17 Meeting. [Online]. Available: http://grouper.ieee.org/groups/802/17/documents/presentations/mar2002

[15] F. Davik, A. Kvalbein, and S. Gjessing, "Congestion domain boundaries in Resilient Packet Rings," Simula Research Laboratory, Tech. Rep. 2005-03, February 2005. [Online]. Available: http://www.simula.no

[16] D. Wang, K. Ramakrishnan, C. Kalmanek, R. Doverspike, and A. Smiljanic, "Congestion control in Resilient Packet Rings," in *Proceedings of the 12th IEEE International Conference on Network Protocols, 2004. ICNP 2004*, October 5-8 2004, pp. 108–117.

[17] H. Kong, N. Ge, F. Ruan, and C. Feng, "Congestion control algorithm for Resilient Packet Ring," *Tsinghua Science and Technology*, vol. 8, no. 2, April 2003.

Paper IV

Congestion Domain Boundaries in Resilient Packet Rings

Fredrik Davik, Amund Kvalbein and Stein Gjessing

Published: The most important results of this paper (technical report) have have been accepted for publication in Proceedings of the 48th annual IEEE Global Telecommunications Conference (GLOBECOM 2005). The accepted paper also includes the most important findings from paper III.

In the combined conference version, the main problems addressed by Paper III and Paper IV are discussed, along with proposed solutions and a short presentation of their performance.

Evaluation: We do not have the conference statistics from 2005. In 2004 however, 2086 papers were submitted to the conference, of which 792 were accepted, divided over 7 symposia. The conference version was reviewed by four reviewers.

Author Contribution: The problem-idea was found by Davik during the work with paper [76] and reported to the RPR Fairness Task force, responsible for the handling of problems related to RPR fairness.

The proposed modification of the introduction of the *passedTail* bit was made by Kvalbein.

All simulations and most of the associated discussions, except for Section IV.B were done by Davik.

The contribution from Gjessing is mainly related to the assurance of a clear and precise presentation of the main ideas and results obtained as well as general discussions on the problem.

Congestion Domain Boundaries in Resilient Packet Rings

February 2005
Revised August 2005

Fredrik Davik
Simula Research Laboratory
University of Oslo
Ericsson Research Norway
Email: bjornfd@simula.no

Amund Kvalbein
Simula Research Laboratory
Email: amundk@simula.no

Stein Gjessing
Simula Research Laboratory
Email: steing@simula.no

***Abstract*— In June 2004, the IEEE approved a new standard for Resilient Packet Rings (RPR). The standard is maintained in the 802 LAN/MAN Standards Committee, and is designated the standard number 802.17. In this paper, we analyze and discuss performance aspects of the Resilient Packet Ring fairness mechanism. We explain that, if the ring is not configured correctly, the fairness mechanism fails to stabilize at a fair division of bandwidth between the active nodes. We present a novel addition to the fairness algorithm, and show that with this modification, RPR reaches a stable state with more optimal parameter settings. We also show that our proposed modification gives shorter convergence time for the Resilient Packet Ring fairness algorithm.**

Keywords: Resilient Packet Ring, Fairness, Simulations

I. Introduction

Resilient Packet Ring (RPR) is a recent IEEE standard (802.17) that uses a dual ring topology to allow for efficient and resilient communication between the nodes connected to the ring [1]. Ring topologies have long been used to connect computers. The first widely known ring network was the Cambridge Ring [2], and we have later seen many different technologies in use, such as IEEE 802.5 Token Ring [3], FDDI [4], SCI [5], MetaRing [6], ATMR [7] and CRMA-II [8], [9]. Depending on the application area of the ring, the choice of medium access control (*MAC*) protocol to use for the ring is crucial. On a ring, where the demand for bandwidth is greater than the supply, and the data-flows from the contending nodes are of equal importance, the challenge becomes to equally or fairly divide the available bandwidth between the contending nodes. The method or algorithm used to solve this problem, is often referred to as a *fairness* algorithm.

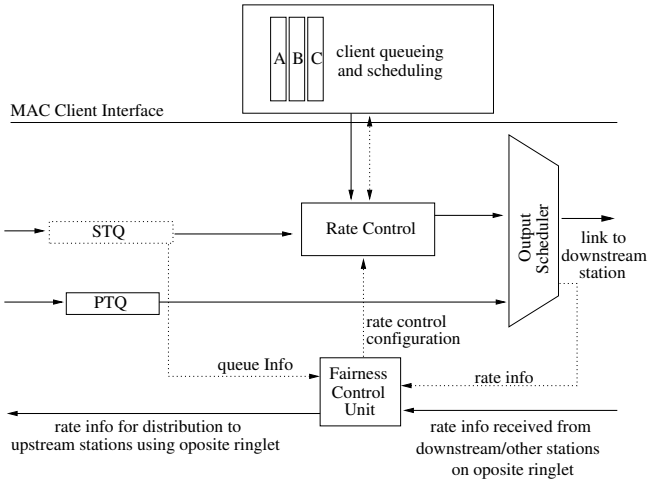
Some ring technologies, like Token Ring and FDDI, use a *token based* access control mechanism. In this class of networks, only the current owner of the token is allowed to transmit data on the ring. Fairness is enforced by limiting the time each node can own the token. Other ring technologies, including SCI, MetaRing, CRMA-II and RPR, use a mechanism known as an *insertion buffer* to control media access [10], [11]. In such insertion buffer rings, the buffer is used to store transit traffic that is held back when a node adds local traffic to the ring. *Slotted* ring technologies, like the Cambridge Ring and ATMR, avoid collisions by only transmitting data in given slots, that circulate the ring. In slotted and buffer insertion rings, various fairness mechanisms are used (see [12] for an overview).

RPR's fairness algorithm has two modes of operation, respectively the *conservative* mode, discussed in [13], and the *aggressive* mode, discussed in [13], [14]. In both modes, a node, with a congested out-link, maintains a local fair rate estimate which it distributes upstream, by use of so called *fairness* messages. Regardless of the mode used, the goal is to arrive at a fair division of sending rates, as defined by the "Ring Ingress Aggregated with Spatial reuse" (RIAS) reference model [15].

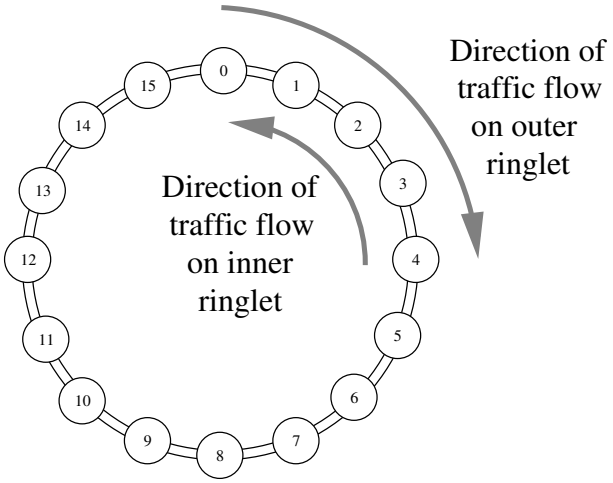
In the *conservative* mode of operation, the congested node transmits a fairness message, and then waits to see the change in traffic from upstream nodes. If the observed effect is not the fair division of rates, then the congested node calculates a new fair rate estimate, and distributes it upstream. The time to wait from a fairness message with a new fair rate estimate is issued, until the effect is evaluated, is based on periodic measurements, and is denoted the *Fairness Round Trip Time* (*FRTT*).

For the *aggressive* mode of operation, the congested node also calculates and advertises a fair rate estimate, but does so periodically, without waiting to evaluate the result of the previously transmitted fairness messages. In the aggressive mode, the calculation of the fair rate estimate is based solely on the node's own send rate and preset parameters. The frequent calculation and advertisement of new fair rate estimates gives rise to a more "aggressive" and opportunistic algorithm, that more quickly attempts to adapt to changing load conditions. Thus, the faster response as compared to the conservative version, comes at a cost. Namely the risk of instabilities when rate adjustments are made faster than the system is able to respond.

In this paper we discuss and analyze a performance deficiency of the RPR fairness algorithm, which relates to the handling and distribution of fairness messages, by nodes upstream of the congestion point. We propose a novel method



a) Generic node design for control of data flowing in one direction on an RPR ring



b) Example RPR topology

Figure 1: The generic RPR node design in a), shows a node's attachment to the ring for transferral of data in the downstream direction and transferral of fairness messages (fair rate estimates) in the upstream direction. The solid lines indicate the flow of data through the node. The dotted lines indicate the exchange of control/configuration information between node internal function blocks. To control the sending of data on both ringlets, every node has two instances of the functional blocks and interconnections below the MAC client interface. I.e. in the example topology in b) every node has one instance controlling the data flowing on the outer ringlet and another instance controlling data flowing on the inner ringlet.

that allows a more optimal setting of one of the most important parameters, the so called *lpCoef* parameter. We use performance evaluation by simulations to show that for both modes of the fairness algorithm, our method reduces the risk of network instability as well as speeding up the convergence process. We also show that minimal modifications are required to incorporate the proposed modification into the next version of the standard.

The rest of this paper is organized as follows. In section II, we outline and discuss parts of the RPR fairness mechanism, and identify the deficiencies we address in this paper. In section III, we discuss the propagation of fairness messages in an RPR network, and how this affects the network stability. Then, in section IV, we present our proposed improvement. In section V, we describe simulation scenarios and results used to illustrate and evaluate the performance of our improved algorithm. Finally, in sections VI, VII and VIII we address related work, conclude and point out some directions for future work.

II. Resilient Packet Ring Fairness Characteristics

Traffic in an RPR network is sent in one of three service (priority) classes, named A, B and C. The high priority, class A, traffic and a configured amount of the medium priority, class B, traffic is sent using reserved bandwidth. The rest of the class B and the low-priority, class C, traffic must compete for the remaining unreserved bandwidth. This traffic is known as *fairness eligible*, since the send rate for this traffic is governed by the fairness algorithm.

An RPR node may contain one or two insertion buffers, also known as transit buffers. In this paper, we consider only the dual transit buffer design (*2TB*). In the *2TB* design, transit traffic awaiting transmission of local traffic, is stored in one of the two transit buffers, depending on the service class of the packet. Packets belonging to the high priority class, are stored in the primary transit queue (PTQ), while other traffic is stored in the secondary transit queue (STQ). An overview of a generic RPR node design is given in Fig. 1.

A. Fairness algorithm parameters

In the RPR fairness algorithm, a node with a congested outlink calculates an estimate of the fair division of send rates (for fairness eligible traffic) traversing the congested link. The calculation is done at periodic intervals (every *aging* interval). According to the standard, the aging interval is 100μs (for line rates $\geq$ 622Mbit/s). The fair rate estimate is sent upstream in a fairness messages. Nodes upstream of the head, having received this messages, will limit their sending rate over the congested link to the fair rate estimate contained in the fairness message.

1) Aggressive Mode Fair Rate Estimation: For the aggressive mode of the fairness algorithm, the fair rate estimate is the value of a rate counter called *lpAddRate* [14].

lpAddRate is calculated by smoothing the local add rate, (*addRate*), using a two-stage second-order low-pass filter. After aging interval *n*, *lpAddRate* is calculated based on its previous value (that was calculated after aging interval *n-1*), and the number of bytes, *addRate(n)*, added by the node during the current aging interval[1]. This is shown in (1) below:

[1] *addRate* is really already slightly smoothed, but this has no effect on the current discussion.

$$lpAddRate(n) =$$

$$\frac{1}{lpCoef}\left(lpAddRate(n-1)\cdot(lpCoef-1)+addRate(n)\right)$$

$$\text{Where: } lpCoef \in \{16, 32, 64, 128, 256, 512\} \tag{1}$$

Observe that the relative weight of the previous value of *lpAddRate*, compared to the relative weight of the current value of *addRate*, is decided by the setting of the *lpCoef* parameter. Since *lpAddRate* becomes the advertised fair rate estimate, the *lpCoef* parameter decides how fast the congested node tracks and signals its local rate reductions to upstream nodes, when congestion occurs or increases. Correspondingly, *lpCoef* determines how fast the congested node tracks and signals local rate increases to upstream nodes when congestion ceases or decreases.

Intuitively, faster tracking of the local add rate, distributed to upstream nodes using fairness messages, should more rapidly resolve the congestion, resulting in faster convergence to the fair rate. Thus, one may believe that the *lpCoef* parameter should be set as low as possible. However, as discussed in [14], setting the *lpCoef* parameter too low, meaning that rate adjustments are performed too fast, will result in a system that oscillates, and fails to converge to the fair division of bandwidth over the congested link. The optimal setting of the *lpCoef* parameter is discussed in section V-B below.

2) Conservative Mode Fair Rate Estimation: For the conservative mode of the fairness algorithm, the fair rate estimate is the value of a variable termed *allowedRate*. The value of this variable is adjusted every Fairness Round Trip Time (FRTT), based on the occupancy of the *STQ*. The value of FRTT is measured periodically by use of special control messages, and is an estimate of the time it takes from a rate regulation is done by the congested node, until the corresponding effect is observable by the same node. By having an estimate of the time it takes from an adjustment is made, until the effect is observable, we effectively have an estimate of the system time-constant. This is then used, to ensure that rate-regulations are not made too fast, regardless of the setting of the lpCoef parameter.

B. Fairness convergence time

We define the *convergence time,* T_c as the time from congestion occurs, until the fairness algorithm has converged to the RIAS fair sharing of bandwidth for the congested link. A feedback control system is defined to be stable at time t_0, if all values for the controlled variable, sampled in the interval $<t_0,\ t_0+t_s>$, are within $\pm p\%$ of the mean value in the same interval [16]:

$$max(X(t)) < \overline{X(t)} * (1 + p/100), \quad t_0 < t < t_0 + t_s$$
$$min(X(t)) > \overline{X(t)} * (1 - p/100), \quad t_0 < t < t_0 + t_s$$

Later in this paper, we will show that the convergence time, T_c, is dependent on the number of and distance between the nodes contributing to the congestion, and the settings of the parameters used in the fairness mechanism.

C. Congestion Domains

Both modes (*aggressive* and *conservative*) of the RPR fairness algorithm works with a concept known as a *congestion domain*. A congestion domain defines a consecutive collection of nodes, of which some or all contribute to a congestion situation for a given link.

The congestion domain is confined within a region specified by two boundary nodes. At one end of the region resides a node, denoted the *head*, which is attached upstream of the most congested link in the region. At the opposite end of the region resides a node, denoted the *tail*. Nodes upstream of the *tail* are considered as not being contributors to the congestion situation at the head.

The declaration of a congestion domain can be considered a two-part problem. The first part of the problem consist of making a node declare itself as the head, responsible for calculating the (RIAS) fair division of unreserved bandwidth among contending nodes sending fairness-eligible traffic over the bottleneck link. The calculated result, the fair rate estimate, is distributed to upstream nodes in the congestion domain using fairness messages.

The second part of the problem consists of making a node declare itself as tail, responsible for stopping the propagation of fairness messages, received from the head. In the RPR standard, the appointment of a tail node can be done for two reasons (denoted TA, as a shorthand for Tail Appointment Condition). For both cases, we assume the node is aware of the presence of a downstream head:

TA 1: When a node finds itself more congested than the downstream head. In this case, this node becomes the tail of the congestion domain that starts at the downstream head. Additionally, it becomes the head of a new congestion domain that extends from this node and upstream.

TA 2: When a node, based on measurements of traffic rates from upstream nodes, decides that the aggregate of fairness eligible traffic from upstream nodes, traversing the congested link, does not contribute to the downstream congestion.

The self-appointment of a tail node according to the rule specified in TA1 appears problem-free. The self-appointment of a tail node according to the rule specified in TA2, however, has shown to degrade the network performance for some scenarios (see section V). In the next section, we will discuss how the self-appointment of a tail node according to TA2 affects the stability of the RPR fairness algorithm.

III. Rate Distribution in Fairness Domains

The RPR standard states that when the aggregate of fairness eligible traffic received from upstream nodes does not exceed the fair rate, the propagation of (fair) rate information further

upstream is not needed. The rationale for this reasoning, is that since the aggregate of traffic from upstream nodes is less than the fair rate, the distribution of the fair rate information further upstream would have no effect. This can be considered a reasonable assumption as none of the upstream nodes send at a rate exceeding the fair rate.

In this section, we will argue that this assumption is incorrect and furthermore, when adhered to, possible side-effects are unfairness and network instability. Below, we describe the origin of the problem and discuss how it degrades the performance of the RPR fairness algorithm.

The problem may occur when the equilibrium point of the aggregate of traffic received from upstream nodes is close to that of the fair rate estimate. We will illustrate this by use of the simple topology shown in Fig. 2. Assume that nodes B, C and D all send over the same congested link, and are part of the congestion domain spanning from D (head) to B (tail). Next, node A starts sending over the same congested link.

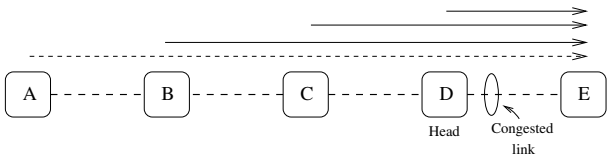


Figure 2: Nodes B, C and D are part of a congestion domain. When node A starts sending across the congested link, the congestion domain is extended to include A.

Initially, since it has no notion in advance of the downstream congestion, node A starts sending at the full capacity of its output link. For a short while, this will cause the sending rates of nodes B, C and D to drop significantly.

Once B's rate measurements of traffic received from A has been affected enough, it does not fulfill any of the tail-appointment conditions. Thus B starts to forward the fairness messages received from the head (D) upstream. By this, B is effectively transferring the role as tail of the congestion domain to A.

Once A receives fairness messages from the head, it will reduce its sending rate to that of the encapsulated fair rate estimate in the fairness message. If at some future time, the *STQ* occupancy of the head falls below a threshold in response to the aggregate rate reductions by its upstream neighbors, the head will gradually start to increase its fair rate estimates. If the delay between the time where B starts receiving increasing fair rate estimates and the time where the furthest upstream node, A, starts to increase is send rate is too large. B will, once again assume tail responsibility according to TA2 and stop the propagation of fairness messages received from the head.

In summary, for the duration of the periods where the propagation of fair rate information to upstream node(s), having a demand that equals or exceeds their fair share, is stopped. This may result in excessive sending and resulting unfairness. Furthermore, this may result in non-convergence of the fairness algorithm. In sections III-A and III-B, we will discuss the effects of this behavior for the aggressive and conservative fairness modes. Then, in section IV, we propose a modification to the fairness algorithm, to resolve these problems.

A. Congestion Domains and Aggressive Mode Fairness

For the *aggressive* mode fairness algorithm, the decrease in the head's amount of added traffic to the ring results in a corresponding decrease in the head's *lpAddRate* counter, which is distributed to the upstream neighbors in the fairness messages. If the duration of the bursts are too long compared to the *lpCoef* setting, the head's *lpAddRate* counter will decrease too much, too fast, and at the end of a cycle, the starting point of the next cycle will be no closer to the theoretical fair division of add rates than the starting point of the previous. Hence there will be no (further) convergence to the fair rate, and the system remains unstable.

Because of the effect described above, there is a connection between the congestion domain size and the minimum setting of the *lpCoef* parameter. In the worst case, a congestion domain spans from node N-1 to node 0 in an N-node ring. To guarantee convergence of the fairness algorithm, the *lpCoef* parameter must be set so that *i)* the head does not make rate adjustments faster than it is able to evaluate the results (at least partially) of the corresponding upstream rate adjustments [14] and *ii)* the effect of tail bursts transiting the head does not affect the value of its *lpAddRate* counter too strongly.

The behavior of the rate-control algorithm executing in the head consists of an infinite series of adjustment cycles. Each cycle consists of two periods. We will illustrate this behavior using the simple scenario shown in Fig. 2 and using two different values of the per link propagation delay.

In the scenario above, let us consider the point shortly after node A has assumed the role as the tail of the congestion domain. In response to the increase in transit traffic, resulting in an increasing *STQ* occupancy, the head will stop the sending of local traffic (as long as the *STQ* occupancy exceeds the high threshold). As a result, the head's fair rate estimate, *lpAddRate*, will decrease monotonically. Further, the upstream nodes will all, upon reception of the fairness messages, reduce their addRate.

As a result, at some future time, the occupancy of the head's *STQ* will change from being higher than the *high-* to being lower than the same threshold. When this happens, the head will start to add fairness eligible traffic over the congested link again. Hence, the value of its *lpAddRate* rate counter starts to increase. Finally, once the *STQ* occupancy, because of the resulting increase in transit traffic, exceeds the high threshold, we are back to the starting point.

Below, in figures 3a, 3b and 3c, this behavior is illustrated, using aggressive mode fairness and plotting rate measurement statistics from node *B* for the scenario. We use a value of 16 for lpCoef and link-lengths of 50 and 250μs. All stations start to send traffic at time 1.1s. The figures plots data for three different data sets. The first is the fair rate estimates, labelled "receivedRate", received by node *B* from

the head. The second, labelled "normLpFwRateCongested", is the aggregate of transit traffic, received by node *B* from node *A*. The third, labelled "rate value sent upstream", is the rate value of the fairness messages, sent upstream from node B.

As seen in figures 3a and 3b, between times t_1 and t_2, the rate information received by node *B* consist of a sequence of monotonically increasing values, where the value received at t_1 is the minimum value in the sequence and the value received at t_2 is the maximum value in the sequence. The monotonically increasing sequence is ended once the head discovers that its STQ occupancy has exceeded the *high* threshold. In Fig. 3a, the rate value calculated in response to this, is the one received by *B*, at time t_2.

Following this, we have a sequence of monotonically decreasing rate messages received in the interval $\langle t_2, t_3]$. The monotonically decreasing sequence is ended once the head discovers that its STQ occupancy has fallen below the *high* threshold. In Fig. 3a, the rate value calculated in response to this, is the one received by *B*, at time t_3.

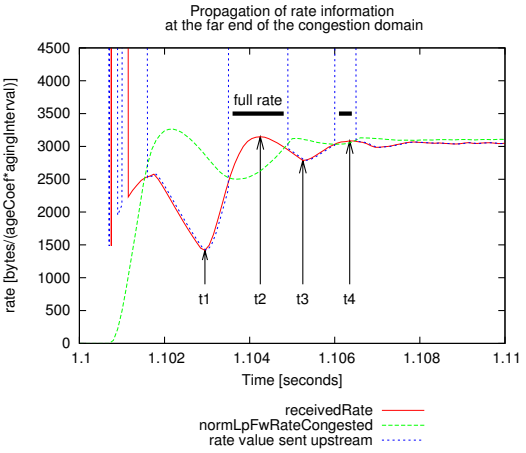
At point t_3 we have reached the end-point of the first cycle and the start point of the next cycle immediately follows. In Fig. 3a, plotting the rate statistics for a link-length of 50μs, we observe that the difference between the max and min value in each cycle decreases towards 0 for each consecutive cycle. I.e. the fairness algorithm converges. In figures 3b and 3c, plotting the rate statistics for a link-length of 250μs, this does not happen. I.e. the fairness algorithm does not converge. During the first few cycles ($t < t_8$) (see Fig. 3c), the difference between the max and min values in each cycle does decrease. After this point however, the difference between the max and min values converges towards a mean value of 1621$\left[\frac{bytes}{ageCoef \cdot agingIntervals}\right]$ with a standard deviation of 1% of the mean. The statistics properties when observing the magnitude of the oscillations in the dataset, are shown in table I below. The analysis is performed from time t_8 (marked in Fig. 3c). Note that the figure shows only a subset of the dataset analyzed, however the oscillatory behavior for the remainder of the period is the same.

[s]		[#]	$\left[\frac{bytes}{ageCoef \cdot agingInterval}\right]$				
$\mathbf{t_{start}}$	$\mathbf{t_{stop}}$	**n**	**max**	**min**	**mean**	**median**	**stdev**
1.11965	1.39855	135	1659	1582	1621	1619	16.90

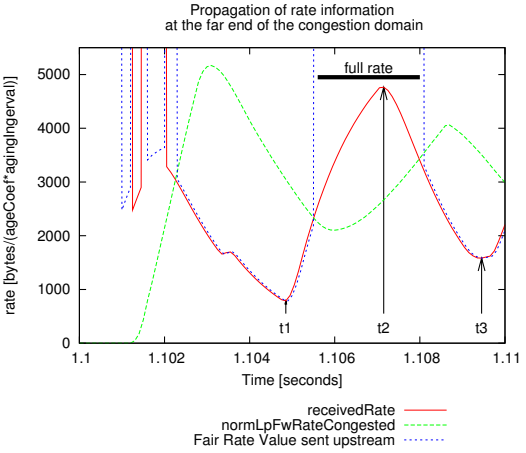
Table I: Magnitude of oscillations for the unstable configuration.

If we investigate the plots closer, starting with Fig. 3a, we can observe the following: At the point between t_1 and t_2, where *receivedRate* increases beyond *normLpFwRateCongested*, node *B* decides that the aggregate of traffic received from upstream nodes no longer contribute to the downstream congestion. Thus, in accordance with TA2, node *B* assumes the role as tail and issues a *full*[2]-rate rate message upstream (the value of rate messages sent upstream is the plot line labelled

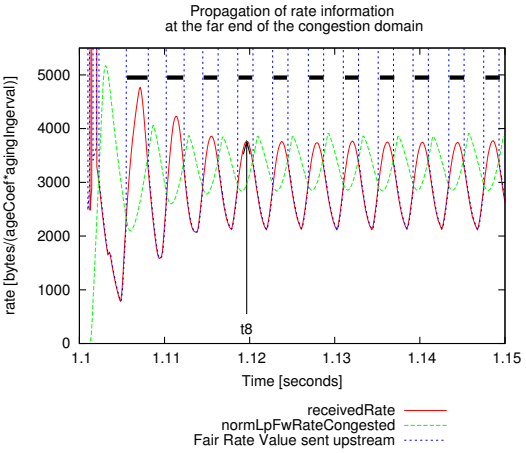
[2]The semantics of the *full*-rate message is that nodes receiving this message no longer have to restrict their local add rates, regardless of the destination of their traffic.



(a) Link-length of 50μs and lpCoef=16, fair rate calculation process converges.



(b) Link-length of 250μs and lpCoef=16. First cycle of fair rate calculation process illustrated for non-converging configuration.



(c) 250 μs and lpCoef=16. Multiple cycles of fair rate calculation process illustrated for non-converging configuration.

Figure 3: Traffic measurements performed at node B. The horizontal lines shows the period where node B issues full-rate fairness messages upstream.

Fair Rate Value sent upstream). The duration of the periods where node B sends *full*-rate messages upstream is indicated by the horizontal lines in figures 3a, 3b and 3c.

The time where node *B* transmits the *full*-rate message upstream, is represented by the vertical line which starts at $t = 1.1035$. One link-propagation time later, when this *full*-rate message is received by node *A*, the effect is that *A* will gradually (every aging interval) increase its amount of added traffic towards the maximum rate. The effect of the increased amount of traffic added from *A* will at the earliest be detected by node *B* two link-propagation delay times after the *full*-rate message was transmitted upstream from *B*.

As a result, *B*'s measurements of traffic from *A*, *normLpFwRateCongested*, will increase beyond the fair rate estimate received from the head, *receivedRate*. At the time between t_2 and t_3, when the value of *normLpFwRateCongested* increases beyond *receivedRate*, node *B* decides, in accordance with TAs 1 and 2, that *A* **does** contribute to the downstream congestion. Thus, it transfers the tail responsibility back to *A*, by passing on the received fairness messages originating from the head. The first of the fairness messages forwarded by *B* at this time, will be received by *A* one link propagation time later. Thus, at the time when *B* detects that its upstream neighbor(s) do indeed contribute to the congestion downstream, they will be allowed to transmit at an excessive rate for an additional period of at least as long as it takes the fairness message to propagate to the upstream active node(s). Thus, the longer the links, the longer the duration of the excessive transmission.

If we investigate Fig. 3c, showing the rate measurements for the unstable configuration, we see that the duration of the periods, where node *A* is allowed to gradually increase its add rate, does decrease slightly initially (while $t < t_8$). Later, the duration of these periods stays relatively constant. As shown in table II, during the observation period, the rise-time (of the fair rate estimate received from the head) from a local minimum- to a local maximum value is on average 1.86 ms. The fall-time from a local maximum- to a local minimum value is on average 2.27 ms.

	[s]		[#]	[ms]			[μs]
flank	$\mathbf{t_{start}}$	$\mathbf{t_{stop}}$	**n**	**max**	**min**	**mean**	**stdev**
Rise	1.11965	1.39855	67	1.90	1.80	1.86	49
Fall	1.11965	1.39855	68	2.30	2.20	2.27	47

Table II: Time between local extremum points at rate plots for unstable configuration.

B. Congestion Domains and Conservative Mode Fairness

The problems described for the aggressive mode fairness algorithm also apply to the conservative mode fairness algorithm. For the conservative mode algorithm however, the rate adjustments are performed every Fairness Round Trip Time as discussed in section II-A.2. Thus as node A in Fig. 2 is alternately included/excluded into/from the congestion domain, the value of FRTT is correspondingly adjusted. However, the adjustment of FRTT takes time, thus when the difference between the system-time constant between the two congestion domain alternatives become too large, this affects the convergence of the fair rate estimation process.

IV. Propagation of Fairness Messages Beyond Tail

At least two different approaches can be imagined to avoid the instabilities discussed above. One possibility would be to create a mechanism that prevents node B in the above example from taking on the role as congestion tail before the traffic from upstream nodes has been below the fair rate for a given amount of time. This would demand a timer to be set once the conditions for being a tail are met. Only when this timer expires, would the new tail stop propagating fairness messages received from the head to upstream nodes. Such a timer would, however, probably contribute to a longer convergence time for the fairness mechanism in many cases.

Instead, we propose a mechanism that alters the responsibility of the tail of a congestion domain. With our modification, the propagation of fairness messages is not stopped by the tail. The rationale behind this is that nodes that do not send traffic over the congested link, are not affected by the received rate limitations. However, nodes that *do* send across the congested link will always receive the fairness messages, containing information on the closest downstream congestion, even if they temporarily send below the allowed rate. By this, nodes upstream of the tail will limit their sending-rate over the congested link, to that of the received fair rate estimates. This way, the oscillations otherwise experienced are avoided.

One way of implementing this, could be to propagate the fair rate estimates unconditionally beyond the congestion tail. This would cause the fair rate estimate of the most congested link to be propagated all the way around the ring, effectively allowing only one congestion domain. This solution however, appears to be in conflict with the RPR standard, as the RPR standard allows the existence of several independent congestion domains on an RPR ring.

What we propose, is to use one bit in the fairness frames to mark the frame before being forwarded upstream by a congestion domain tail. The RPR fairness frame format contains 13 reserved and currently unused bits. We propose using one of these as a *passedTail* bit, indicating that the fairness frame contains a fair rate estimate that has been propagated beyond a congestion tail. With our modification, the role of a congestion tail is no longer to stop the propagation of the fairness message received from the head, but instead to set the *passedTail* bit of forwarded fairness messages to one. With our modification, the fair rate estimate calculated at one congestion head is not terminated before it reaches the head of the next congestion domain. If there is only one congested link on the ring, the fair rate estimate calculated by the node immediately upstream of the congested link will be propagated all the way around the ring. If there are several congested links on the ring, the fairness message will propagate upstream from the head of one congestion domain, until it reaches the head of the next congestion domain. Thus effectively allowing the existence of several congestion domains on a ring. For the remainder of this article, we will refer to this mechanism as the *tail-fix*.

Fig. 4 shows the difference between the original and our modified mechanism. In the figure, we have two congestion domains A and B, spanning from node 3 to 1 and node N to N-3 respectively. With the original RPR fairness algorithm, the tails of the respective congestion domains (nodes 1 and N-3), will terminate the propagation of the fairness message received from the downstream head. Thus, they will advertise *full*-rate to the upstream nodes. With our proposed mechanism, the tail nodes instead propagate the fairness messages received from their respective heads, after setting the *passedTail* bit to one. The fair rate estimate from node N is not terminated before it reaches the head of the next congestion domain (node 3).

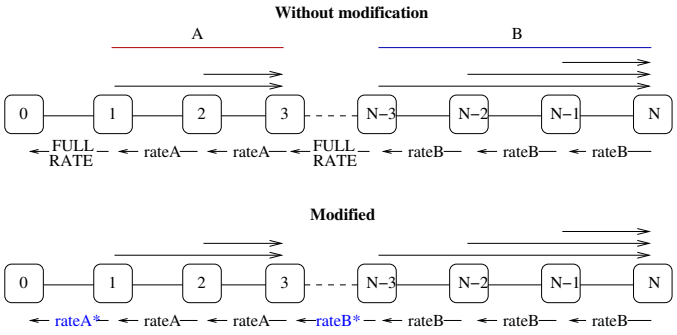


Figure 4: With our proposed solution, fairness messages from one congestion domain are not terminated before they reach the head of the next domain.

V. Simulation Scenarios and Results

In this section, we describe simulations made to evaluate our modified method. The experiment described in section V-B was run on our simulator written in the Java programming language, within the J-Sim [17] simulation framework. For the remaining experiments, we have used our simulator model implemented within the OPNET [18] simulation framework.

A. Fair Rate Propagation Beyond Congestion Tail

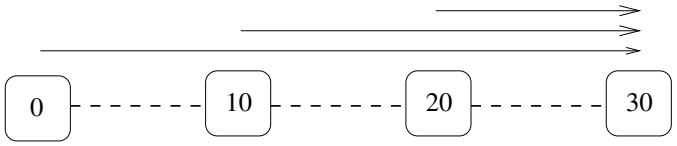
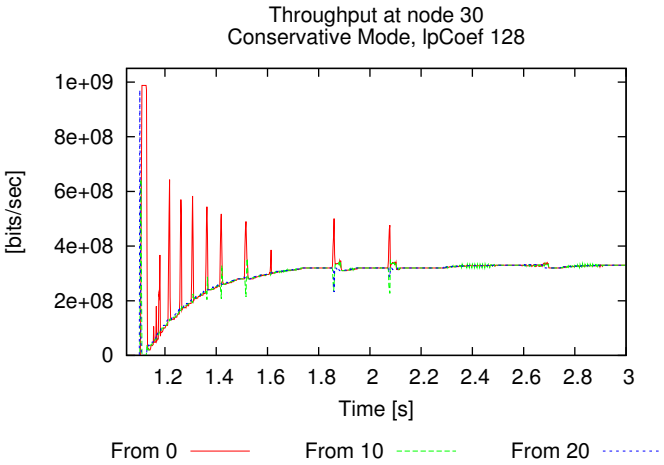


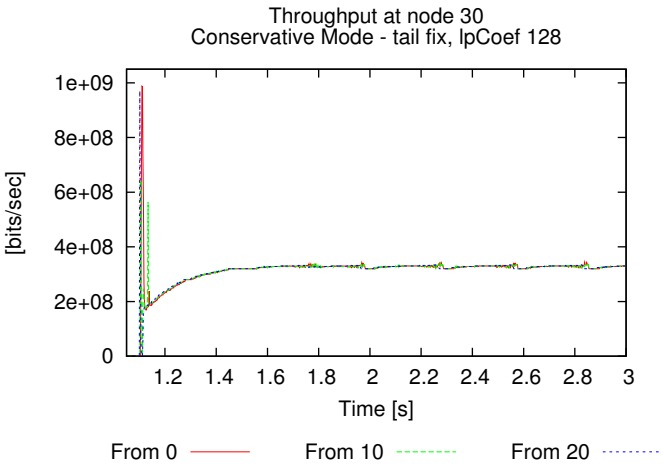
Figure 5: In this scenario, nodes 0, 10 and 20 send at their maximum allowed rate to node 30.

As discussed above, the Resilient Packet Ring fairness algorithm does not converge if the ring size is too large compared to the parameter settings, notably the *lpCoef* parameter. In this section a simulation scenario illustrating this behavior is presented. It shows that propagation of the fair rate estimate beyond the congestion tail allows the fairness algorithm to converge with a lower value of the *lpCoef* parameter than would otherwise be needed.

In this scenario, we have a 64 node ring with 40 km links. The link capacity is 1 Gbit/s. Nodes 0, 10 and 20 send class C traffic at their maximum allowed rate to node 30, making node 20 the congestion head and node 0 the congestion tail. They all start sending simultaneously, at time 0.1 s. The scenario is illustrated in Fig. 5.



(a) With the original conservative mode fairness and lpCoef=128 for the scenario shown in Fig. 5, the fair rate estimation process does converge. However, there are some brief periods of unfairness, where the furthest upstream node (node 0) gets to send excessive amounts of traffic.



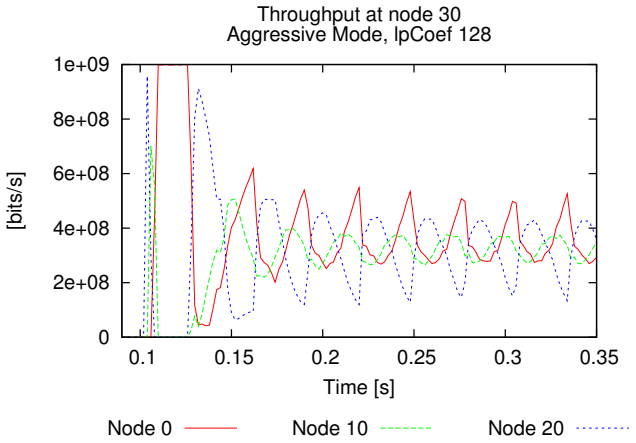
(b) With the conservative mode fairness with the tail-fix implemented and lpCoef=128 for the scenario shown in Fig. 5, the fair rate estimation process converges faster and without any unfairness.

Figure 6: Fairness convergence for the conservative fairness mode with and without the tail-fix.

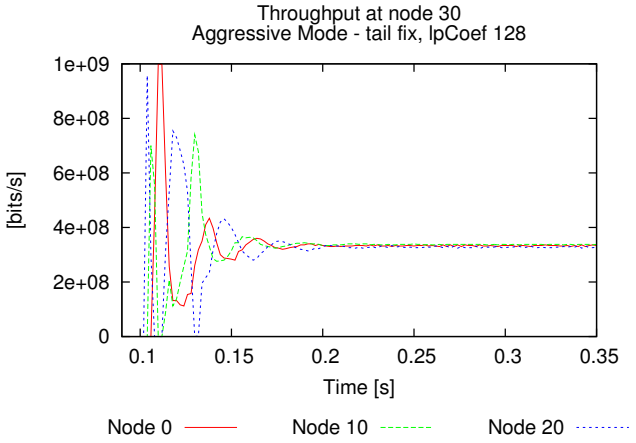
Figures 6a and 6b shows the convergence towards the fair sharing of bandwidth, by measuring the throughput of traffic received by node 30. We use a value for *lpCoef* of 128 and conservative mode fairness. As shown, the algorithm converges with or without the tail-fix. With the tail-fix however, the convergence time is reduced and excessive sending by the most upstream node is prevented.

In figures 7a-7c, we have the same set of measurements for the aggressive fairness mode. Fig. 7a shows the results for the original RPR standard implementation, while Fig. 7b shows the result when the fair rate estimate calculated at node 20 is

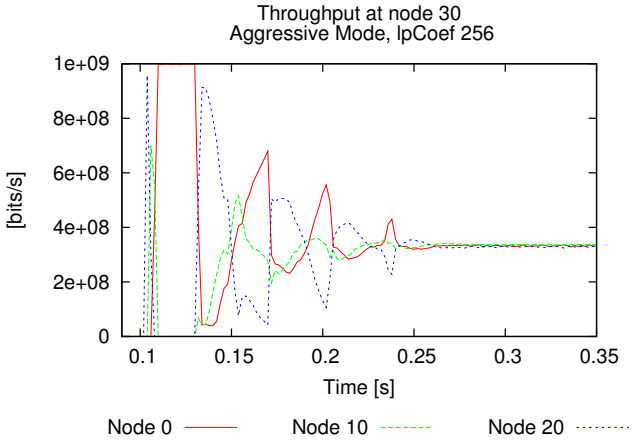
propagated beyond the congestion tail with the *passedTail* bit set. We see that with the original RPR implementation, the sending rate of each active node does not converge to the fair division of bandwidth ($fairRate = \frac{1Gbit/s}{3} = 333Mbit/s$), while with our modified method, the system converges after about 0.2s of simulated time. Fig. 7c shows that if the value of the *lpCoef* parameter is doubled (256), the original RPR fairness algorithm will also converge to a stable state, but the convergence time will be longer.



(a) With the original aggressive mode fairness and lpCoef=128, the fair rate estimation process fails to converge to the fair rate for the scenario described in Fig. 5 above.



(b) With the aggressive mode fairness with the tail-fix implemented and lpCoef=128, the fair rate estimation process converges to the fair rate for the scenario described in Fig. 5 above.



(c) With the original aggressive mode fairness and lpCoef=256, the fair rate estimation process converges to the fair rate, but as seen, the convergence time increases with approximately 30% when compared to aggressive mode with tail-fix implemented and lpCoef=128.

Figure 7: Fairness convergence for the aggressive fairness mode with and without the tail-fix.

B. Fairness Convergence

The simulation results described above, showed that our new mechanism improved the convergence time for the fairness algorithm. In the scenario described in this section, we investigate the relation between the size of a congestion domain, the setting of the *lpCoef* parameter, and the convergence time T_c as defined in section II-B. We use $t_s = 50ms$, a sampling interval of $2ms$, and $p = 8\%$. This relatively high setting of p is needed to allow small oscillations in a visually stable system.

We use a ring with 40 km links, with a capacity of 1 Gbit/s. There are three active nodes, node 0, node *i/2* and node *i*, that all send traffic to node 30 at their maximum allowed rate, as shown in Fig. 8. This makes node *i* the head of a congestion domain spanning from node *i* to node 0. *i* is varied from 2 to 28 with step 2, to adjust the size of the congestion domain. Note that the topology described in section V-A is a special case of this scenario, with i=20.

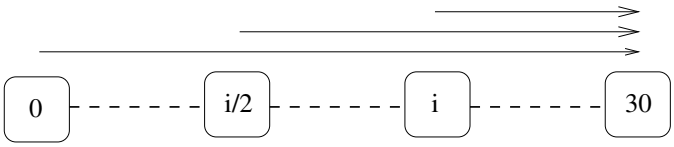


Figure 8: In this scenario, nodes 0, i/2 and i send at their maximum allowed rate to node 30.

Fig. 9 shows the convergence time T_c for different congestion domain sizes and *lpCoef* settings. Results are shown for the original RPR standard (Fig. 9a) and the modified version propagating the fair rate estimate beyond the congestion tail (Fig. 9b).

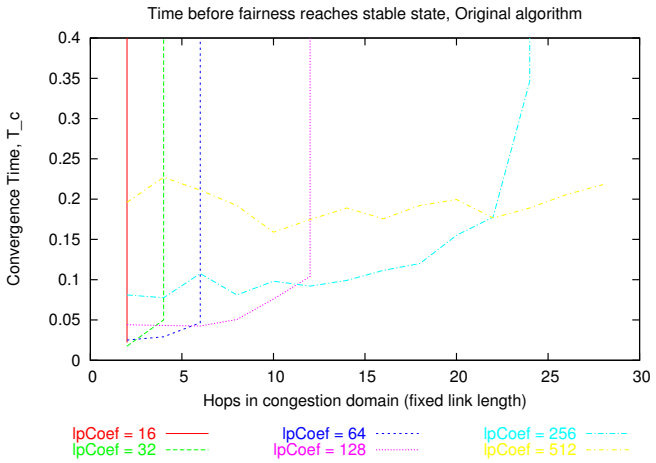
The plots in Fig. 9 show that the propagation of fairness messages beyond the congestion tail allows for a lower setting of the *lpCoef* parameter for a given congestion domain size. The simulations indicate that the value of the *lpCoef* parameter can be at least halved for a given ring size, while maintaining stability/convergence of the fairness algorithm.

We observe that for a given congestion domain size and *lpCoef* setting, our modified algorithm gives a shorter convergence time T_c than the original RPR fairness algorithm.

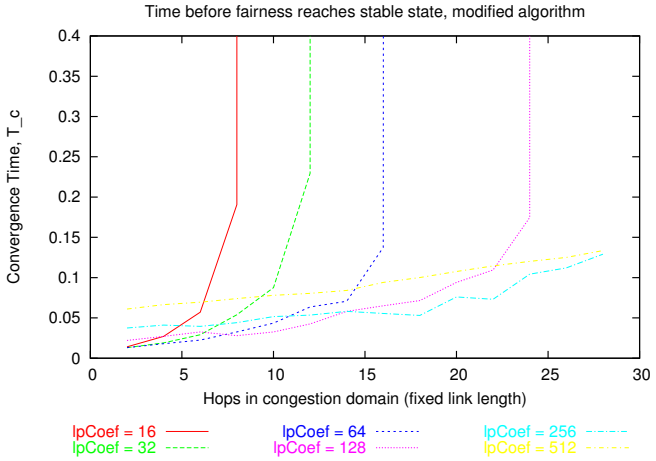
Finally, we see that when the congestion domain size approaches the maximum size for a given *lpCoef*, the increase in stabilization time T_c increases rapidly. Consequently, the optimal setting of the *lpCoef* parameter with respect to T_c, is not always the lowest possible value resulting in a stable system. However, in a dynamic network the traffic patterns and congestion domains will vary. In such an environment, we believe that a low value of the *lpCoef* parameter gives the best overall performance.

C. A General Congestion Scenario

In this section, we want to illustrate the behavior of our proposed modification for a general congestion scenario, where some senders are modest and some are greedy and there are some local flows that do not interfere for others. An example of a such scenario is shown in Fig. 10.



(a) Aggressive mode.



(b) Aggressive Mode with tail-fix.

Figure 9: The figure shows fairness convergence as a function of congestion domain size and lpCoef setting. Propagating the fair rate estimate beyond the congestion tail allows for a lower setting of the lpCoef parameter, giving reduced fairness convergence time. The link length is kept constant, so the propagation delay increases linearly with the hop count.

In the figure, the most congested link, is the outgoing link from node 40. There are 7 flows traversing this link, thus the RIAS fair rate (with all greedy senders) is 14.29% of the line-rate. However, as some of the nodes are modest, the RIAS fair rate will be $\frac{100-1-5-15}{4} = 19,75\%$ (the 1, 5 and 15% flow should all get their full demand, while the remaining senders must share the remaining available bandwidth).

Further, the local flows (8,12) and (21,24) should take whatever spare capacity is left on the respective outgoing links. Thus these flows should not interfere with the flows traversing the congested link. That is, other than causing packets transiting nodes 8 or 12 having to await the transmission of one packet, if the transit packet arrives once the transmission of a local packet has started.

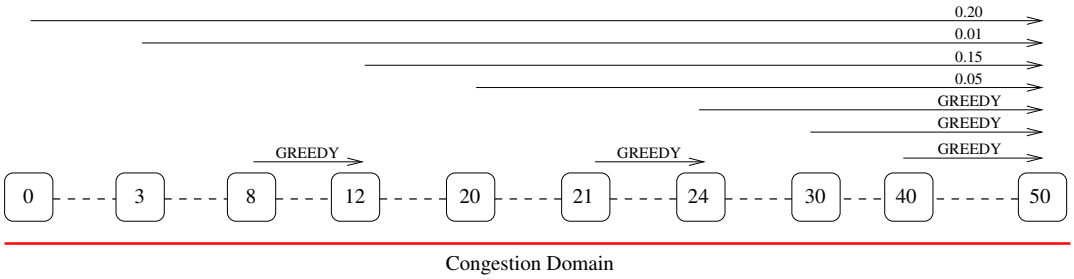
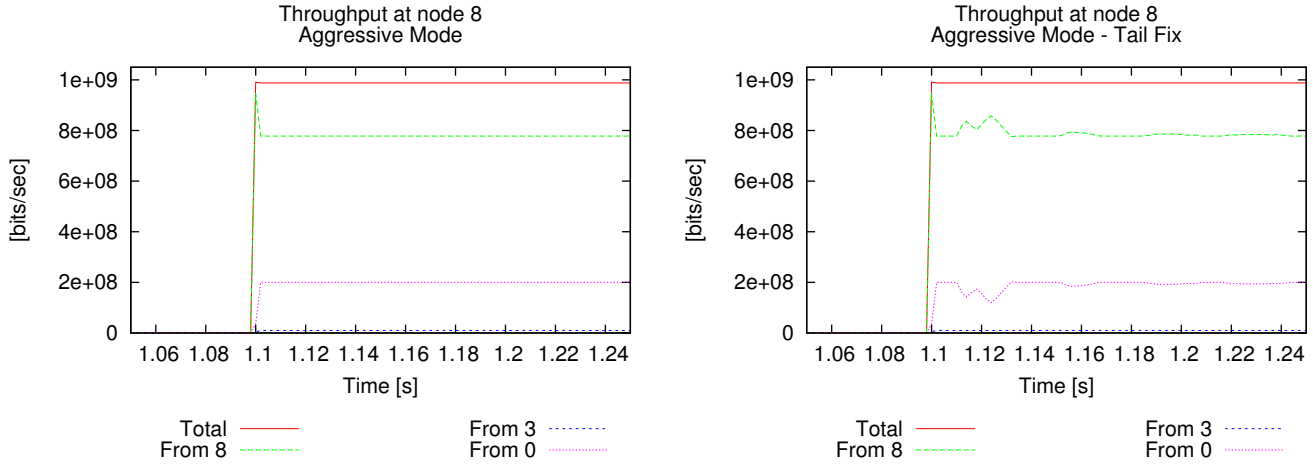


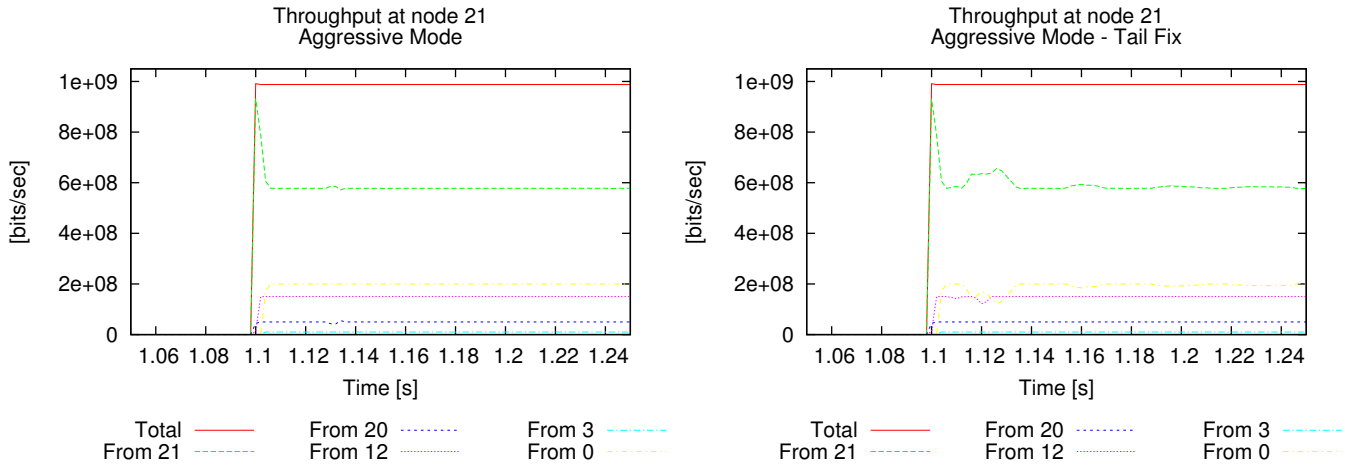
Figure 10: A general congestion scenario. Some senders are modest while others are greedy. Additionally, there are some local flows that do not interfere with any of the other flows.



(a) Original algorithm.

(b) Tail-fix.

Figure 11: Throughput measured at node 8 for aggressive mode fairness. With the tail-fix, the throughput of traffic from node 0 is throttled by the head during convergence to the fair rate, thus node 8 can send more local traffic during this period.



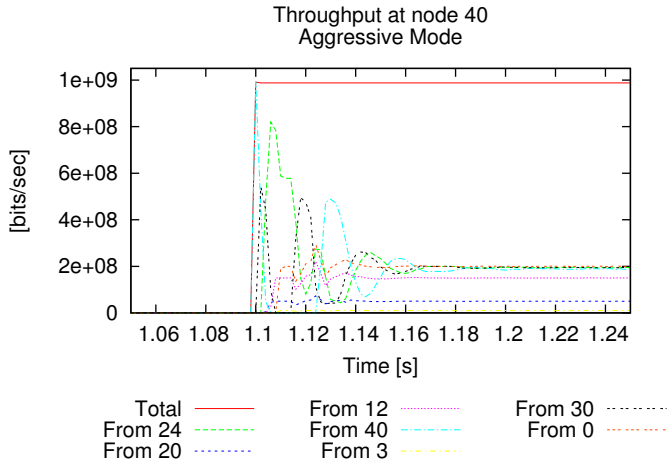
(a) Original algorithm.

(b) Tail-fix.

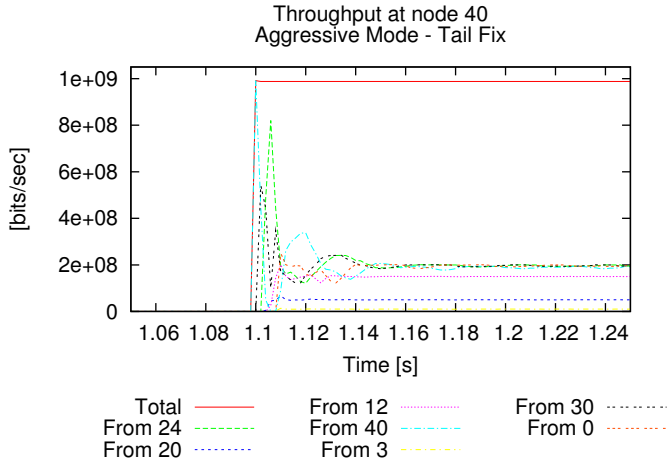
Figure 12: Throughput measured at node 21 with and without the tail-fix.With the tail-fix, the throughput of traffic from nodes 0 and 12 is throttled by the head during convergence to the fair rate, thus node 21 can send more local traffic during this period

We have collected throughput statistics for the scenario illustrated in Fig. 10 for both the original aggressive fairness mode as well as for the aggressive mode with the tail-fix. In figures 11a-13b, we show the throughput measured at nodes 8, 21 and 40 respectively, since, given the applied load, we should expect full link-utilization at these links.

As seen from the figures, the total link-utilization for these links is close to 100% both with and without the tail-fix. Thus our proposed modification does not degrade the link-utilization. The convergence towards the fair rate however



(a) Original algorithm.



(b) With the tail-fix, the fair rate estimation processes converges somewhat faster and with a lesser degree of unfairness.

Figure 13: Throughput measured at node 40 for aggressive mode fairness with and without tail-fix.

differs slightly for aggressive mode fairness and aggressive mode fairness using our proposed tail-fix.

As seen from figures 13a and 13b, the throughput convergence for flows transiting the head is somewhat faster when using our proposed tail-fix. The behavior at the head however has some impact on the convergence at the upstream nodes 8 and 21 which transmit local greedy flows. The fact that the head gradually allows upstream flows with a demand greater than the fair rate to increase their send rate, enables the local flows to utilize a greater share of the available bandwidth for local traffic (during the transient period). While for the original algorithm, the upstream greedy nodes send at a rate greater than their fair share (during the convergence period), thus the local traffic flows suffers from this and have to decrease their send-rate somewhat.

D. Multiple Congestion Domains

In sections V-A, V-B and V-C, we showed how our proposed modification improves the performance when we have (ingress aggregated) flows that are confined within one congestion domain. For multiple and independent congestion domains (i.e. no flows traverses multiple congestion points), we can expect the performance improvement to increase.

In this experiment, we want to evaluate the performance of the system when we have flows, that are not confined to one congestion domain. Fig. 14 shows such a scenario consisting of two congestion domains. The downstream congestion domain consist of the node set: $n \in [3..16]$ (node 3 is tail, while node 16 is head), while the upstream congestion domain consist of the node set: $n \in [0..3]$ (node 0 is tail, while node 3 is head).

The normalized RIAS fair shares for this scenario are shown in the figure. As seen, the downstream head is the most severely congested, having infinite demand flows traversing its outgoing link, each getting 20% of the available bandwidth. Two of these infinite demand flows are crossing the upstream head as well, thus the remaining available bandwidth available for other flows in the upstream congestion domain is limited to $100 - 2 \cdot 20 = 60\%$. Thus, each of the two remaining flows gets $60/2 = 30\%$.

However, the bandwidth shares as given by the RIAS reference model are theoretical values. For this general type of scenarios, where we have a downstream head, more severely congested than the upstream one, there will be a constant alternation between the existence of one and two congestion domains on the ring. A brief explanation of this behavior, in context of the given scenario, is given below.

Let us assume both congestion domains are active on the ring. The downstream one covering nodes 3-16 and the upstream one covering nodes 0-3.

As long as there are two congestion domains in effect, the flows from nodes 0-3 are rate restricted by the upstream head (node 3), thus the upstream head gradually allows the upstream active nodes to increase their send rate towards 25% of the line rate (the fair rate over the upstream congestion point). This will lead to an increasingly severe congestion situation at the downstream head (node 16). Thus the fair rate estimate from node 16 will be gradually lowered, until the point where node 3 no longer fulfills TA1 or TA2. At this point node 3 will no longer maintain its tail responsibility, thus the downstream congestion domain is extended to cover the whole region, covering nodes 0-16.

When this happens, only flows traversing node 16 are rate limited at their ingress point. Thus the aggregate of traffic from nodes 0 and 3 will increase towards the link-rate until the point where node 3 becomes more congested than node 16. At this point, node 3 fulfills TA1, and thus we are back to the starting point, having two congestion domains as shown in Fig. 14.

Thus we have an endless series of cycles, where the sending rates of the nodes are controlled by two different nodes, having two different theoretical target rates. Thus the scenario never converges to the RIAS fair division of rates.

This behavior is illustrated in figure 15. Plotting respectively the throughput and cumulative throughput of the different

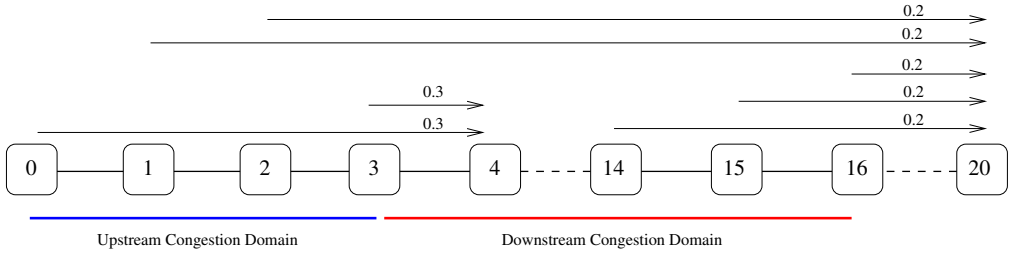


Figure 14: A scenario with (ingress aggregated) flows between two different congestion domains. All flows have infinite demand. The normalized bandwidth shares are the RIAS fair shares.

flows, measured at the upstream congestion point and the downstream receiver. We show the performance using aggressive mode fairness with and without our proposed tail-fix.

When looking at the throughout for the flows measured at both node 3 and node 20, as expected, the fairness algorithm does not converge to the fair rates. If we look at the throughput of the flows measured at node 20, shown in Fig. 15 a) and b). It is clear that for the original algorithm (15 a)), we have an initial period of unfairness, where the upstream nodes (1 and 2) are allowed to send more than their downstream neighbors. This is caused by TA2, where node 3 assumes tail responsibility and prevents nodes 1 and 2 from receiving rate information from the downstream head. With our proposed modification (15 b)), node 1 and 2 will still receive rate information from the downstream head (node 16). Thus the initial period of unfairness is prevented.

In the long term, all flows traversing the downstream head (node 16), should ideally receive 20% of the bandwidth each. However, as the flows from nodes 1 and 2 have to traverse two bottleneck links we should expect the long term throughput of these flows to be somewhat lower than that of the flows only traversing the downstream bottleneck. Fig. 15 c) (original algorithm) and d) (with our proposed improvement) illustrates this. In the figure, we show the cumulative throughput, aggregated over 4s of simulated time. In Tab. III, we show the actual numbers (represented by each flow's share of the total). As seen from the table, the cumulative throughput of flows 1 and 2 are on average approximately 2% lower than those of the downstream ones. For both cases, the total cumulative throughput is the same.

When looking at the cumulative throughput performance of the flows confined within the upstream congestion domain shown in Fig. 15 g) and h), (i.e. the flows from nodes 0 and 3 both going to node 4), we observe that we do not achieve RIAS fair sharing of the 60% bandwidth portion not used by the flows from nodes 2 and 3. In fact, as shown in figures and Table IV, the flow from node 0 receive more than the double amount of bandwidth than the flow from node 3. This is caused by the periods where there is only one congestion domain on the ring. In these periods, node 0 will transmit as much as possible, while node 3 must take whatever portion is left. Thus node 3 is suffering from an excessive sending behavior of its upstream neighbor.

This behavior, as can be expected, will remain regardless of whether we use our proposed improvement or not. To achieve RIAS fairness in this kind of scenario, where we have flows crossing two congestion domains, it is not sufficient for the individual nodes to have knowledge about the nearest congestion point only.

After studying the obtained throughput results, we conclude that the performance for our proposed improvement for this type of scenario is marginally better than that of the original fairness algorithm. It is only for the initial period of unfairness shown in Fig. 15 there are some clear differences.

	From 1	**From 2**	**From 14**	**From 15**	**From 16**
Original	18.8%	18.9%	20.9%	20.9%	20.5%
Modified	19.0%	18.9%	20.8%	20.8%	20.4%

Table III: Cumulative bandwidth shares for flows crossing the downstream congestion head.

	From 0	**From 1**	**From 2**	**From 3**
Original	42.9%	18.9%	18.9%	19.4%
Modified	42.6%	19.0%	19.0%	19.5%

Table IV: Cumulative bandwidth shares for flows crossing the upstream congestion head.

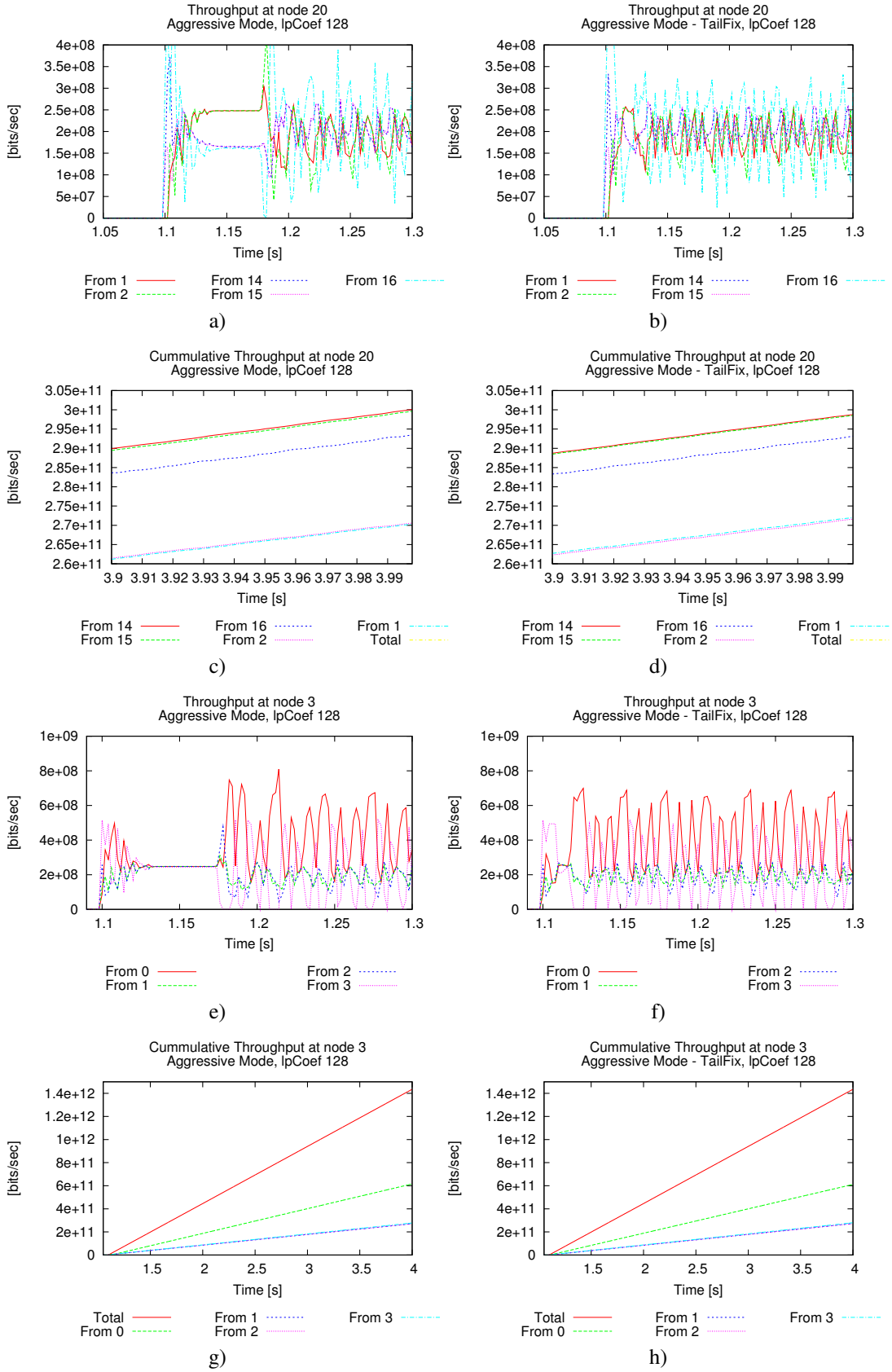


Figure 15: Multiple Congestion Domains with two congestion points. a)-d) shows throughput measurements for the furthest downstream congestion point (measured at the receiver node (node 20)), while e)-h) shows throughput measurements at the furthest upstream (node 3) congestion point. We show performance for the aggressive mode fairness with and without our proposed tail-fix.

VI. Related Work

In an insertion-buffer ring, where the demand for link bandwidth is larger than the available capacity, a fairness algorithm is required to provide fair sharing of bandwidth resources between contending nodes. Many groups have studied the performance and implementation of different algorithms for various insertion-ring architectures [15], [19]–[22]. Several papers have been published studying different RPR performance aspects, both for hardware implementations [15], [23] and simulator models [13], [15], [24]–[26]. Huang et al. presents a thorough analysis of ring access delays for nodes using only one transit queue [24]. Robichaud et al presents ring access delays for class B traffic for both one- and two transit queue designs [25]. Gambiroza et al. focus on the operation of the RPR fairness algorithm and their alternative proposal, DVSR, and their ability, for some given load scenarios to converge to the fair division of rates according to their RIAS fairness reference model [15].

We are not aware of work by others addressing the problem discussed in this paper.

VII. Conclusion

In this paper, we have analyzed a problem, where the the tail of a congestion domain stops the propagation of fair rate estimates (fairness messages) received from a downstream head. The negative side-effect of this is that a node upstream of the tail may send excessive amounts of traffic, thus preventing the convergence of the fairness algorithm. Thus we incur both unfairness as well as non-convergence of the fairness algorithm.

We have proposed a modification, termed the *tail-fix*, that solves these problems by propagating the fair rate estimate beyond the tail of a congestion domain. This modification is easily implemented using one of the currently unused bits in the fairness message as our *passedTail* bit. Our simulations indicate that the modification allows for a substantially lower setting of the *lpCoef* parameter for a given ring size. Further, it gives improved performance as compared to the current RPR standard, by shortening the convergence time for a given ring configuration.

VIII. Further Work

The Resilient Packet Ring is a complex technology, with many operational settings to be configured by the operator. Wrong setting of these parameters can, as illustrated in this paper, lead to unwanted behavior. One area of improvement, that would benefit both the owner and the user of an RPR network, would be the implementation of functionality to ease the task of configuring a Resilient Packet Ring. One opportunity could be the use of protocols that automatically discover the optimal settings for a given network. Specifically, we believe that the *lpCoef* parameter can be more optimally set by dynamically monitoring the conditions in the system.

References

[1] IEEE Computer Society, "IEEE Std 802.17-2004," September 24 2004.

[2] R. Needham and A. Herbert, *The Cambridge Distributed Computing System*. London: Addison-Wesley, 1982.

[3] "IEEE Standard 802.5-1989," IEEE Standard for Token Ring.

[4] F. E. Ross, "An Overview of FDDI: the Fiber Distributed Data Interface," *IEEE J. Select. Areas Commun.*, vol. 7, no. 7, pp. 1043 – 1051, September 1989.

[5] "IEEE Standard 1596-1990," IEEE Standard for Scalable Coherent Interface (SCI).

[6] I. Cidon and Y. Ofek, "MetaRing - A Full Duplex Ring with Fairness and Spatial Reuse," *IEEE Trans. Commun.*, vol. 41, no. 1, pp. 110 – 120, January 1993.

[7] "ISO/IECJTC1SC6 N7873," January 1993, specification of the ATMR Protocol (V.2.0).

[8] H. R. van As, W. W. Lemppenau, P. Zafiropulo, and E. Zurfluh, "CRMA-II: A Gbit/s MAC Protocol for Ring and Bus Networks with Immediate Access Capability," in *Proceedings of the Ninth Annual European Fibre Optic and Local Area Networks Conference (EFOC/LAN'91)*, London, June 1991, pp. 262 – 277.

[9] W. W. Lemppenau, H. R. van As, and H. R. Schindler, "Prototyping a 2.4 Gbit/s CRMA-II dual-ring ATM LAN and MAN," in *Proceedings of the 6th IEEE Workshop on Local and Metropolitan Area Networks*, 1993, pp. 17 – 18.

[10] E. Hafner, Z. Nendal, and M. Tschanz, "A Digital Loop Communication System," *IEEE Trans. Commun.*, vol. 22, no. 6, pp. 877 – 881, June 1974.

[11] C. C. Reames and M. T. Liu, "A Loop Network for Simultaneous Transmission of Variable-length Messages," in *Proceedings of the 2nd Annual Symposium on Computer Architecture*, vol. 3, December 1974.

[12] H. R. van As, "Media Access Techniques: The Evolution Towards Terabit/s LANs and MANs," *Computer Networks and ISDN Systems*, vol. 26, no. 6-8, pp. 603 – 656, March 1994.

[13] F. Davik, M. Yilmaz, S. Gjessing, and N. Uzun, "IEEE 802.17 Resilient Packet Ring Tutorial," *IEEE Commun. Mag.*, vol. 42, no. 3, pp. 112–118, March 2004.

[14] F. Davik, A. Kvalbein, and S. Gjessing, "An Analytical Bound for Convergence of the Resilient Packet Ring Aggressive Mode Fairness Algorithm," in *Proceedings of the 40th annual IEEE International Conference on Communications*, Seoul, Korea, May 16-20 2005, To appear in.

[15] V. Gambiroza, P. Yuan, L. Balzano, Y. Liu, S. Sheafor, and E. Knightly, "Design, analysis, and implementation of DVSR: a fair high-performance protocol for packet rings," *IEEE/ACM Trans. Networking*, vol. 12, no. 1, pp. 85–102, 2004.

[16] C. L. Phillips and R. D. Harbor, *Feedback Control Systems*, 2nd ed. Prentice-Hall, 1991.

[17] H. Tyan, "Design, Realization and Evaluation of a Component-Based Compositional Software Architecture for Network Simulation," Ph.D. dissertation, Ohio State University, 2002.

[18] "OPNET Modeler. http://www.opnet.com." [Online]. Available: http://www.opnet.com/

[19] I. Cidon, L. Georgiadis, R. Guerin, and Y. Shavitt, "Improved fairness algorithms for rings with spatial reuse," *IEEE/ACM Trans. Networking*, vol. 5, no. 2, pp. 190–204, 1997.

[20] I. Kessler and A. Krishna, "On the cost of fairness in ring networks," *IEEE/ACM Trans. Networking*, vol. 1, no. 3, pp. 306–313, 1993.

[21] D. Picker and R. Fellman, "Enhancing SCI's fairness protocol for increased throughput," in *IEEE Int. Conf. On Network Protocols*, October 1993.

[22] J. Schuringa, G. Remsak, and H. R. van As, "Cyclic Queuing Multiple Access (CQMA) for RPR Networks," in *Proceedings of the 7th European Conference on Networks & Optical Communications (NOC2002)*, Darmstadt, Germany, June 2002, pp. 285 – 292.

[23] A. Kirstadter, A. Hof, W. Meyer, and E. Wolf, "Bandwidth-efficient resilience in metro networks - a fast network-processor-based RPR implementation," in *Proceedings of the 2004 Workshop on High Performance Switching and Routing, 2004. HPSR*, 2004, pp. 355 – 359.

[24] C. Huang, H. Peng, F. Yuan, and J. Hawkins, "A steady state bound for resilient packet rings," in *Global Telecommunications Conference, (GLOBECOM '03)*, vol. 7. IEEE, December 2003, pp. 4054–4058.

[25] Y. Robichaud, C. Huang, J. Yang, and H. Peng, "Access delay performance of resilient packet ring under bursty periodic class B traffic load," in *Proceedings of the 2004 IEEE International Conference on Communications*, vol. 2, June 20-24 2004, pp. 1217 – 1221.

[26] H. Kong, N. Ge, F. Ruan, and C. Feng, "Congestion control algorithm for resilient packet ring," *Tsinghua Science and Technology*, vol. 8, no. 2, April 2003.

Paper V

Performance Evaluation and Improvement of Non-Stable Resilient Packet Ring Behavior

Fredrik Davik, Amund Kvalbein and Stein Gjessing

Published: In Proceedings of the 4th International Conference on Networking (ICN'05)

Evaluation: 651 papers were submitted to this conference, of which 238 were accepted. Each paper was reviewed by several members of the Technical Program Committee.

Author Contribution: In this paper, the three authors contributed equally to the general chapters. Kvalbein performed the simulations described in the paper, while Gjessing described the individual simulations scenarios and associated results.

Performance Evaluation and Improvement of Non-Stable Resilient Packet Ring Behavior

Fredrik Davik[1,2,3], Amund Kvalbein[1] and Stein Gjessing[1]

[1] Simula Research Laboratory
[2] University of Oslo
[3] Ericsson Research Norway
{bjornfd,amundk,steing}@simula.no

Abstract. Resilient Packet Ring (RPR) is a new networking standard developed by the IEEE LAN/MAN working group. RPR is an insertion buffer, dual ring technology, utilizing a back pressure based fairness algorithm to distribute bandwidth when congestion occurs. In its attempt to distribute bandwidth fairly, the calculated fair rate in general oscillates and under some conditions the oscillations continue indefinitely even under stable load conditions. In this paper, we evaluate the performance of the RPR ring during oscillations. In particular, we analyze transient behavior and how the oscillations of the fairness algorithm influence the throughput, both on a per node basis and for the total throughput of the ring. For congestion-situations, we conclude that, in most cases, RPR allows for full link-utilization and fair bandwidth distribution of the congested link. A modification to the RPR fairness algorithm has previously been proposed by the authors. We compare the improved fairness algorithm to the original, and find that the modified algorithm, for all evaluated scenarios perform at least as well as the original. In some problem scenarios, we find that the modified algorithm performs significantly better than the original.

Keywords: Resilient Packet Ring, Fairness, Performance evaluation, Simulations, Next generation protocol design and evaluation, Communications modeling, Next Generation Networks Principles, High-speed Networks.

1 Introduction and motivation

Resilient Packet Ring (RPR) is a new networking standard developed by the IEEE 802 LAN/MAN Standards Committee, assigned standard number IEEE 802.17-2004 [1,2]. Although RPR was developed by the LAN/MAN committee, it is designed mainly to be a standard for metropolitan and wide area networks.

RPR is a ring topology network. By the use of two rings (also called ringlets), resilience is ensured; if one link fails, any two nodes connected to the ring, still have a viable communication path between them. When a node wants to send a packet to another node on the ring, it adds (sends) the packet onto one of the

two ringlets. For bandwidth efficiency, the ringlet that gives the shortest path is used by default, but a sender can override this (on per packet basis) if it for some reason has a ringlet preference. When the packet travels on the ring, it *transits* all nodes between the sender and the receiver. When it reaches the destination, the packet is removed (stripped) from the ring. Hence the bandwidth that would otherwise be consumed by the packet on its way back to the sender (as is the case in a Token Ring), can be used by other communications. Such destination stripping of packets leads to what is commonly known as *spatial reuse*.

RPR uses *insertion buffer(s)* for collision avoidance [3,4]. When a packet in transit arrives at a node that is currently adding a packet to the ring, the transiting packet is temporarily stored in an insertion buffer, called a *transit queue* in RPR. In order to get some flexibility in the scheduling of link bandwidth resources between add- and transit traffic, the transit queues may be in the order of hundreds of kilobytes large. In a buffer insertion ring like RPR, a *fairness algorithm* is needed in order to divide the bandwidth fairly between contending nodes, when congestion occurs[4] [5,6]. The RPR fairness algorithm runs in one of two modes, called respectively the conservative and the aggressive mode. The aggressive mode of operation is simpler than the conservative one, and is used by e.g. Cisco Systems. The aggressive mode of the fairness algorithm is used in this paper.

The feedback control system nature of the RPR fairness algorithm makes the amount of add traffic from each sending node oscillate during the transient phase where the feedback control system tries to adjust to a new load [7]. Several papers have reported that in some cases the oscillations decreases and (under a stable traffic pattern) converges to a fair distribution of add rates, while under other conditions the algorithm diverges, and oscillations continues [7,8,9,10].

A stable throughput per sender, decreases the jitter observed by the users of the network, and is hence obviously desirable. In this paper, we analyze and discuss RPR oscillations and, among other things, try to answer a crucial question; do these oscillations degrade the throughput from each node, and also, do they degrade the aggregate throughput performance of the ring?

The main contribution of this paper is the development of understanding of how bandwidth is divided among contending nodes when the system has not reached a stable state, or when a stable state is not possible to reach.

The RPR algorithm is complex, involving a number of different process-models interacting in parallel at both the intra- and inter-node levels as well as variable queueing delays. This makes it hard to analyze an RPR system using analytical methods, although some attempts using simplistic analytical models to study different algorithmic properties for specific load scenarios do exist [7,8]. Hence, in this paper we use simulations to gather knowledge about the behavior we want to study. We will discuss which scenarios and benchmarks we consider important in trying to acquire knowledge that will be valid for a broad range of RPR traffic patterns.

[4] RPR nodes may have different weights, so a fair division might not be an equal one. In this paper, however, we assume all nodes have the same weight.

The rest of this paper is organized as follows. In the next section we give a short introduction to the RPR fairness algorithm, and describe some of its strengths and weaknesses. In section 3, we discuss how to find a good set of scenarios to base our evaluation on. In sections 4 and 5, we present and discuss results from the execution of the different scenarios. In sections 6 and 7, we discuss our proposed modification and assess how it improves the behavior of the fairness algorithm. In section 8, we discuss a notorious worst-case scenario. In the final sections we present related work, conclude and outline further work.

2 The RPR Fairness Algorithm

When several sending nodes try to send over a congested link concurrently, the objective of the RPR fairness algorithm is to divide the available bandwidth fairly between the contending nodes. RPR has three traffic classes, high, medium and low priority. Bandwidth for high and medium traffic is pre-allocated, so the fairness algorithm distributes bandwidth to low priority traffic only. In this paper all data traffic is low priority.

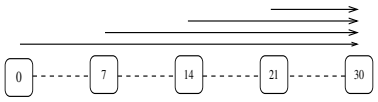


Fig. 1: Nodes 0, 7, 14 and 21 all send traffic to node 30.

The fairness algorithm is a closed-loop control system. The goal of the fairness algorithm is to arrive at the "Ring Ingress Aggregated with Spatial Reuse" (RIAS) fair division of rates over the congested link [8]. The control system encompasses all nodes that send over the same congested link, known in RPR as a *congestion domain*. The node directly upstream of the most congested link is called the *head* of the congestion domain. The node in the congestion domain that is furthest away from the head is called the *tail* of the congestion domain. Later in this paper we are going to use a scenario depicted in figure 1. Here nodes 0, 7, 14 and 21 all send traffic to node 30. When these nodes in total want to send more than the available bandwidth, the most congested link will be the link immediately downstream of node 21 (as well as all other links between node 21 and 30). The congestion domain will consist of all the 22 nodes from node 0 to node 21, i.e., 18 passive and 4 active nodes. Node 0 will be the tail of the domain, node 21 the head. When a node's total transit and add traffic amounts to more than the full bandwidth, the transit queue of the node with a congested out-link will fill up[5]. When the transit queue occupancy is above a threshold called *low*, the node enters a state called *congested* and if it is has not observed that there are downstream nodes that are more congested, it becomes head of a congestion domain. As head, it starts sending

[5] RPR nodes may have one or two transit queues. In the case of a node with two transit queues, the highest priority traffic will use one transit queue, while the two lower priority classes will use the other transit queue. In the RPR model used in this paper there are two transit queues, but since all data traffic will be of the lowest priority, the high priority transit queue will be empty (except for control traffic).

fairness messages, which is the feedback mechanism of the control system. These feedback messages instruct the upstream nodes to restrict their add rate to the head's own current send rate. The head node continues to add traffic until the transit queue occupancy reaches another, higher, threshold called *high*. At this point the head stops its add traffic until the upstream nodes have reduced their send rate so much that the head's transit queue occupancy decreases below the high threshold.

In the aggressive version of the fairness algorithm, the value used by the head as its approximation to a fair rate, is the head's own add rate run through a low-pass filter. When received by the upstream nodes, these nodes restrict their add rate to the received fair rate. The head estimates and advertises new fair rates with short intervals (by default every 100 microseconds).

The time it takes from the head advertises a new fair rate, until the head sees the effect of this action, is the time it takes for a fairness message to reach an upstream node, and then the time it takes for the packets from this node, generated in accordance with the newly received rate message, to reach the head. Hence, in general there is a considerable feedback latency in this system. This latency, combined with the configuration of the algorithm in the head for calculating fair rates, decides the stability of the RPR fairness algorithm.

It has been shown that the fairness algorithm does not always reach a stable state [7]. In general, the calculated fair rate always varies (oscillates) initially in response to transient load conditions. If the (new) traffic load is stable, these oscillations should decay as the fairness algorithm converges to the new fair division of sending rates. For some scenarios however, even under (new) stable load conditions, the fairness algorithm does not converge, and the rate at which each different node is allowed to send, continues to oscillate.

3 Scenario discussion

The goal of this paper is to shed light on the behavior of the RPR ring when the fairness algorithm calculates and advertises oscillating fair rate values. In particular, we want to investigate how throughput performance is affected by transient load- and congestion conditions.

We have written two RPR simulators that run the fairness algorithm according to the final RPR standard text [1]. One simulator is based on OPNET Modeler [11], the other on J-Sim [12]. The results shown in the sequel are results from running the J-Sim simulator (but the OPNET simulator gives the same results).

As described above, congestion occurs when several senders at the same time want to transmit more data than the link capacity (bandwidth) over the same link can sustain (all links in RPR have the same capacity). Some senders may be greedy, i.e. they want to send as much as possible. Other senders send at a limited rate. For very modest senders, i.e. senders sending less than their fair share, RPR should not impose any rate restrictions. For nodes having more to

send than the fair share, RPR should restrict their sending rate to their fair share.

In order to clearly illustrate this behavior of the fairness algorithm, and at the same time not use too much space, we have selected three sets of scenarios. The first uses four greedy senders, the second also uses four senders, but now each sender sends much less than the available link-bandwidth. In the final set there are only two senders, one very modest and one greedy sender. We have run several other scenarios, and to our experience, the main results shown in the sequel are the same as the ones obtained even if there are more senders, or the mix of senders are more complex than what is shown by the presented examples. For all scenarios, the link-length used is 40km (0.2 ms) and the packet size is 500B.

We will use stable traffic load from senders that all start at the same time. We have run other scenarios where all senders do not start at the same time, and, again, we found that this does not contribute more to the understanding. A long term stable traffic load is not very realistic, but we mainly observe the behavior of RPR during the time immediately after traffic startup. Hence we observe RPR during a transient phase. In fact, such transient phases will occur all the time when the traffic load changes in a real system, hence we indeed get realistic information from our scenarios.

One of our main objectives is to see how bandwidth is divided between nodes on a longer time scale compared to a shorter time scale. As described above, the fairness algorithm computes a new fair rate every 100 microsecond, a period that is called an *aging interval*. By experimenting with the sample period, it seems that 20 aging intervals, or 2 ms, i.e. when 2 Mbit has been transmitted over a fully utilized link in a one Gbit/sec system, is a reasonable short term sampling period. Using even shorter sampling periods gives visually the same results.

To see a more coarse grained behavior, the sampling period must be extended. By combining 25 of the short samples (into 50 ms sampling periods) we have found that the (relatively) long term behavior of the fairness algorithm is well illustrated.

Our greedy traffic senders send traffic as fast as they can while our non-greedy senders send packets with fixed intervals. These traffic models are very unlike the self-similar traffic patterns reported in [13,14], believed by many to represent the best models of traffic in real Ethernet- and Internet environments. However, the simple traffic models we have chosen illustrate very clearly the transient behavior of an RPR system trying to adjust to a changing traffic load.

4 All greedy senders – convergence

The scenario we run first is depicted in figure 1. In order to easily understand the results, we have only 4 senders, namely nodes 0, 7, 14 and 21 sending to node 30. (We wanted the distance between the senders to be long. We would have reached the same results with 4 senders with no other nodes in between, but with approximately 7 times as long links.) All senders start transmitting at

full speed at time 0.10 sec. Node 21 becomes the head and node 0 the tail of the congestion domain. Figure 3a shows how the fairness algorithm converges to an approximate fair division of the bandwidth after about another 0.15 sec. From figure 3a we also see that the head (21) and the tail (0) are the two nodes whose add traffic are oscillating the most. We want to investigate how this oscillation influence the throughput of the individual nodes as well as the aggregate system throughput.

We measured the utilization of the link from node 21 to node 22, and found that this congested link is fully utilized all the time. This is good news, because it means that the most scarce resource does not become underutilized under heavy load. In order to see if the bandwidth has been divided equally among the sending nodes before the algorithm converges, we plot the throughput from each node on a more coarse grained scale, using 50 ms sampling periods. The result is depicted in figure 3b. Here we see that initially, the tail node is sending much more than the other nodes, and also that the head is sending more than the nodes inside the congestion domain. It is unfortunate that the tail node is able to send more than the other nodes. A remedy for this has been found by the authors [15]. We return to this problem and its solution in section 6.

The reason the head sends more than the two middle nodes is that whenever the upstream nodes have been slowed down below their real fair rate, the head takes all the rest of the available bandwidth over the congested link. This is also the reason why this link is 100% utilized all of the time. We measure that the transit queue in the head is almost all the time filled to the high threshold, while the transit queues in the other nodes are empty. (Except the transit queue in the active node downstream of the tail, i.e. node 7. Also this behavior will be discussed in section 6.)

Both from figures 3a and 3b, it can be observed that the throughput of the middle nodes are not oscillating much. In fact, from figure 3b, we see that both middle nodes in our scenario, nicely converges towards the fair rate.

5 All greedy senders – no convergence

The RPR fairness algorithm does not converge if the communication latency between the head and the tail is long, and the low-pass filter is not smoothing its own add-rate measurements enough [7]. The degree of smoothing by the low-pass filter is determined by the filter's time-constant. Given a low time-constant, the filter's output-value is affected more (smoothed less) by short transients on the filter-input than is the case of a higher time-constant. The exact same scenario as above is run again, but this time we equip the fairness algorithm with a slightly less smoothing (quicker) low-pass filter. This is done by setting a configurable parameter called *lpCoef* to 128 instead of 256 (allowed values for *lpCoef* are 16, 32, 64, 128, 256 and 512). In figure 3c the outcome of a run is plotted with a 2 ms sampling period, to illustrate how the add rate of all the four senders oscillates. Figure 3d shows the same run with a sampling period of 50 ms. We see also here that node 0 initially gets far more of the bandwidth than its fair

share. After 0.1 sec of packet sending (at time 0.2 sec), node 0 has transmitted 32.9 Mbytes, node 7 has transmitted 20.2 Mbytes, node 14 21.2 Mbytes and node 21 23.8 Mbytes of data. After time 0.2 sec, however, the fairness algorithm manages, on a coarse scale, to divide the capacity of the congested link relatively equally between the nodes. Node 0 (the tail) gets approximately 27.3% of the bandwidth of the congested link, while the others on average gets 24.2%. In the long run, it seems again that it is node 0 (the tail) that gets a little too much, and node 21 (the head) that gets too little.

6 Performance of an improved fairness algorithm

We now return to the problem we identified above, i.e. that the tail node is sending significantly more than the other nodes. Going back to our scenario, and looking at figure 1, node 21 is the head, node 0 is the tail and node 7 is the node in the congestion domain closest to the tail that is active. If node 7 detects that its upstream active neighbor(s) (node 0) currently send less than the fair bandwidth advertised by the head, node 7 informs its upstream neighbor(s) (node 0) that it can transmit without any rate restrictions. The reason for this is that node 7 believes that node 0 is not (currently) contributing to the downstream congestion. Node 7 now assumes the role as the tail of the congestion domain (for a short while), until node 0 has sent so much that node 7 again detects that node 0 indeed contributes to the downstream congestion. During this period the transit queue of node 7 also fills up. Because of low-pass filtering, it takes some time for node 7 to find out that node 0 indeed contributes to the congestion.

In order to avoid this unfair extra sending from node 0, a modification to the RPR fairness algorithm has been proposed by the authors [15]. The proposal includes never sending full rate messages upstream when there is a downstream congestion. Instead the fair rate is propagated (by the tail) further upstream, either all the way around the ring, or until a new congestion domain is encountered. Such fairness messages do no harm, because before they reach the head of a new congestion domain they will mostly pass nodes that send very little (less than the fair rate). The only effect they have is the wanted one; to stop excessive sending from nodes that in reality (but not formally) are part of the congestion domain in which this fairness message originate from the head. Also, the forwarding of the message does not consume any extra resources (bandwidth, per node processing and per node state information) as the messages are sent anyhow, but with a different value in the rate field.

We have looked into the possibility of changing the formal definition of congestion domain tails, by changing the tests and some state variables in the tail. This however seems not to be an easy task, and such changes will also be harder to introduce in a future improved version of RPR.

The described improvement leads to faster convergence of the fairness algorithm and also that the algorithm converges in several scenarios when it used not to converge. Figure 3e shows how the traffic load that lead to persistent os-

cillations in figure 3c, now, with our simple modification, stabilize quite quickly (in approximately 0.1 s). In the present paper we, for the first time, show how our improvement also leads to more fair allocation of bandwidth; From figure 3f it can be observed that the tail (node 0) has no upstream advantage at all and that all nodes, except the head, smoothly converges to the fair rate. For reasons described above, the head still has a small initial advantage, but it is seen that also the head soon converges to the fair rate.

7 Non-greedy senders

In this scenario, we use the same set of active senders as above (figure 1), but now the senders are more modest. When we run with a total load from all four active nodes less than the full bandwidth, RPR behaves very nice, and the fairness algorithm does not even kick in. Then we let each of the active senders transmit at 30% of the full bandwidth, resulting in an aggregate demand of 120% of the full bandwidth. The results are seen in figures 3g and 3h. This time the two plots are not that much different. Figure 3g shows some oscillations, but the most interesting period is clearly visible in both plots, i.e. from time 0.2 to about 0.45 sec. In this period the tail (node 0) is allowed to send much more than its fair share. The reason is the same as explained above in section 6. This time the unwanted behavior is even more noticeable. The reason is that when node 0 this time gets a signal from node 7 that it can send at full speed, it does not, but instead send at 30% of full rate, that is, just above its fair rate. Hence it will take much longer for node 7 to again understand that it has a serious contributor to congestion upstream. After time 0.5 sec, we have reached the steady-state (fair division of rates).

Our modified algorithm alleviates the problem of the tail sending too much. By continuing to send the fair rate upstream, node 0 continues to send at a rate much closer to the real fair rate. The results are shown in figures 3i and 3j. Notice how fast the algorithm now stabilizes, and that it is only node 21 (the head) that has an initial small downstream advantage. The reason for this advantage is, as described above, that the head will utilize any spare bandwidth on the congested link (limited upwards by its demand). Initially, there will be periods with spare bandwidth as the fairness algorithm converges towards the fair rate.

8 Mixed greedy and non-greedy senders

We want to understand how RPR behaves when some senders are greedy and others are not. In order to get an example that illustrates this clearly and simply, we use a notorious worst-case scenario, first presented by Knightly et al. [16].

The scenario contains two senders only; one greedy and one that sends at only 5% of the total bandwidth. The ring used for this experiment is the same as the one used previously in this article, but this time long latencies between nodes do not matter, hence we now let nodes 0 and 1 send to node 2. First we let the 5% sender be the head of the congestion domain (node 1 in our experiment),

and the greedy sender the tail (node 0) . Figure 2 shows the throughput of the two senders.

Initially both nodes send at their assigned rate. Node 0 will send, at full rate, traffic that transits node 1. Because node 1 sends at 5% only, the transit queue in node 1 will then fill up very slowly. When it is filled to the low threshold, a fairness message is sent to node 0. As we now know, the contents of this message is the fair rate, as seen by node 1, and this is the 5% rate, run through a low-pass filter, so it is even less. Consequently, node 0 must reduce its rate to less than 5% of full rate. In fig. 2 this can be seen as the dip at about time 0.11 sec. When node 0 has sent at this low rate for a short time, node 1 is no longer congested, and notifies upstream nodes of this, by advertisement of full-rate messages. Having received full-rate messages, the fairness algorithm gradually allows node 0 to increase its send rate, as seen in fig 2. When the send rate has increased above 95% of full bandwidth usage, node 0 again becomes congested, and once more it will advertise its low-pass filtered add rate, that is below 5% of full bandwidth, and node 0 will again (at about time 0.2 sec) have to decrease its send rate, etc. Because of the large, long-lasting oscillations that can be observed, the total throughput is no longer 100%. In fact, after some time, the average total throughput arrives at approximately 70% of the link-bandwidth, i.e 700 Mbit/sec. Our findings in this scenario confirms what have been reported by others [8,9,10,17].

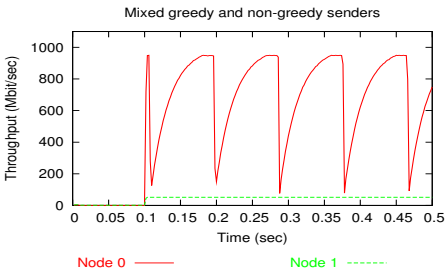


Fig. 2: Modest congestion head node. The figure plots throughput per sender node measured at node 2 using a quick low-pass filter (lpCoef=64) and 2 ms sampling intervals

If the sequence of the greedy and the humble senders is reversed, the greedy head will tell the tail to slow down, but the tail's rate is already below the fair rate calculated by the (greedy) head. The only effect of the fairness algorithm is then that the head has to stop sending in the periods when its transit queue occupancy exceeds the high threshold. The result is a stable division of bandwidth, where the head utilizes 5% of the congested link's bandwidth, while the head utilizes 95% of the congested link's bandwidth. Thus, in total, the congested link is fully utilized all the time. No plot is shown for this scenario.

9 Related Work

In an insertion-buffer ring, where the demand for link bandwidth is larger than the available capacity, a fairness algorithm is required to provide fair sharing of bandwidth resources between contending nodes. Many groups have studied the performance and implementation of different algorithms for various insertion-ring architectures [18,8,19,20,21]. Several papers have been published studying different RPR performance aspects, both for hardware implementations [8,22] and simulator models [2,8,23,24]. Huang et al. presents a thorough analysis of

ring access delays for nodes using only one transit queue [23]. Robichaud et al presents ring access delays for class B traffic for both one- and two transit queue designs [24]. Gambiroza et al. focus on the operation of the RPR fairness algorithm and their alternative proposal, DVSR, and their ability, for some given load scenarios to converge to the fair division of rates according to their RIAS fairness reference model [8].

10 Conclusion

In this paper we have investigated how the bandwidth of a congested RPR system is utilized when the operation of the aggressive mode fairness algorithm results in oscillatory sending behavior. We have executed performance evaluation experiments on our RPR simulation platforms in order to observe RPR behavior during non-stable executions. The set of traffic scenarios have been carefully selected in order to learn as much as possible about general RPR performance. We have also argued that even though we run with stable load, we mainly analyze the initial (transient) phase of the runs. This transient phase resembles the transient phase in a running network where the traffic pattern suddenly changes. Hence, we believe that the knowledge obtained is valid for many real traffic scenarios as well.

We have found that even when the estimated fair rate oscillates, the bandwidth allowances distributed to the contending nodes are relatively equal measured over a longer time scale. In several scenarios we have, however, observed that the most upstream node (the tail) in a congestion domain is initially assigned more bandwidth by the RPR fairness algorithm, than its fair share.

The authors have previously developed a modification to the fairness algorithm. This modification is designed to prevent oscillations caused by the tail getting more bandwidth than the other nodes. In the present paper it is shown, for the first time, that this modification also distributes bandwidth fairly to the tail, both initially, while the modified algorithm stabilizes the fair rate, and in the long run (in case of a stable traffic scenario).

In this paper we have also shown that the RPR fairness algorithm has the nice property that when a link is congested, it is usually utilized fully. The reason for this is that the node just upstream of the most congested link (the head of the congestion domain), may use all free capacity on this link. Hence the most congested link will be fully utilized as long as the head has enough data to send. Sometimes this causes the head to get a little more than its fair share of the bandwidth. In our experiments, the only time the most congested link is not fully utilized is when the head node is sending at a very low rate. In this paper we have also confirmed that such a head node, sending at a very low rate, may cause upstream nodes to send below their fair rate.

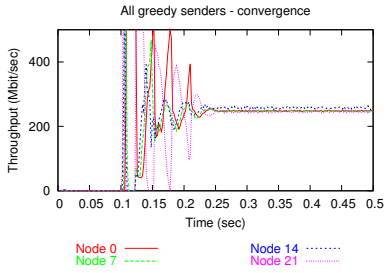
11 Further Work

In future work we want to see if there are other ways to modify the fairness algorithm to give it more wanted properties. Specifically, we want to investigate methods that addresses the problem, discussed in section 8, of throughput loss in congestion domains where the head sends traffic at a rate lower than the fair rate. We would also like to investigate how the oscillations observed in unstable configurations of an RPR ring, running the aggressive mode fairness algorithm, affects packet delays and jitter.

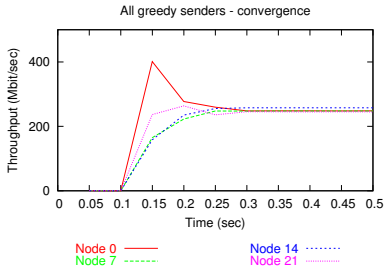
References

1. IEEE Computer Society: IEEE Std 802.17-2004 (2004)
2. Davik, F., Yilmaz, M., Gjessing, S., Uzun, N.: IEEE 802.17 Resilient Packet Ring Tutorial. IEEE Commun. Mag. **42** (2004) 112–118
3. Hafner, E., Nendal, Z., Tschanz, M.: A Digital Loop Communication System. IEEE Trans. Commun. **22** (1974) 877 – 881
4. Reames, C.C., Liu, M.T.: A Loop Network for Simultaneous Transmission of Variable-length Messages. In: Proceedings of the 2nd Annual Symposium on Computer Architecture. Volume 3. (1974)
5. Van-As, H., Lemppenau, W., Schindler, H., Zafiropulo, P.: CRMA-II a MAC protocol for ring-based Gb/s LANs and MANs. Computer-Networks-and-ISDN-Systems **26** (1994) 831–40
6. Cidon, I., Ofek, Y.: MetaRing - A Full Duplex Ring with Fairness and Spatial Reuse. IEEE Trans. Commun. **41** (1993) 110 – 120
7. Davik, F., Gjessing, S.: The Stability of the Resilient Packet Ring Aggressive Fairness Algorithm. In: Proceedings of The 13th IEEE Workshop on Local and Metropolitan Area Networks. (2004) 17–22
8. Gambiroza, V., Yuan, P., Balzano, L., Liu, Y., Sheafor, S., Knightly, E.: Design, analysis, and implementation of DVSR: a fair high-performance protocol for packet rings. IEEE/ACM Trans. Networking **12** (2004) 85–102
9. Alharbi, F., Ansari, N.: Low complexity distributed bandwidth allocation for resilient packet ring networks. In: Proceedings of 2004 Workshop on High Performance Switching and Routing, 2004. HPSR. (2004) 277 – 281
10. Zhou, X., Shi, G., Fang, H., Zeng, L.: Fairness algorithm analysis in resilient packet ring. In: In Proceedings of the 2003 International Conference on Communication Technology (ICCT 2003). Volume 1. (2003) 622 – 624
11. : (OPNET Modeler. http://www.opnet.com)
12. Tyan, H.: Design, Realization and Evaluation of a Component-Based Compositional Software Architecture for Network Simulation. PhD thesis, Ohio State University (2002)
13. Leland, W.E., Taqqu, M.S., Willinger, W., Wilson, D.V.: On the self-similar nature of Ethernet traffic (extended version). IEEE/ACM Trans. Networking **2** (1994) 1–15
14. Paxson, V., Floyd, S.: Wide area traffic: the failure of Poisson modeling. IEEE/ACM Trans. Networking **3** (1995) 226–244
15. Davik, F., Kvalbein, A., Gjessing, S.: Congestion Domain Boundaries in Resilient Packet Rings. Submitted to ICC05 (2004)

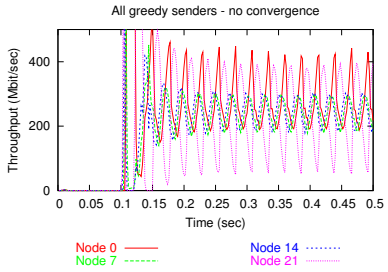
16. Knightly, E., Balzano, L., Gambiroza, V., Liu, Y., Yuan, P., Sheafor, S., Zhang, H.: Achieving High Performance with Darwin's Fairness Algorithm. http://grouper.ieee.org/groups/802/17/documents/presentations/mar2002 (2002) Presentation at IEEE 802.17 Meeting.
17. Yue, P., Liu, Z., Liu, J.: High performance fair bandwidth allocation algorithm for resilient packet ring. In: Proceedings of the 17th International Conference on Advanced Information Networking and Applications. (2003) 415 – 420
18. Cidon, I., Georgiadis, L., Guerin, R., Shavitt, Y.: Improved fairness algorithms for rings with spatial reuse. IEEE/ACM Trans. Networking **5** (1997) 190–204
19. Kessler, I., Krishna, A.: On the cost of fairness in ring networks. IEEE/ACM Trans. Networking **1** (1993) 306–313
20. Picker, D., Fellman, R.: Enhancing SCI's fairness protocol for increased throughput. In: IEEE Int. Conf. On Network Protocols. (1993)
21. Schuringa, J., Remsak, G., van As, H.R.: Cyclic Queuing Multiple Access (CQMA) for RPR Networks. In: Proceedings of the 7th European Conference on Networks & Optical Communications (NOC2002), Darmstadt, Germany (2002) 285 – 292
22. Kirstadter, A., Hof, A., Meyer, W., Wolf, E.: Bandwidth-efficient resilience in metro networks - a fast network-processor-based RPR implementation. In: Proceedings of the 2004 Workshop on High Performance Switching and Routing, 2004. HPSR. (2004) 355 – 359
23. Huang, C., Peng, H., Yuan, F., Hawkins, J.: A steady state bound for resilient packet rings. In: Global Telecommunications Conference, (GLOBECOM '03). Volume 7., IEEE (2003) 4054–4058
24. Robichaud, Y., Huang, C., Yang, J., Peng, H.: Access delay performance of resilient packet ring under bursty periodic class B traffic load. In: Proceedings of the 2004 IEEE International Conference on Communications. Volume 2. (2004) 1217 – 1221



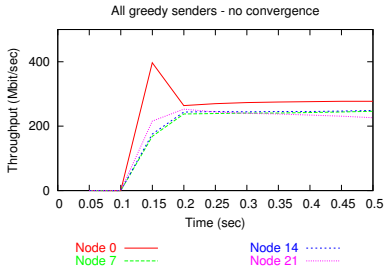
(a) lpCoef=256, 2 ms sampling interval



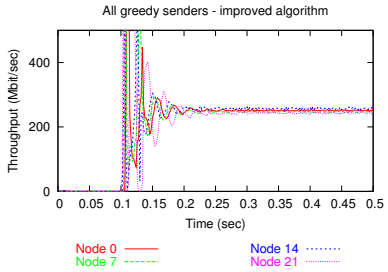
(b) lpCoef=256, 50 ms sampling interval



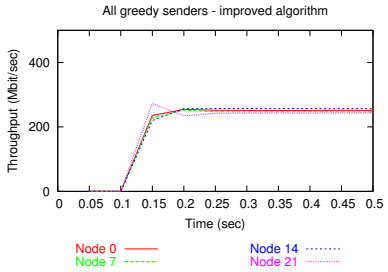
(c) lpCoef=128, 2 ms sampling interval



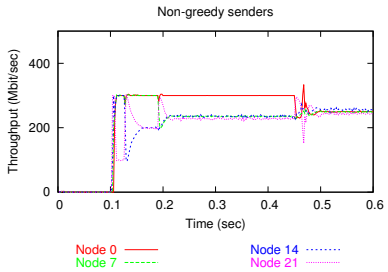
(d) lpCoef=128, 50 ms sampling interval



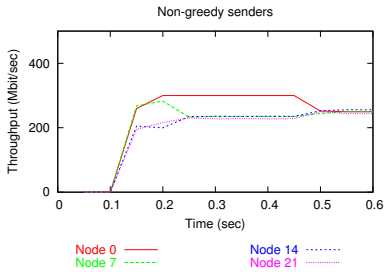
(e) lpCoef=128, 2 ms sampling interval



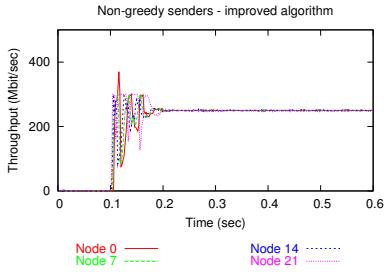
(f) lpCoef=128, 50 ms sampling interval



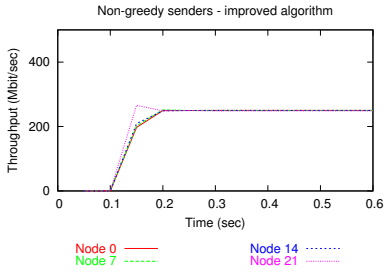
(g) lpCoef=128, 2 ms sampling interval



(h) lpCoef=128, 50 ms sampling interval



(i) lpCoef=128, 2 ms sampling interval



(j) lpCoef=128, 50 ms sampling interval

Fig. 3: Throughput per sender node measured at node 30 using various settings for the lpCoef parameter and two different sampling interval settings.

Paper VI

Resilient Packet Ring Low Priority Traffic Latency

Fredrik Davik, Amund Kvalbein and Stein Gjessing

Published: This paper was published as a Regular Research Paper in the Proceedings of the 2005 International Conference on Communications in Computing (CIC'05)

Evaluation: The CIC'05 proceedings states that the acceptance ratio for the Regular Research Papers is no higher than 50%.

Author Contribution: In this paper, the three authors contributed equally to the general chapters. Kvalbein performed the simulations described in the paper, while Gjessing described the individual simulations scenarios and associated results.

Resilient Packet Ring Low Priority Traffic Latency

Fredrik Davik*†‡, Amund Kvalbein* and Stein Gjessing*
†University of Oslo
‡Ericsson Research Norway
*Simula Research Laboratory
Email: {bjornfd,amundk,steing}@simula.no
Box. 134, 1325 Lysaker, NORWAY
Phone/Fax: +47 67 82 82 00/01

Abstract*— Resilient Packet Ring (RPR - IEEE 802.17) is an insertion buffer, dual ring technology, utilizing a back pressure based fairness algorithm to distribute bandwidth when congestion occurs. The fairness algorithm may oscillate and under some conditions the oscillations continue indefinitely even under stable load conditions. In this paper, we evaluate the latency experienced by packets sent during such oscillations. We analyze transient behavior and how the oscillations of the fairness algorithm influence the jitter caused by unfair access to the ring, as well as jitter caused by the insertion buffers around the ring. We conclude that, in most cases, latency and jitter are within acceptable bounds. A modification to the RPR fairness algorithm has previously been proposed by the authors, but its implications on latency has never before been demonstrated. We compare the improved fairness algorithm to the original, and find that the modified algorithm, for all evaluated scenarios, perform at least as well as the original with respect to latency and jitter. In some problem scenarios, we find that the modified algorithm performs significantly better than the original.

Keywords: Resilient Packet Ring, Fairness, Latency, Jitter, Performance evaluation by Simulations.

1. INTRODUCTION

Resilient Packet Ring (RPR) is a communication standard developed by the IEEE in the 802 LAN/MAN Committee, and is assigned standard number IEEE 802.17-2004 [1], [2]. The ring access method used by RPR was developed in the 1970's, and is called the *insertion buffer* principle [3], [4]. When a frame in transit arrives at a node that is currently adding a packet onto the ring, the transiting packet is placed in an insertion buffer, termed a *transit queue* in RPR. An RPR node may have two transit queues. Then the *Primary Transit Queue (PTQ)* stores high priority frames, while the *Secondary Transit Queue (STQ)* stores medium and low priority frames. The size of the STQ may be in the order of hundreds of kilobytes. In this paper we investigate RPR systems with two transit queues.

When congestion occurs, a buffer insertion ring, like RPR, employs a *fairness algorithm* to divide the bandwidth fairly between the contending nodes [5], [6]. In this paper the so called *aggressive mode* of the fairness algorithm will be used. This mode of operation is simpler than the the other mode, the *conservative mode*, and it is also the one used by e.g. Cisco.

A stable throughput per sender decreases the jitter observed by the users of the network, and is hence obviously desirable. When there is little traffic on the RPR-ring, all packets may be added immediately, and they will not have to wait in the transit queues on their way along the ring to the sender. Hence the latency consist of the propagation delay plus the time it takes for the packets to be forwarded by each node on the ring between the sender and the receiver. During periods when the load is larger than the capacity, the latency may increase for two reasons: i) access delay at the ingress node and ii) increasing node transit delays caused by high *STQ* occupancy.

The main contribution of this paper is the development of understanding of how latency and jitter varies with time and among different nodes when the system has not reached a stable state, or when a stable state is not possible to reach. A modification to the RPR fairness algorithm has previously been proposed by the authors [7], but its implications on latency has never before been demonstrated. In this paper we analyze our modification to the RPR fairness algorithm with respect to latency and jitter. Among other things, we want to investigate how latency is affected by transient load- and congestion conditions. High and medium priority traffic has been shown to have nice latency characteristics [2], [8], [9]. Although low priority traffic (termed class C traffic in RPR) is

said to be "best effort" traffic, its latency characteristics is indeed interesting. For example, a vendor may decide to let all traffic be of equal priority, and since there are certain restrictions with respect to the total amount of high and medium priority traffic, all traffic must then be of the low priority class (class C).

In an RPR system, there are several complex scheduling algorithms, queue dependencies and complex feedback systems that makes it difficult to use analytical models for detailed performance studies. Some attempts using simplistic analytical models, however, have been used to study different algorithmic properties for specific load scenarios [10], [11]. For the purpose of this paper, to obtain sufficient accuracy of the results, performance evaluation by simulations is the only feasible method.

This paper is organized as follows: in section 2, we give a short introduction to RPR, and in particular its associated fairness algorithm. In section 3, we discuss how to find a good set of scenarios to base our evaluation on. In section 4, we present and discuss results from the execution of the different scenarios. In sections 5 and 6, we discuss our proposed modification and assess how it improves the behavior of the original fairness algorithm. In section 7, we discuss a well known worst-case scenario. In the final sections we show some related work, conclude and identify some possibilities for further work.

2. The Resilient Packet Ring Fairness Algorithm

In an RPR network, when a link becomes congested because several sending nodes try to send over it concurrently, the objective of the RPR fairness algorithm is to divide the available bandwidth fairly between the contending nodes [11].

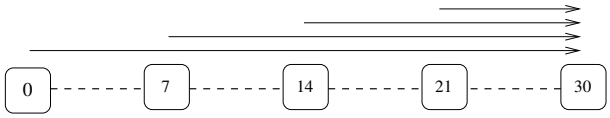


Figure 1: Nodes 0, 7, 14 and 21 all send traffic to node 30

All nodes that send over the same congested link, are members of a *congestion domain*. The node directly upstream of the most congested link is called the *head* of the congestion domain. The node in the congestion domain that is furthest away (upstream) from the congested link is called the *tail* of the congestion domain. Later in this paper we are going to use a scenario depicted in Fig. 1. Here nodes 0, 7, 14 and 21 all send traffic to node 30. The congestion domain will consist of all the 22 nodes from node 0 to node 21, i.e., 18 passive and 4 active nodes. Node 0 will be the tail of the domain, node 21 the head.

When a node becomes head of a congestion domain, it starts sending fairness messages, which is the feedback mechanism of the control system. These feedback messages instruct the upstream nodes to restrict their add rate according to the rate value of the fairness message. The value used by the head as its approximation to the fair rate, is the head's own add rate run through a low-pass filter. The head estimates and advertises new fair rates with short intervals, by default every 100 microseconds.

The fairness algorithm does not always reach a stable state [10]. The calculated fair rate will oscillate when adjusting to changing load conditions. If the (new) traffic load is stable, these oscillations should decay as the fairness algorithm converges to the new fair division of sending rates. For some scenarios however, even under stable load conditions, the fairness algorithm does not converge, and the rate at which each different node is allowed to send, continues to oscillate.

3. Discussing which Scenarios to use

We have developed two RPR simulators that run the fairness algorithm according to the final RPR standard text [1]. One simulator is based on OPNET Modeler [12], the other on J-Sim [13]. The results shown in this paper are results from running the J-Sim simulator (but the OPNET simulator gives the same results).

In a congestion domain, some senders may be greedy, i.e. they want to send as much as possible, while others send at a limited rate. For modest senders, i.e. senders sending less than their fair share, RPR should not impose any rate restrictions. For nodes having more to send than their fair share, RPR restricts their sending rate. As class C traffic is subject to competition for bandwidth resources at all stages from the ingress node and through the network to the egress node, the latency of this traffic class is difficult to estimate, and the RPR standard states that this traffic class has no associated delay guarantees. The purpose of the scenarios analyzed in this paper is to shed more light on the latency characteristics of this traffic class.

In the scenarios used in this paper, all RPR nodes are 40km (0.2 ms) apart, there are a total of 64 nodes on the ring, and the frame size is 500 bytes. By letting all nodes initially be idle, and then let them all start sending at the same time, we demonstrate behavior under transient load conditions.

4. GREEDY SENDERS

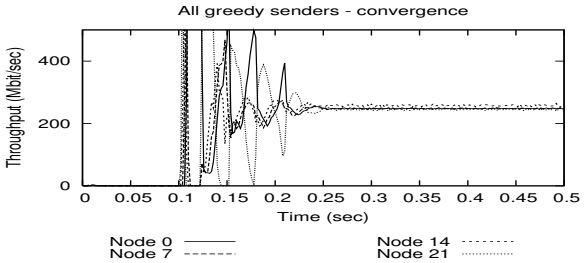
Fig. 1 illustrates our first scenario. For simplicity, there are 4 senders only, namely nodes 0, 7, 14 and 21 sending to node 30. Node 0 becomes the tail, and node 21 the head of the congestion domain.

The first symptom that a congestion situation is imminent, is that the head's *STQ* occupancy starts growing. Once the *STQ* occupancy exceeds a threshold termed *low*, the head is by definition congested. By itself, this does not have a dramatic effect on the delay imposed on traffic passing through the congestion point (both add and transit traffic). However, when the growing *STQ* occupancy of the head reaches another, higher threshold, termed *high*, the RPR scheduler refrains from the sending of local traffic. Thus in this period, until the working of the RPR fairness algorithm makes the transit buffer occupancy fall below the *high* threshold again, the access delay for traffic from the head is greatly increased.

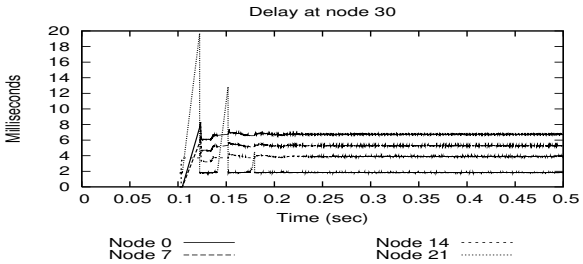
This can be observed in Fig. 2a in terms of reduced throughput for the head in the period [0.11,0.12] and in Fig. 2b by the peak in delay for traffic from node 21, at time 0.125. The effect of the increased transit delay (due to a higher *STQ* occupancy) can be observed for the other traffic flows as well. But as seen, this increase is less significant than the increase in access delay for the head.

During the convergence process towards the fair division of sending rates, the head's *STQ* occupancy may exceed the *high* threshold several times. This is the reason for the additional spikes in delay for traffic from the head at about 0.15 and 0.17 sec.

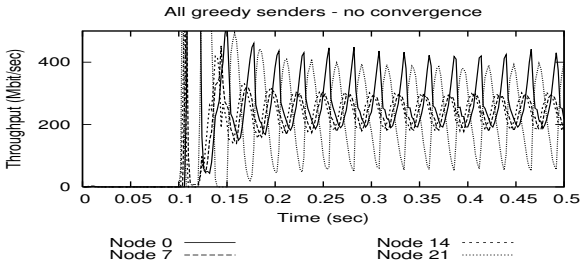
When the distance between the head and the tail is long, and the low-pass filter is not smoothing the add-rate measurements enough, the RPR fairness algorithm does not converge [10]. We run the exact same scenario as above once more, but this time we let the fairness algorithm use a slightly less smoothing (quicker) low-pass filter. The throughput and delay performance is shown in Fig. 2c and Fig. 2d. Initially node 21 takes far more of the bandwidth than its fair share (remember all nodes are greedy), its transit buffers starts filling, and the node becomes congested. We see the same initial spikes in delay for traffic from the head. However, since the fairness algorithm never converges, due to large variations in the fair rate received from the head, we incur large sustained variations in throughput for all nodes. This also incurs variations in the access delay for the nodes upstream of the head. I.e. as the fair rate increases, a packet has, on average, to wait a shorter time



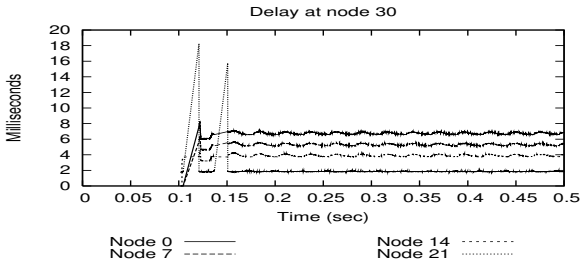
(a) Throughput using a slow low-pass filter (lpCoef=256)



(b) Latency when using a slow low-pass filter (lpCoef=256)



(c) Throughput using a quicker low-pass filter (lpCoef=128)



(d) Latency when using a quicker low-pass filter (lpCoef=128)

Figure 2: Greedy Senders - Aggressive Fairness Algorithm. Throughput and latency per sender node measured at node 30 using 2 ms sampling intervals.

before the scheduler has accumulated enough credits to be added to the ring. And vise versa, as the fair rate decreases, a packet has, on average, to wait a longer time before the scheduler has accumulated enough credits to be added to the ring. The head is to a lesser degree affected by this, as it is not rate controlled directly by the fair rate estimate.

5. PERFORMANCE OF AN IMPROVED FAIRNESS ALGORITHM

The authors have previously proposed an improvement to the RPR fairness algorithm [7], but the latency performance characteristics of an RPR ring where this improvement is implemented has so far been unknown.

The reason that an improvement is needed is that the tail node often is sending significantly more than the other nodes. Looking at Fig. 1, node 21 is the head, node 0 is the tail and node 7 is the tail's closest active downstream neighbor. If node 7 detects that its upstream active neighbor(s) (node 0) currently send less than the fair bandwidth advertised by the head, node 7 informs its upstream neighbor(s) (node 0) that it can transmit without any rate restrictions. When node 0 has a greedy sending behavior, this will result in an average throughput of node 0, in excess of its fair share.

By modifying the RPR fairness algorithm it is possible to avoid this unfair extra sending from node 0. The modification includes never sending full rate messages upstream when there is a downstream congestion. Instead the fair rate is propagated (by the tail) further upstream, either all the way around the ring, or until a new congestion domain is encountered. Such fairness messages do no harm, because before they reach the head of a new congestion domain they will mostly pass nodes that send very little (less than their fair share). The only effect they have is the wanted one; to stop excessive sending from nodes that in reality (but not formally) are part of the congestion domain in which this fairness message originate from the head. Also, the forwarding of the messages does not consume any extra resources as the messages are sent anyhow, but with a different value in the rate field.

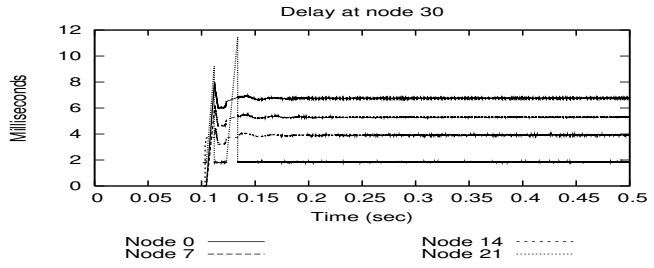


Figure 3: Latency per sender node measured at node 30 using the modified fairness algorithm

Our modification not only leads to improved throughput characteristics as reported in [7], in the present paper it is also shown that the latency performance is improved. The performance of our modified algorithm is depicted in Fig. 3. We see that, when not taking into consideration latency caused by the propagation delays (9*0.2=1.8ms for traffic from the head), the maximum latency is now down from about 18 ms. to about 10 ms.; that is about a 50 % improvement. Notice that the maximum on the y axis now is 12 ms., as opposed to 20 ms. in the previous plots. Also notice that now all nodes have about the same latency increase during the first oscillations (at time 0.11 sec.). At approximately time 0.12 sec. the transit queue of node 21 completely empties, and all upstream nodes (0, 7 and 14) experience no added latency. Also node 21 believes it is not congested any more, and tell the upstream nodes to send at full speed. But this greedy sending by all nodes will very soon result in the transit queue of node 21 to fill up to the *high* threshold, thus temporarily disallowing the adding of additional traffic by node 21. In Fig. 3 we see that at time 0.13 sec. this causes the (access) delay of node 21 to increase as well as introducing a moderate increase in the (transit) delay for traffic from the upstream neighbors. But soon after, the fairness algorithm converges, and all latencies become stable.

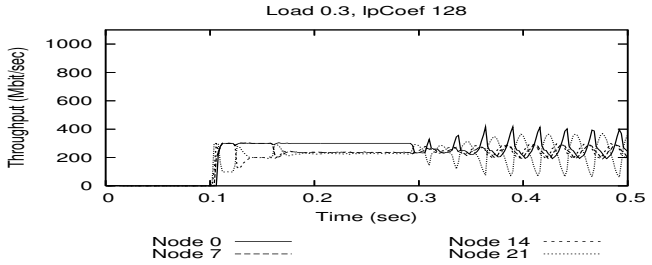
6. NON-GREEDY SENDERS

In this scenario, we use the same set of active senders as above (Fig. 1), but now the senders are more modest. When the total load from all four active nodes is less than the full bandwidth, RPR behaves nicely; the fairness algorithm does not even kick in and latency and jitter behaves very controlled (not shown).

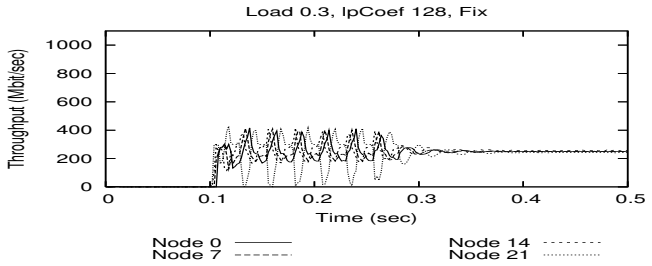
If we let each of the four active senders transmit at 30% of the full bandwidth, resulting in an aggregate demand of 120% of the full bandwidth, the system becomes congested. Some results are shown in Figs. 4a (throughput) and 4c (latency).

As seen in Fig. 4a, the aggressive fairness algorithm has an initial, semi-stable period, where the operation of the fairness algorithm maintains an excessive sending behavior by the tail. As the simulated time progresses, at time 0.3, the operation of the aggressive fairness algorithm makes a transition to the unstable state and remains there. Thus the algorithm never reaches a steady-state and the oscillatory behavior of the throughput and latency performance (Fig. 4c) is sustained.

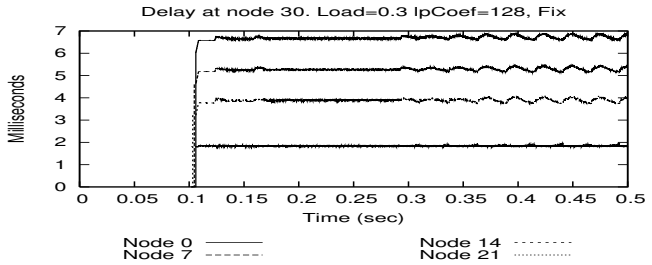
The use of our modified algorithm alleviates the problem of the tail sending too much. During the initial (transient) phase the throughput (Fig. 4b) and latency (Fig. 4d) of the individual flows varies. However, at time 0.3, when the fairness algorithm has converged,



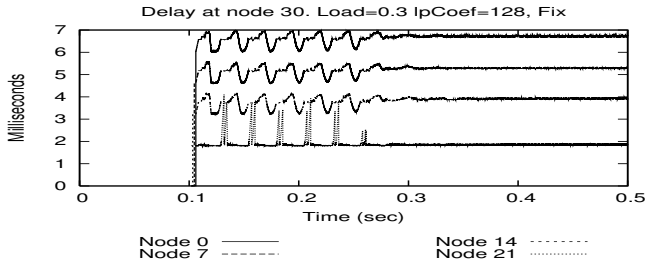
(a) Throughput Aggressive Fairness



(b) Throughput Modified Fairness



(c) Latency Aggressive Fairness



(d) Latency Modified Fairness

Figure 4: 30% Senders - Aggressive and Modified Fairness - Throughput (a and b) and latency (c and d) per sender node measured at node 30 when each node contributes with 30% of link capacity, using a medium low-pass filter (lpCoef=128) and 2 ms sampling intervals.

the throughput and latency performance for all flows stabilizes at a constant level.

7. GREEDY AND NON-GREEDY SENDERS

It has been reported, first by Knightly et al., that RPR throughput oscillates considerably when the head of the congestion domain transmits at a modest rate [14]. Below we analyze how latency and jitter are influenced by these oscillations. In order to analyze the situation, we use the scenario depicted in Fig. 5, containing two senders; one greedy sender and one that sends very little; i.e. at only 5% of the total link capacity. Fig. 6 shows the latencies of the traffic sent from the two nodes as it is received by node 2.

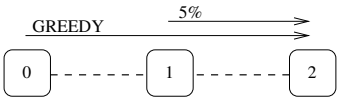


Figure 5: Unbalanced Scenario. The upstream node has an infinite demand (greedy sending behavior), while the downstream node has a modest sending behavior (sends at 5% of the line rate).

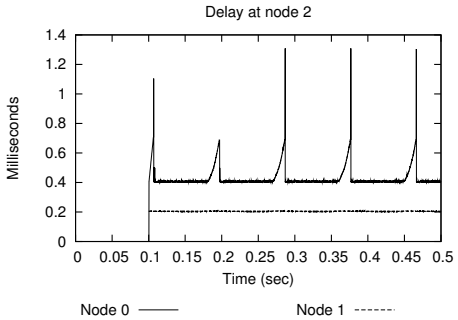


Figure 6: Modest congestion head node. The figure plots latency per sender node measured at node 2 using a quick low-pass filter (lpCoef=64)

The most noticeable characteristics of Fig. 6 is the latency peaks at times 0.1 sec., 0.2 sec, 0.29 sec. etc. The reason for this is that when the tail (node 0) sends at full speed, the transit queue in the head (node 1) is filling up. Then the latency of the packets transiting node 1 starts to increase. This can bee seen in Fig. 6 as the latency gradually increases before the peaks.

Node 1 has been sending at a very low rate, and since the head's approximation to the fair rate is its own current send rate, the fair rate becomes a very low rate. The fairness message is sent to node 0, that has to adjust its send rate to this very low fair rate. Some packets will then have to wait a long time at the head of the add queue before the scheduling rules allow transmission, and this results in the peaks in Fig. 6.

8. Related Work

The performance and implementation of different algorithms for various insertion-ring architectures have been studied extensively [11], [15]–[17]. Papers have been published studying different RPR performance aspects, both for hardware implementations [11], [18] and simulator models [2], [8], [11], [19]. Huang et al. presents a thorough analysis of ring access delays for nodes using only one transit queue [19]. Robichaud et al presents ring access delays for class B traffic for both one- and two transit queue designs [8]. Gambiroza et al. focus on the operation of the RPR fairness algorithm and their alternative proposal, DVSR, and their ability, for some given load scenarios to converge to the fair division of rates according to their RIAS fairness reference model [11].

9. Conclusion and future work

In this paper we have focused on the performance of the RPR fairness algorithm with respect to jitter and latency for low priority traffic. Our simulator has been used in evaluation experiments to observe RPR performance during stable and non-stable executions.

The traffic scenarios used have been carefully selected in order to learn as much as possible about the general RPR performance. Due to a larger diversity in real traffic, exactly the same performance as shown in this article will not be seen in real traffic, but the underlying characteristics will be the ones shown.

A previously published modification to the fairness algorithm was designed by the authors to prevent oscillations caused by the tail getting more bandwidth than the other nodes. In the present paper it is shown, for the first time, that this modification also leads to better latency characteristics for the involved nodes.

In future work, it would be interesting to look further into the performance of the RPR conservative fairness algorithm. In particular, it would be interesting to investigate whether it is possible to improve its steady-state throughput performance.

References

[1] IEEE Computer Society, "IEEE Std 802.17-2004," September 24 2004.

[2] F. Davik, M. Yilmaz, S. Gjessing, and N. Uzun, "IEEE 802.17 Resilient Packet Ring Tutorial," *IEEE Commun. Mag.*, vol. 42, no. 3, pp. 112–118, March 2004.

[3] E. Hafner, Z. Nendal, and M. Tschanz, "A Digital Loop Communication System," *IEEE Trans. Commun.*, vol. 22, no. 6, pp. 877 – 881, June 1974.

[4] C. C. Reames and M. T. Liu, "A Loop Network for Simultaneous Transmission of Variable-length Messages," in *Proceedings of the 2nd Annual Symposium on Computer Architecture*, vol. 3, December 1974.

[5] H. Van-As, W. Lemppenau, H. Schindler, and P. Zafiropulo, "CRMA-II a MAC protocol for ring-based Gb/s LANs and MANs," *Computer-Networks-and-ISDN-Systems*, vol. 26, no. 6-8, pp. 831–40, March 1994.

[6] I. Cidon and Y. Ofek, "MetaRing - A Full Duplex Ring with Fairness and Spatial Reuse," *IEEE Trans. Commun.*, vol. 41, no. 1, pp. 110 – 120, January 1993.

[7] F. Davik, A. Kvalbein, and S. Gjessing, "Congestion Domain Boundaries in Resilient Packet Rings," Simula Research Laboratory, Tech. Rep. 2005-03, February 2005. [Online]. Available: http://www.simula.no

[8] Y. Robichaud, C. Huang, J. Yang, and H. Peng, "Access delay performance of resilient packet ring under bursty periodic class B traffic load," in *Proceedings of the 2004 IEEE International Conference on Communications*, vol. 2, June 20-24 2004, pp. 1217 – 1221.

[9] F. Davik and S. Gjessing, "Applying the DiffServ Model to a Resilient Packet Ring Network," Simula Research Laboratory, Technical Report 2005-1, February 2005. [Online]. Available: http://www.simula.no

[10] F. Davik, A. Kvalbein, and S. Gjessing, "An Analytical Bound for Convergence of the Resilient Packet Ring Aggressive Mode Fairness Algorithm," in *Proceedings of the 40th annual IEEE International Conference on Communications*, Seoul, Korea, May 16-20 2005, To appear in.

[11] V. Gambiroza, P. Yuan, L. Balzano, Y. Liu, S. Sheafor, and E. Knightly, "Design, analysis, and implementation of DVSR: a fair high-performance protocol for packet rings," *IEEE/ACM Trans. Networking*, vol. 12, no. 1, pp. 85–102, 2004.

[12] "OPNET Modeler." [Online]. Available: http://www.opnet.com/

[13] H. Tyan, "Design, Realization and Evaluation of a Component-Based Compositional Software Architecture for Network Simulation," Ph.D. dissertation, Ohio State University, 2002.

[14] E. Knightly, L. Balzano, V. Gambiroza, Y. Liu, P. Yuan, S. Sheafor, and H. Zhang, "Achieving High Performance with Darwin's Fairness Algorithm," July 2002, presentation at IEEE 802.17 Meeting. [Online]. Available: http://grouper.ieee.org/groups/802/17/documents/presentations/mar2002

[15] I. Cidon, L. Georgiadis, R. Guerin, and Y. Shavitt, "Improved fairness algorithms for rings with spatial reuse," *IEEE/ACM Trans. Networking*, vol. 5, no. 2, pp. 190–204, 1997.

[16] I. Kessler and A. Krishna, "On the cost of fairness in ring networks," *IEEE/ACM Trans. Networking*, vol. 1, no. 3, pp. 306–313, 1993.

[17] J. Schuringa, G. Remsak, and H. R. van As, "Cyclic Queuing Multiple Access (CQMA) for RPR Networks," in *Proceedings of the 7th European Conference on Networks & Optical Communications (NOC2002)*, Darmstadt, Germany, June 2002, pp. 285 – 292.

[18] A. Kirstadter, A. Hof, W. Meyer, and E. Wolf, "Bandwidth-efficient resilience in metro networks - a fast network-processor-based RPR implementation," in *Proceedings of the 2004 Workshop on High Performance Switching and Routing, 2004. HPSR*, 2004, pp. 355 – 359.

[19] C. Huang, H. Peng, F. Yuan, and J. Hawkins, "A steady state bound for resilient packet rings," in *Global Telecommunications Conference, (GLOBECOM '03)*, vol. 7. IEEE, December 2003, pp. 4054–4058.

Paper VII

Applying the DiffServ Model to a Resilient Packet Ring Network

Fredrik Davik and Stein Gjessing

Published: A previous short version of this paper (technical report) is published in Proceedings of Networking 2005 were it was accepted as a poster. This technical report is a significantly revised and expanded version of that presented in the proceedings of Networking 2005.

Evaluation: 430 papers were submitted to Networking 2005, of which 105 were accepted as full papers and 36 were accepted as posters. The paper was reviewed by three reviewers.

Author Contribution: In this paper, the derivation of the analytical expressions, the simulations and the associated explanations are all done by Davik.

The contribution from Gjessing is mainly related to the assurance of a clear and precise presentation of the main ideas and results obtained.

Applying the DiffServ Model to a Resilient Packet Ring Network

February 2005
Revised August 2005

Fredrik Davik[1,2,3] and Stein Gjessing[1]

[1] Simula Research Laboratory
[2] University of Oslo
[3] Ericsson Research Norway
{bjornfd,steing}@simula.no

Abstract. In June 2004, the IEEE approved a new standard called Resilient Packet Ring (RPR), that is maintained in the 802 LAN/MAN Committee and designated standard number IEEE 802.17-2004. Among the features provided by the RPR technology are built-in QoS capabilities for traffic class differentiation, bidirectional transfer of data with destination stripping and spatial reuse and fast protection against node and link failure(s). In this paper, we introduce a framework used to specify the throughput of RPR, and propose a simple mapping between RPR's service classes and DiffServ Per Hop Behavior groups. We evaluate this mapping analytically, using a simple generic example, and by simulation using some selected scenarios. All our findings support that our proposed mapping between RPR's traffic classes and the PHB groups is indeed a viable one.

Keywords: Resilient Packet Ring (RPR, IEEE 802.17), Differentiated Services (DiffServ), Per Hop Behavior.

1 Introduction

Transfer of data using the Internet is commonly considered as being a best-effort service: there are no guarantees associated to the transfer of data along any of the traditional Quality of Service (QoS) dimensions: throughput, delay, jitter, data corruption or data loss. For the past two decades, a multitude of mechanisms have been proposed in order to enable Internet service providers to offer IP-based data transfer with QoS guarantees to their customers. These range from simple queueing and scheduling mechanisms, on a per router basis, to more advanced queueing and scheduling mechanisms in combination with resource reservation, packet classification, admission control, policing, shaping and different types of back-pressure. Despite the existence of technical frameworks and actual implementations that can provide IP QoS in some form, the availability of differentiated IP services is not commonplace [1]. In addition to the technical challenges related to the provisioning of IP QoS, there are also a number of non-technical issues that must be handled such as accounting, charging and billing [2].

Today, the DiffServ [3] framework, proposed by the Internet Engineering Task Force (IETF), appears to be the most promising and accepted technical solution for the provisioning of IP-based QoS [4].

A recent addition to the IEEE family of standards for LAN/MAN networks is the IEEE 802.17 Resilient Packet Ring (*RPR*) [5]. In this paper, we introduce a formal specification of parts of the service differentiation mechanisms of the RPR standard, and assess RPR's suitability for use in a DiffServ environment. We propose a simple mapping between RPR's service classes and three standardized DiffServ Per Hop Behavior groups. When using this mapping, conformance to the DiffServ Per Hop Behavior groups is discussed and evaluated based on analytical as well as simulation results. For the simulation, we have implemented the RPR standard in the OPNET Modeler discrete event simulator [6].

The rest of this paper is organized as follows: To provide the reader with sufficient background to understand our contribution, in the next sections we provide a short introduction to the RPR technology and the DiffServ Per Hop Behavior groups. Next, in section 4, we present a formal framework for the delay and per service class traffic differentiation mechanisms of RPR. Then, in section 5, our mapping between RPR and DiffServ is discussed and specified. By use of a simple generic example, the performance of this mapping is demonstrated analytically in section 6. In section 7, we proceed to the evaluation of our proposed mapping by use of three simulation scenarios. Then, in section 8, we asses the mapping based on both the analytic and simulation results. Finally, in sections 9, 10 and 11, we present related work, conclude and point out directions for future work.

2 The Resilient Packet Ring Architecture

RPR is a dual ring technology, i.e. data and control messages can be sent from a source node to a destination node on the ring in either of the two directions

(clockwise or counterclockwise). The buffer insertion principle [7, 8] allows a node on the ring to transmit variable length locally-generated data packets on an output link as long as there are no data packets in transit. This simple insertion buffer principle, however, must be extended considerably to constitute a full fledged Medium Access Control protocol providing fairness, service class differentiation and protection [9].

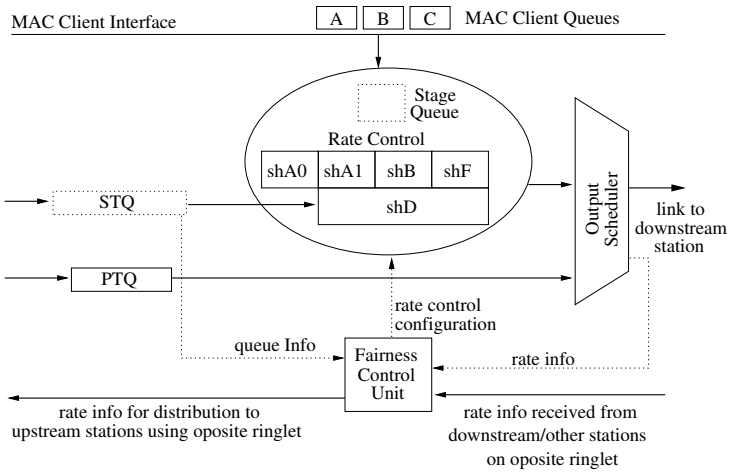


Fig. 1: RPR node design, showing a node's attachment to the ring for transfer of data in the East direction and control information in the West direction. The solid lines indicates the flow of data through the node. The dotted lines indicates the exchange of control/configuration information between node internal function blocks.

Fig. 1 shows the design of an RPR node. The standard allows for use of one or two transit queues in the transit path, denoted as respectively a *1TB*- or *2TB*[4]-design. In the case of a *1TB* design, the transit queue is referred to as the primary transit queue (PTQ), in the *2TB* design, the additional transit queue is referred to as the secondary transit queue (STQ).

The *Rate Control Block* consist of a set of shapers, denoted *shA0*, *shA1*, *shB*, *shF* and *shD*, which enforces rate control for the different service classes. Additionally, the *Stage Queue* can be considered a logical[5] queueing element, providing temporary storage for a packet received from the MAC client immediately before its transmission onto the ring.

In the *1TB* design, all transit traffic, awaiting scheduling onto the output link, is stored in the PTQ. In the *2TB* design, the PTQ stores high priority transit

[4] The purpose of the *2TB* design is to maximize bandwidth reuse by reducing the bandwidth allocation requirements for provisioning of high priority traffic.

[5] The RPR standard does not mandate the use of a stage queue, neither does it specify any requirements regarding its implementation.

traffic, while the *STQ* stores the remaining transit traffic. For the reminder of this article, the authors focus on the use of a *2TB* transit path design.

The *Output Scheduler* is used for selecting between contending packet sources when the output link is ready to accept a new packet.

Finally, the *Fairness Control Unit*, maintains the state for the distributed fairness algorithm, ensuring that for congested links, the contending nodes gets their fair share of the available link bandwidth. The operation of the RPR fairness algorithm is discussed in [9, 10, 11, 12].

It is the responsibility of the MAC client (denoted *client* onwards) to classify node ingress traffic in the three available service classes A, B and C. Service class A is the highest priority service class and is implemented to provide guaranteed throughput and low jitter values independent on the ring circumference. Service class B, which is the 2nd priority service class, is intended to provide guaranteed throughput and a bounded jitter range which is dependent on the circumference of the ring. Class B traffic transmitted in excess of the allocated portion (the *B-CIR* (Committed Information Rate) portion), is sent on the expense of class C traffic. The excess portion is denoted *B-EIR* (Excess Information Rate). Finally, service class C, which is the lowest priority service class, is a best-effort/scavenger service class and has no associated guarantees. We will discuss the properties of the RPR service classes more in section 4.

3 The DiffServ Architecture

In this chapter, we give a brief introduction to DiffServ and some of its most commonly used building blocks (PHBs). We also discuss the requirements that must be met by a DiffServ implementation, in order to claim conformance to these PHBs. In section 5, we will discuss these building blocks used in the context of an RPR network.

Differentiated Services (*DiffServ*) [3] was introduced as a "simple" and scalable framework for providing IP-based service class differentiation in an IP-network. In a DiffServ enabled network, IP packets are classified and marked at the network's ingress node(s). Based on some classification rule at the ingress node, the packet is assigned a DiffServ Code Point value (DSCP), carried within the packet's DS field [13]. This value maps the packet onto the network's available Per Hop Behaviors (PHB[6]), resulting in a specific packet forwarding at each DiffServ compliant node traversed by the packet.

DiffServ has a PHB named Expedited Forwarding (EF) [14]. The Expedited Forwarding PHB is specified with the intent of providing a DiffServ building block that can be used for the provisioning of services that provides low loss, low delay and low jitter. To be EF compliant, a DiffServ node has to comply to quantified delay and jitter values that are function of the rate R provided by the PHB. The EF basic conformance requirements for a DiffServ EF PHB implementation is specified in [14]. The most important conformance requirements

[6] A single PHB is a special case of a PHB group.

are shown in (1) and (2) below.

$$\forall j > 0 : D_j <= F_j + E_a, \text{ where } F_j \text{ is defined iteratively by} \quad (1)$$

$$F_0 = 0, D_0 = 0, \forall j > 0 : F_j = max(A_j, min(D_j - 1, F_j - 1)) + \frac{L_j}{R} \quad (2)$$

In short, (1) (2), introduces a bound on the actual time (D_j), when a packet, j, should leave the node. In addition to the actual arrival time (A_j) of packet j, the ideal departure time (F_j) takes into account the actual- and ideal departure times of the previous packet, $j-1$, that was sent from the same node. The expression also includes an error term, E_a, that represents the worst case deviation between the actual and ideal departure time of any EF packet to traverse this node. Finally the fraction $\frac{L_j}{R}$ accounts for the ideal per packet transmission delay for an EF PHB with a committed service rate of R transmitting a packet j of length L_j.

Another DiffServ PHB, namely the Assured Forwarding (AF) PHB is a specification of PHB group, that may contain up to four independent AF classes [15]. Each class must be allocated a separate amount of forwarding resources. Within each AF class, an implementation must provide a minimum of two different drop probabilities and a maximum of four different drop probabilities. Conformance to the AF PHB by a DiffServ node is described in terms of the throughput obtained, relative service for the different drop probabilities within an AF class and no reordering of packets within an AF micro-flow.

A third PHB, is the so called Class-Selector PHB specified in [13]. This PHB can be implemented in a DiffServ node to provide a network that is compatible with the historical IP Precedence use.

The last PHB we want to introduce to the reader, is DiffServ's default PHB, which is specified in [13] as a best-effort (BE) forwarding behavior. For the reminder of the paper, we refer to traffic belonging to the default PHB, for BE traffic.

In this paper we limit the discussion the EF, AF and the default PHBs.

4 Delay Guarantees and Rate Control Functionality in RPR

In this chapter we start in section 4.1 by providing a detailed description of the workings of the RPR *Rate Control Block* and its associated interface towards the *client*. This is necessary in order to provide to the reader sufficient understanding of how the service differentiation between the three service classes A, B and C is performed in an RPR node. At the end of that section, we present our set of invariants that constitutes a specification of the differentiation between the RPR service classes.

Then, in section 4.2, we provide a brief introduction to RPR delay guarantees for high-priority traffic.

4.1 Rate Control

The MAC layer provides, via the rate control block (see Fig. 1), the client with per service class information on for which ringlet traffic can currently be accepted. In the case of class C traffic, the MAC layer also specifies an additional per ringlet constraint, namely the maximum distance (hop count) a packet is allowed to travel on the associated ringlet. If this value is less than 255 (the maximum number of nodes on an RPR ring), this indicates two things: i) there is a congestion point (a congested link) on the associated ringlet and ii) transmission of class C traffic beyond the congestion point is currently not allowed. This information can be utilized by *client*s implementing some form of Virtual Destination Queueing [16] to avoid Head-Of-Line (*HOL*) blocking [17]. In this context, *HOL* blocking would be that packets at the head of a client queue, if transmitted onto their associated ringlet, would traverse the congested link on their way to their destination and thus have to remain in the queue, while other packets in the same queue destined for nodes before the congested link are blocked.

When the MAC layer, via the rate control block, indicates that it can accept class C traffic, the *client* can make a local decision on whether it wants to transfer class B traffic instead. This can be done, as long as the distance the class B packet will travel on the ring, is within the current maximum (see discussion above). The effect of this is that the MAC client may transmit class B traffic in excess of the (configured) *shB* shaper setting. If the *client* chooses to do so, when the demand is greater than the (allowed portion) of the link capacity, this is done on the expense of class C traffic.

Once a packet has been transferred from the *client* to the MAC layer, the rate control block assigns it into one of the following subclasses $A0$, $A1$ (*2TB* implementations only), *B-CIR*, *B-EIR* and C.

Subclass $A0$ traffic is rate controlled via the *shA0* shaper and is together with subclass $A1$ traffic the highest priority service class. Subclass $A1$ traffic is rate controlled via the the union of shapers *shA1* and *shD*.

The purpose of the $A1$ traffic class for *2TB* implementations is for a given guaranteed amount of A traffic, to allow for lower priority traffic classes to reuse some of this capacity, the $A1$ capacity, when not in use. By use of the *STQ*, this is done without compromising the associated throughput and delay/jitter guarantees.

When a packet marked as class A is received by the *Rate Control Block* from the *client*, it is marked as belonging to subclass $A0$ or $A1$, depending on the status of shaper *shA0* and the union of shapers *shA1* and *shD*. Packets of subclasses $A0$ and $A1$ experience the same delays both at the ingress node and in the transit path (both subclasses goes through the *PTQ* in transit nodes).

It it the user's (network operator) responsibility to configure the nodes on the ring. To facilitate (spatial) reuse of the available bandwidth resources, the amount of traffic of subclass $A0$ should be kept as low as possible.

Class B traffic is rate controlled by the union of shapers *shB* and *shD* and the union of shapers *shF* and *shD*. The class B traffic sent within the rate bounds

of shaper *shB* is denoted *B-CIR* (Committed Information Rate). The class B traffic sent in excess of the rate bounds of shaper *shB*, maybe on the expense local class C traffic, is denoted *B-EIR* (Excess Information Rate).

Class C traffic is rate controlled by the union of shapers *shF* (shaper for fairness eligible traffic) and *shD*. The RPR standard defines different methods for implementing the *shF* shaper. The configuration of shaper *shF* is performed dynamically, based on rate measurements performed at the output of the *Output Scheduler* (see Fig. 1) and calculations performed by the *Fairness Control Unit*.

The *Rate Control Block* imposes several per ringlet and per node rate constraints on local ingress (add)- and transit traffic.

Below to simplify the presentation form, we introduce some notation to be used for the reminder of this paper.

We use the notation R_X to specify the rate constraint in effect for a particular type of traffic. In the cases where $X \in \{A, B, C\}$, X specifies a rate constraint for a particular RPR service class. *offered(X)* represents the amount of traffic of a particular traffic class that a node or a set of nodes **want(s)** to transmit. *accepted(X)* refers to the corresponding amount of traffic that **can** be transmitted.

In (3) and (4) below, we define reserved (preallocated) and unreserved (reclaimable) bandwidth on an RPR ring, denoted respectively R_R and R_U. The bandwidth of the link is denoted R_L.

$$R_R \triangleq \sum_{j \in \{nodes\}} R_{A0_j} \tag{3}$$

$$R_U \triangleq R_L - R_R \tag{4}$$

The invariant below expresses that the sum of preallocated (subclass $A0$) bandwidth, cannot exceed the actual link capacity (R_L).

invariant 1 $R_R \leq R_L$

Invariant 1 is enforced only during ring initialization (start-up and any topology change (node addition or removal)) and triggers a node alarm if violated. In a running system, it is not possible to transmit preallocated traffic at a rate greater than the link-rate. However, if the configuration is done so that invariant 1 is violated, this effectively prohibits the sending of traffic of all other traffic classes. This follows from the invariants specified below.

Invariant 2, restricts the sum of $A1$ and *B-CIR* traffic upwards to the rest of the available bandwidth. Currently, there is no functionality in the RPR standard that ensures that this invariant is met. If the configuration of corresponding shapers, *shA1* and *shB*, violates the invariant, this may break class A and B service guarantees. Thus, it is left to the operator of an RPR network to ensure that the configuration of *shA1* and *shB* shapers does not violate this invariant.

invariant 2 $\sum_{j \in \{nodes\}} \left(R_{A1_j} + R_{B_j} \right) \leq R_U$

The restrictions shown in invariants 3-8 effectively throttles the amount of $A0$ and $A1$ add traffic a node can add to the ring (to a configured value). The parts of the invariants relating to $A0$ traffic is enforced solely by the *shA0* shaper shown in Fig. 1. For the parts of the invariants related to $A1$ traffic, this is enforced by the union of the *shA1* and *shD* settings.

One interesting observation is that when both A0 and A1 traffic is supported (i.e. $R_{A0} > 0 \wedge R_{A1} > 0$), it is impossible to predict the portion of offered traffic that is accepted as respectively A0 or A1 traffic. This also applies on a per packet basis. Thus, when the *client* delivers a class A packet to the MAC layer, it does not know, whether the packet will be transferred as a A0 or A1 packet.

invariant 3 $\forall nodes : offered(A) < R_{A0} \wedge R_{A1} = 0 \Rightarrow accepted(A0) = offered(A)$

invariant 4 $\forall nodes : offered(A) < R_{A1} \wedge R_{A0} = 0 \Rightarrow accepted(A1) = offered(A)$

invariant 5 $\forall nodes : offered(A) \geq R_{A0} \wedge R_{A1} = 0 \Rightarrow accepted(A0) = R_{A0}$

invariant 6 $\forall nodes : offered(A) \geq R_{A1} \wedge R_{A0} = 0 \Rightarrow accepted(A1) = R_{A1}$

invariant 7
$$\forall nodes : R_{A0} > 0 \wedge R_{A1} > 0 \wedge offered(A) < R_{A0} + R_{A1} \Rightarrow accepted(A0) + accepted(A1) = offered(A)$$

invariant 8
$$\forall nodes : R_{A0} > 0 \wedge R_{A1} > 0 \wedge offered(A) \geq R_{A0} + R_{A1} \Rightarrow accepted(A0) = R_{A0} \wedge accepted(A1) = R_{A1}$$

In invariants 1-8, we have worked with per ringlet invariants. This was because for class A traffic, the sum of traffic is calculated, regardless of the destination of the traffic transmitted. For the amount of class C (and *B-EIR*) traffic accepted however, this may vary on a per link basis and depends on the number of stations sending traffic over the same link, and their individual sending pattern. Thus, for the remaining invariants 9-12, it is more relevant to present per-link invariants, in order to express the relative priority of the RPR service classes.

The *B-CIR* portion of the accepted class B traffic, is restricted by the union of *shB* and *shD* shapers. The *B-EIR* portion as well as the amount of accepted class C traffic is controlled by union of *shF* and *shD* shapers. The rate setting of the *shF* shaper is controlled by the fairness algorithm.

In invariants 9 and 11, no portion of the offered class B (and class C) traffic passes a congested link, or if it does, the amount of offered traffic is equal to or lesser than rate constraints in effect over the congested link. Thus all the offered class B (and class C) traffic traversing this link is accepted.

invariant 9 $\forall links : offered(B) + accepted(A1) \leq R_U \Rightarrow accepted(B) = offered(B)$

In invariants 10 and 12, the link under observation is congested (the demand is greater than the capacity). Assuming that the class B (and class C) traffic traversing the link does not traverse a downstream link that is more congested, the amount of accepted class B (and class C) traffic equals the portion of the R_U bandwidth not already in use by $A1$ traffic.

invariant 10 $\forall links : offered(B) + accepted(A1) > R_U \Rightarrow accepted(B) = R_U - accepted(A1)$

invariant 11 $\forall links : offered(C) + accepted(A1) + accepted(B) \leq R_U \Rightarrow accepted(C) = offered(C)$

invariant 12 $\forall links : offered(C) + accepted(A1) + accepted(B) > R_U \Rightarrow accepted(C) = R_U - accepted(A1) - accepted(B)$

4.2 Delay Properties

The delay of class A traffic consists of access delay at the ingress point, transmission delay (in ingress and transit nodes), link propagation delay and queuing delay in the transit path. The transmission delay per node transited is fixed and dependent on the packet size and link capacity. The transit queue delay for class A packets is in the worst case the aggregate of times we have to wait for nodes in the transit path to finish transmission of a locally added data packet (as well as locally added control packets), since RPR does not support preemption of lower priority packets.

Access delay is measured from a packet is placed at head of line in the *client* queue until it is transmitted onto the link connected to the downstream node. When within its rate bounds, class A add traffic is prioritized over other add traffic, but has lower priority than transmission of idle, fairness and (high priority) control packets as well as transmission of packets from the PTQ. Thus when within its rate bounds, in the worst case, a class A packet can risk waiting several packet transmission times before being scheduled for transmission on the output link. Given a ring with N nodes, utilizing shortest path routing of packets, in the worst case, we can receive a packet train consisting of N/2 or N back-to-back MTU-sized class A packets. This observation builds on the assumption that upstream nodes are configured so that they are not able to transmit more than one (for a node supporting A0 traffic only) or two (for a node supporting both A0 and A1 traffic) back-to-back maximum sized class A packets and that they are not able to accumulate enough credits to insert more class A packet into the packet train. For the simplicity of the analysis, we also disregard the sending of control traffic.

Thus, based on the above assumptions and with L_{MTU} representing the bit size of a MTU sized packet and R_L representing the link rate, the worst case access delay, D_a, is given by (5) below. On average, with an offered load larger than the available capacity, the average access delay time for a class A packet

will be close to the time required to accumulate a sufficient amount of credits for the transmission of the packet.

$$D_a = \begin{cases} \frac{L_{MTU}}{R_{A0}} + \frac{N}{2} \cdot \frac{L_{MTU}}{R_L}, R_{A1} = 0 \wedge R_{A0} > 0 \\ \frac{L_{MTU}}{R_{A1}} + \frac{N}{2} \cdot \frac{L_{MTU}}{R_L}, R_{A1} > 0 \wedge R_{A0} = 0 \\ \frac{L_{MTU}}{R_{A1}} + N \cdot \frac{L_{MTU}}{R_L}, R_{A1} \geq R_{A0} > 0 \\ \frac{L_{MTU}}{R_{A0}} + N \cdot \frac{L_{MTU}}{R_L}, R_{A0} > R_{A1} > 0 \end{cases} \quad (5)$$

In (5), the first and second cases represents a scenario where only A0 or A1 traffic is supported in the network. Using the above assumptions, the worst case access delay will be when a packet arrives at head of line in the *client* queue right after the previous packet has been accepted by the MAC layer. Thus in the worst case, following the transmission of the previous packet, the A0 or A1 shaper must accumulate an amount of credits equal to the packet size. Then, once the accumulation of credits has finished, upon arrival to the MAC layer, the transmission has to await the transmission of a transiting packet train of class A traffic consisting of $N/2$ packets. That is, all upstream nodes contributes one class A0 or A1 packet to the packet train.

In (5), in the third and fourth cases, both A0 and A1 traffic is supported in the network. In this case, the worst case time for the accumulation of credits for the sending of a packet equals the maximum packet size divided by the highest service rate In this case however, a station is able to send two maximum sized packets back-to-back. As a result of this, the worst case length of a transiting packet train is the double of the cases where only A0 or A1 traffic is supported.

5 Proposal for DiffServ PHB Mappings in an RPR Network

When proposing a mapping between DiffServ PHBs and RPR service classes, we would like to remind the reader of the fundamental properties of the three RPR service classes discussed in section 2. In addition to a guaranteed and limited throughput, service class A class is characterized by low delay and jitter values. Class B traffic is characterized by guaranteed bandwidth and bounded delay and jitter values. Finally, class C is a best-effort/scavenger service class and has no associated guarantees.

What we propose, is to implement the three standard PHBs: Expedited Forwarding, Assured Forwarding and default PHB and to map these onto RPR service classes A, B and C. Given the the strict priority scheduling rules presented in section 4.2 and the various invariants used to maintain the per-service class rate configuration presented in section 4.1, this results in a minimum access delay as well as per node transit delay for class A traffic. Also, the above invariants ensures a guaranteed bandwidth share for both class A and class B traffic. Based on this, the implementation of an EF PHB for a DiffServ enabled RPR node by use of RPR service class A seems feasible.

RPR's service class B does provide guaranteed throughput and in-order deliver of packets during normal operation of the ring. To provide an AF conformant PHB based on RPR's service class B however, the *client* has to implement the relative packet drop priorities. The implementation of relative drop priorities in the *client* should be a relatively easy task, using some form of priority queueing scheduling algorithm. Thus achieving an AF compliant DiffServ implementation based on RPR seems feasible.

Finally, we have an obvious match between DiffServ's default PHB, specified in [13] as a best-effort forwarding behavior, and RPR's service class C. In an RPR network, when mapping DiffServ's default PHB to RPR's service class C, this will work in conformance with the requirements of DiffServ's default PHB. Following the presentation of simulation scenarios and results in section 7, in section 8, we will conclude on the conformance to the DiffServ PHB requirements for the proposed mapping between the DiffServ PHBs and RPR service classes. But first, in section 6, we show an analytical example using the proposed mapping.

6 Evaluation of RPR Service Class Differentiation by Analytical Example

In Fig. 2, we have shown the analytical resulting throughput when mapping the EF-, AF and default PHBs to RPR's service classes A, B and C.

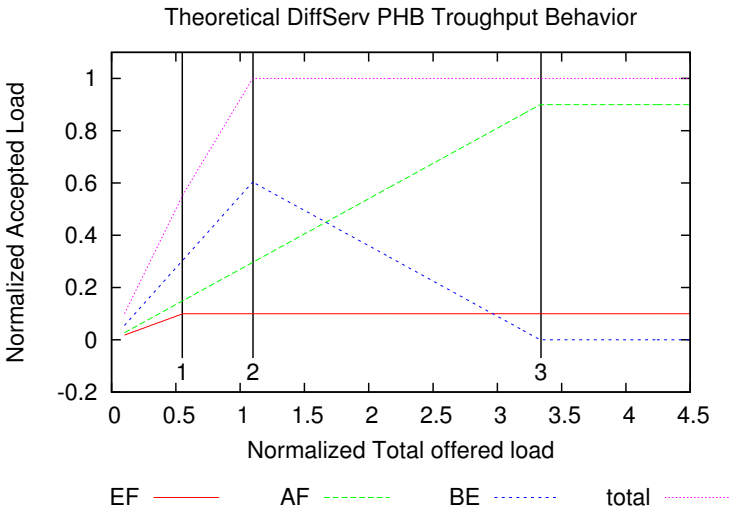


Fig. 2: Theoretical Throughput for the EF-, AF and default PHBs when mapped to RPR service classes A, B and C.

The analytical example uses the invariants introduced in section 4.1 to assign bandwidth to the DiffServ PHBs and plots the accepted amount of traffic for the three PHBs for a linearly increasing load level. In the example, we calculate the per PHB- as well as the total throughput for a single node aggregate, using all bandwidth resources on the ring-segment from the source- to the destination node. For scenarios with more than one node sending DiffServ traffic over a common ring segment, use of analytical models becomes too difficult, thus we use our simulator model to analyze the PHB behavior. For the analytical example, the initial load consist of 55% best-effort traffic, 27% AF traffic and 18% EF traffic. This represent a reasonable setting, where the majority of the traffic consist of low-priority (BE) traffic, and the remaining fraction of this is divided using a ratio of 3:2 between the amount of medium (AF)- and high-priority (EF) traffic offered. The aggregate of the initial load is lesser than the link-bandwidth. From this starting point, we increase the amount of traffic offered linearly for the three PHBs while maintaining the 55/27/18 ratio. The R_R fraction is set to 10% of the link bandwidth. The offered load is increased until the point where the division of link bandwidth remains at a constant level, regardless of any additional increases in the amount of offered traffic.

Three points for offered load in this plot are of particular interest. As long as we are on the left side of point 1, the aggregate EF traffic can be increased further while maintaining invariant 3 or 7. At point 1, we have reached the load level where an additional load increase causes the premise of both invariants 3 and 7 to be false. Beyond this point, the maintenance of invariant 8 effectively prohibits the acceptance of additional EF traffic. On the left side of point 2, as long as invariant 11 is maintained, an increase in offered BE traffic leads to an equivalent increase in accepted BE traffic. At point 2, we have reached the load level where an additional load increase causes the invariant's premise to be false. Beyond this point, the maintenance of invariant 12 effectively prohibits the acceptance of more BE traffic. Finally, between points 2 and 3, as long as invariants 9 and 12 are maintained, an increase in offered AF traffic leads to an equivalent increase in accepted AF traffic, on the expense of an equivalent decrease in accepted BE traffic. At point 3, we have reached the load level where an additional load increase causes the premise of invariant 9 to be false. Beyond this point, the maintenance of invariants 10 and 12 effectively prohibits the acceptance of additional AF traffic as well as excluding BE traffic from the link.

7 Performance Evaluation by Simulations

In section 6, the suitability of the Resilient Packet Ring's built-in mechanisms for service differentiation in a DiffServ environment was demonstrated analytically by a simple example. In this section, we use our RPR discrete event simulator model implemented in the OPNET Modeler environment to perform the same evaluation by use of some simulation scenarios. It is impossible to verify the behavior of RPR by simulation for all conceivable combinations of traffic, node configurations and ring configurations. So in this section, we analyze three se-

lected scenarios that we believe are of importance for the evaluation of RPR's performance when used in a DiffServ context, as well as in a general context, where an RPR network is used to provide traffic differentiation.

In particular, we want to show that for the scenarios evaluated, delay guarantees for high-priority traffic (RPR class A/DiffServ EF) are kept, as well as throughput guarantees for medium priority traffic (RPR class B/DiffServ AF). And lastly, but not least, we want to show that traffic of the BE PHB is able to utilize the bandwidth not used by the EF and AF PHBs.

The discussion of conformance to the DiffServ PHB requirements, is postponed to section 8, where we will discuss the conformance, based on results obtained from the full set of simulation scenarios.

In all the simulated scenarios, we evaluate the delay (for EF traffic only) and throughput performance of an RPR ring as a function of increasing load. For all experiments where we calculate confidence intervals for the delay, we use the median as the estimator for the 90th and 99th percentiles. We have not presented delay results for the AF and BE PHBs, as there are no associated DiffServ delay compliance requirements for these PHBs. Further, as traffic on the two ringlets in an RPR network is handled independently, we evaluate traffic flowing in one direction only.

The load offered is DiffServ EF, AF and BE traffic, using the mapping between DiffServ's PHBs and RPR service classes as proposed in section 5.

In the cases where we use self-similar packet sources, these are implemented based on a model outlined by Mondragón et al., using chaotic maps [18]. The implementation is done by by Blomskøld and Nilsen and is described in [19].

For the reminder of the experimental part, we start in section 7.1, with the evaluation of delay and throughput performance for a so-called "Hot Receiver" or "Hub" scenario. This may be the case when an RPR ring is used as an access ring for the aggregation of traffic, with one node (the "Hot Receiver" or "Hub") on the ring providing access to the backbone.

Then, in section 7.2, we evaluate a scenario constructed to resemble an RPR ring used as a backbone ring.

Finally, in section 7.3, we perform an extensive series of tests for a set of randomly selected sender-receiver configurations.

7.1 Hot Receiver Scenario

In this experiment, we want to evaluate a configuration, where the majority of traffic on a ring(let) is destined for one particular node, a so-called "Hot-Receiver" or "Hub". This is typically the case in an access or metro-ring where we have one gateway, connecting the ring to a metro-ring or the backbone.

The scenario we use is shown in Fig. 3, where nodes 1-26 all send traffic to node 30. These nodes all send traffic of the DiffServ AF and BE PHBs, mapped onto RPR service classes B and C as proposed in section 5.

To make the scenario more realistic, we have added self-similar background traffic, entering the ring at node 0 and traversing the whole segment under observation. We model traffic flowing in one direction only. In a hot-receiver

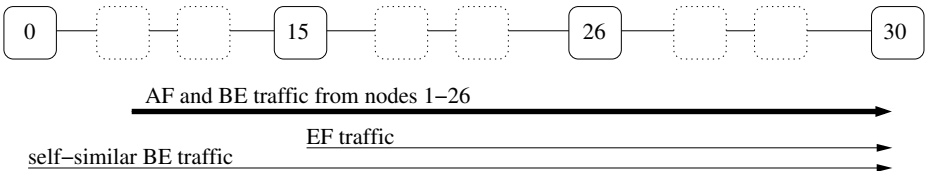


Fig. 3: A hot receiver scenario for a 64 node ring, with nodes 0-26 sending data to node 30. Node 0 sends self-similar BE traffic only. Nodes 1-26 all send AF and BE traffic. Node 15 also sends EF traffic.

scenario, in the reverse-direction, we will have a one-to-many configuration of flows, limited effectively by the hot-receiver's immediate downstream link as the load increases. Thus the likelihood of congestion in the reverse direction is much smaller than in the forward direction.

For the interested reader, Appendix A shows a detailed overview of the parameter settings used for our experiment.

We run a set of simulations for a series of load-points, and present a subset of the obtained data (i.e. throughput and delay as a function of offered load). The relative division of offered traffic of the EF, AF and BE traffic classes is approximately 2/40/60 (EF/AF/BE percentages of aggregate offered load not considering the contribution from the self-similar traffic source). Given the base load settings as given in Appendix A, as the relative offered load increases, the network runs relatively rapidly into the congested area of operation, while it takes somewhat longer (e.g. a higher relative load value) before the amount of accepted EF traffic reaches its maximum value.

We model the traffic from nodes 1-26 so that it has as relatively fixed (CBR-like) communications demand on the ring that is a function of the relative load value. That is, for a simulation run, the expected average rate equals the base load times the relative load parameter. To introduce some variability however, the inter-arrival times of packets entering a node are drawn independently according to a Poisson distribution. To provoke worst-case jitter for the EF traffic, we let all nodes send traffic to node 30.

By experience, we know that for a reasonable mix of traffic of the various priorities, the access-delay part for high-priority traffic contributes heavily to its jitter values. Thus, we use a probe flow of high-priority traffic, which is inserted at a node, located in the middle of the congestion domain and that traverses all downstream nodes in the congestion domain. By this, we ensure that at the ingress point of the high-priority traffic, there will be a significant portion of transit traffic, competing for the capacity of the node's out-link. Furthermore, we have a significant share of AF traffic, not constrained by the RPR fairness algorithm. Finally, at the tail of the congestion domain (node 0), we have a self-similar source sending low-priority traffic, ensuring large variations in demand, especially at low values of aggregate load from the CBR-like sources. At higher loads, the significance of the self-similar traffic is less, as it is strongly throttled by the RPR fairness algorithm.

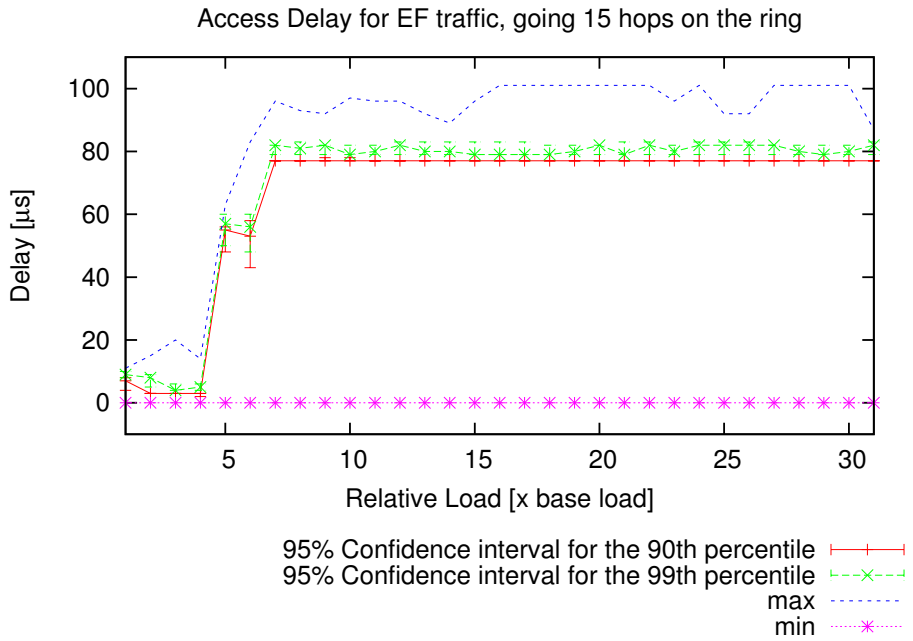


Fig. 4: Access Delay for EF traffic going 15 hops on the ring for the "Hot Receiver" scenario. The worst-case jitter is 101[μs]. As the offered load increases, so does the delay. This is because packets must wait longer at the head of the client queue before a sufficient amount of A0 or A1 credits are available for the sending of the packet. Note that from the point where the relative load equals 10, the 95% confidence interval for the 90th percentile is a straight line. This is because the lower 90% of the delay measurement does not change significantly. For the upper 10% of the measurements, there are some minor differences, leading to some variations for the 95% confidence interval for the 99th percentile. Due to some bursts of high-priority control data related to conservative mode delay measurements from node 15, the maximum delay values are somewhat higher than the 99th percentiles.

The delay measurements of EF traffic inserted at node 15, destined for node 30 are shown in Fig. 4. The figure shows maximum and minimum delays as well as 95% confidence intervals for the 90th and the 99th percentile.

In this scenario, we are serving EF traffic at a rate $R = 100$[Mbit/s], assigning $R/2$ to respectively A0 and A1 traffic, then, according to (5), the worst case analytical access delay is $\frac{L_{MTU}}{R_{A1}} + N \cdot \frac{L_{MTU}}{R_L}, R_{A1} \geq R_{A0}$. In this experiment however, except for high priority control traffic related to the RPR conservative fairness mode from upstream nodes, upstream nodes do not send neither class A nor EF traffic. Thus in this case, N is set to 2, and the worst case analytical access delay becomes $\frac{L_{MTU}}{R_{A1}} = \frac{500 \cdot 8}{50 \cdot 10^6} + 2 \cdot \frac{500 \cdot 8}{10^9} = 88[\mu s]$.

As seen in the figure, the maximum jitter is 101[μs], which is slightly above the analytical worst-case expression provided in (5). This is caused by some bursts of high-priority control data related to conservative mode delay measurements from node 15. The delay incurred on the the high-priority traffic however, can be reduced by setting the rate restriction for control traffic to a lower value (i.e. limit the burstiness of the control traffic).

The 95% confidence intervals for the 90th and 99th percentiles for the delay, for high values of offered load, are centered right below and above 80[μs] respectively.

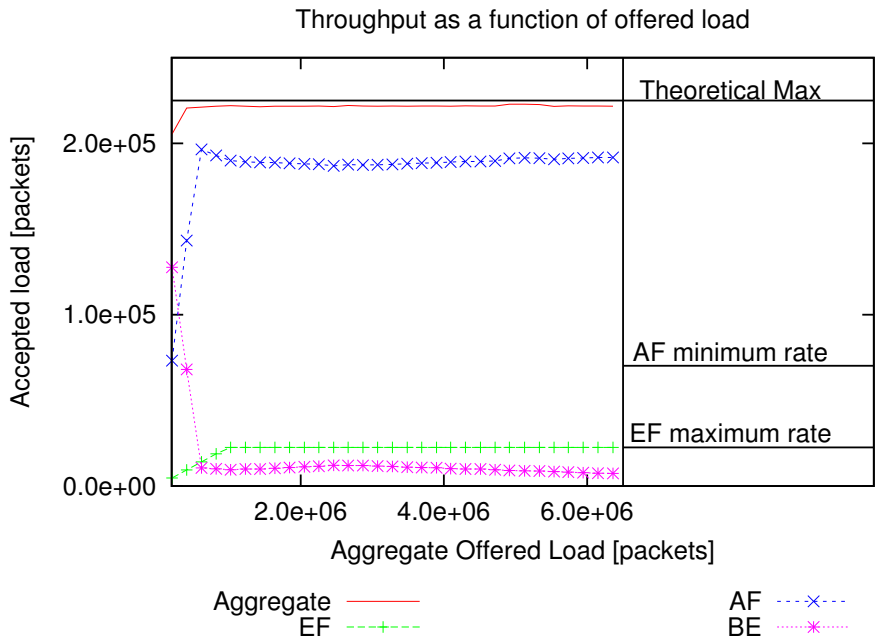


Fig. 5: Accepted load as a function of offered load for the EF, AF and BE PHBs for the "Hot Receiver" scenario. On the right-hand side of the plot, the configured EF maximum and AF minimum rates are shown as well as the theoretical maximum for the given configuration and topology. The line labelled "AF Minimum rate" represents the minimum ordinate value that must be reached for accepted AF traffic as the amount of offered AF traffic reaches or exceeds the same value. As seen, the amount of AF traffic exceeds by far this level as the offered load increases, thus AF throughput guarantees are met.

Throughput results for the EF, AF and BE PHBs are shown in Fig. 5. The figure presents the throughput as the aggregate number of packets accepted as well as the number of packet accepted per PHB as a function of the aggregate offered load. That is, for the range of relative load values presented in Appendix

A, for each value of the relative load parameter, the expected average offered load for each node equals the relative load parameter multiplied with the base load per PHB. The points plotted for accepted load are mean values, obtained from 12 independent simulation runs. The "noise" on the curves, is mainly caused by the self-similar traffic source, which for the given configuration, has an expected average throughput which equals the relative load times the base load. For different seed values however, the short term throughput of this packet generator will vary greatly. Thus introducing some "noise" on measured BE (and AF) traffic.

As seen in the figure, the amount of accepted EF traffic equals the amount of offered EF traffic as long as the amount of EF traffic is below the configured maximum. When the amount of offered EF traffic exceeds the configured maximum, the amount of accepted EF traffic is limited to the configured maximum (as given by the configuration in Appendix A).

For AF traffic, it is clear that the amount of AF traffic is not restricted to the value given by the configured AF rate given by the configuration in Appendix A. This is however not the intention. As specified by the RPR standard, AF traffic (when mapped to RPR service class B) has precedence[7] over BE traffic (when mapped to RPR service class C). Thus, as specified by invariant 12, when the sum of AF and BE traffic over a link is larger than the link can support, the BE traffic will only receive the remaining portion when the AF demand is satisfied. This is clearly shown in Fig. 5. As the aggregate offered load increases, the aggregate demand of AF traffic is satisfied on the expense of BE traffic. For high values of offered load, the small fraction of accepted BE traffic is most likely caused by BE traffic transmitted during the initial transient period, where the fairness algorithm has not yet converged.

7.2 Backbone/Metro Ring

In this experiment, we will consider a scenario where an RPR ring is operating as a large metro or backbone network. In this scenario, each RPR node on the ring serves as a ring ingress/egress point, handling traffic of the three PHBs EF, AF and BE.

In this scenario, illustrated in Fig. 6, let us assume that the EF and AF PHBs are used as a service for providing low-latency (EF only) and guaranteed throughput (both EF and AF) for communication between fixed node-pairs on the ring. This may be the case where a company subscribes to a gold and silver service from a service provider, to enable real-time (e.g. voice and video conferencing) as well as high-quality multimedia-traffic between two (or more) geographically distributed locations.

We model the EF traffic as traffic with a relatively fixed (CBR-like) communications demand on the ring, but to introduce some variability, the inter-arrival times of EF packets entering a node is drawn independently according to a Pois-

[7] On the ring ingress point.

son distribution. To provoke worst-case jitter for the EF traffic, we let all nodes send EF traffic transiting node 24..

The AF traffic is expected to be more bursty (e.g. variable rate multimedia traffic) and is modelled as self-similar traffic. The destination for AF traffic is chosen randomly according to a uniform distribution at simulation start-up, so that that the hop-count of the per node AF traffic will be in the range $\langle min, max \rangle$ (these are configuration settings specified in Appendix B). Once the destination is chosen, it remains fixed for the reminder of the simulation run. Note that for all destinations drawn, the traffic is sent on ringlet 0.

For BE traffic, we do not expect a fixed communication pattern on the ring, rather it is reasonable to assume a long-term behavior, where for traffic entering a ring node, the egress point of this traffic will be distributed in a uniformly manner between the various egress points on the ring. In the short term, we will typically have a behavior termed "source locality" [20], where, if we observe an arbitrary packet on the ring, there is a very high probability that the next packet on the ring will have the same source and destination address. Thus for a burst of packets entering the ring from an ingress point, a number of the packets in the burst is expected to have the same ring egress point.

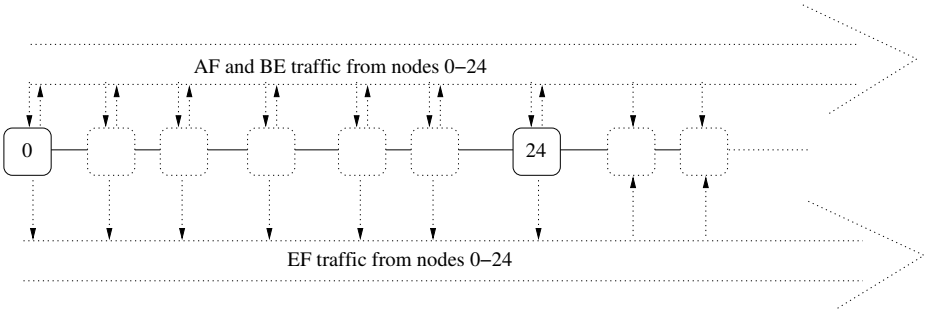


Fig. 6: A "Backbone/Metro Ring" scenario for a 64 node ring, with nodes 0-24 sending data to various nodes on ringlet 0 (i.e. the ringlet with data flowing to the right in the figure). For each simulation run, All EF traffic is sent to random destinations located beyond node 24. All AF traffic is sent a minimum number of hops (according to the configuration parameters), to a destination selected according to a uniform distribution at simulation startup. For BE traffic, the destination of traffic is selected so that the traffic is sent a minimum number of hops. The destination is selected according to a uniform distribution at the start of each on-period of the self-similar generator.

Just as for AF traffic, we also expect the BE traffic to be self-similar, with great variations in throughput, both on a short time-scale and a long time-scale. To obtain the "source locality" behavior of the BE traffic, the destination address is kept fixed for the duration of an on-period of the self-similar packet source, before we, at the end of the on-period, draw the destination address to use for

the next packet burst uniformly (in the same manner as is done at simulation startup for the AF traffic).

To provoke worst case access delays for EF traffic, we use a traffic mix with a relatively high maximum share of EF traffic (10% of the line rate). For the base load used, we have a 10/20/70 division for the offered traffic of the EF/AF/BE PHBs.

For the interested reader, Appendix B shows a detailed overview of the parameter settings used for our experiment.

If we start by evaluating the delay performance of the EF traffic, we find the worst case jitter for EF traffic entering the ring at node 10. As is shown in Fig. 7 (and with an excerpt showing the details for high values of offered load in Fig. 8), as the offered load increases, so does the access delay of the EF traffic. But as expected, as the offered load increases, the maximum delay stabilizes at an upper bound, close to time it takes to accumulate an amount of credits, sufficient for the transmission of a maximum sized package.

The results obtained fall within the upper bound provided by the worst-case analytical expression given in (5). In this expression, for a network supporting both A0 and A1 traffic, the worst case access delay is specified as $\frac{L_{MTU}}{R_{A1}} + N \cdot \frac{L_{MTU}}{R_L}, R_{A1} \geq R_{A0}$. In this experiment, with a 64 node ring, $N = 64$, however, as there are only 25 active nodes, the value to use for N is 50. Thus we get the worst case analytical access delay: $D_a = \frac{500 \cdot 8}{2 \cdot 10^6} + 50 \cdot \frac{500 \cdot 8}{10^9} = 2.2ms$

In the results obtained, the worst case access delay is measured to 2.121ms, a value well within the predicted worst case.

If we consider the throughput results shown in Fig. 9, the behavior is as expected. As the offered EF traffic is increased, as given by invariant 7, the accepted EF traffic increases in an equivalent fashion as long as the offered load stays below the (class A0 and class A1) rate constraints in effect. This is shown in the region in the plot to the left of the vertical line labelled 1. Then, to the right of this line, the amount of offered EF traffic exceed the rate constraints specified for A0 and A1 traffic, thus as specified in invariant 8, no additional amounts of EF traffic is accepted.

Similarly, for BE traffic, as specified in invariant 11, as long as there are no congested links on the ring, the amount of accepted BE traffic increases linearly with the offered amount of BE traffic. However, as links starts to become congested, as specified in invariant 12, the BE traffic gets the remaining bandwidth not used by EF and AF traffic. The vertical line labelled 2 marks the point where, to accommodate an increase in demand for AF capacity, the amount of BE traffic accepted must be correspondingly reduced.

Finally, the vertical line labelled 3 marks the point where the ring starts to become saturated. To maintain invariant 10 as the offered AF traffic increases, the ring is no longer able to support a linear growth rate of accepted AF traffic.

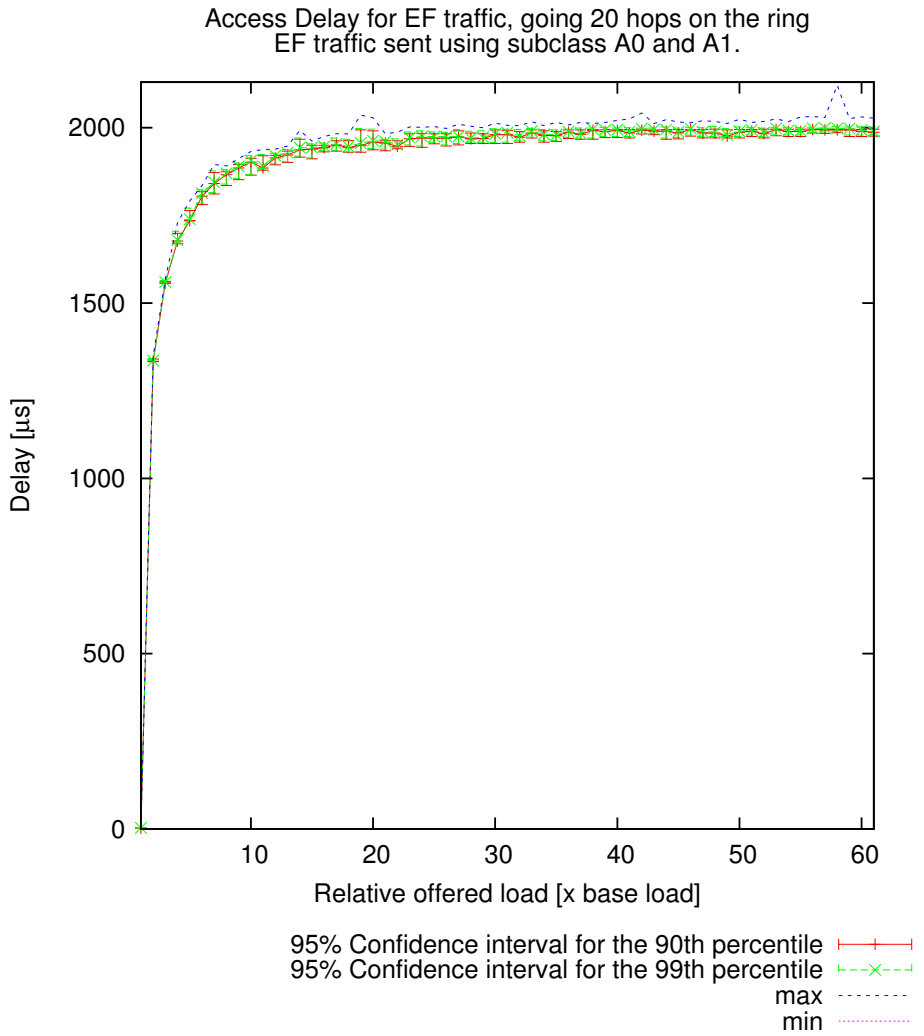


Fig. 7: Worst case Access delay for the worst case "Backbone/Metro Ring" scenario is is found for EF traffic entering the ring at node 10. The worst-case jitter is found for the value of 57 for the relative offered load, and is $2121[\mu s]$. *As is seen from the plot, when the offered load increases, the 90 and 99th percentiles for the access delay converges towards the time it takes to accumulate credits for the transmission of an EF packet, using subclass A0 or A1. The maximum delay values are caused by delayed access to the output link, caused mainly by a train of EF packets from upstream nodes transiting the node. However, the sending of control traffic also contributes to the deviation. Fig. 8 contains an excerpt of these results, showing more details for the delay results for high values of offered load.*

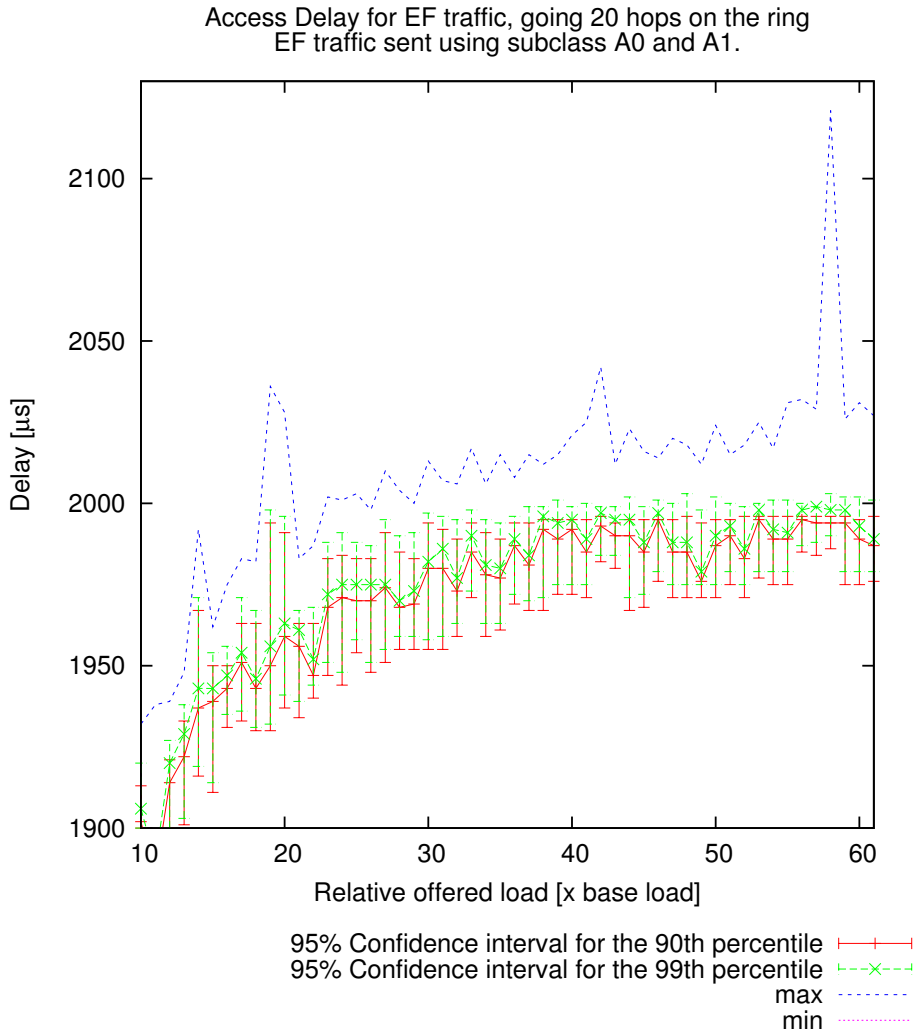


Fig. 8: Excerpt of results shown in Fig. 7 for the "Backbone/Metro Ring" scenario. As seen, the 95% confidence intervals for the 90th and 99th percentiles are rather close. The maximum delay value at a value of 57 for relative offered load, is caused by a number of back-to-back EF packets transmitted by upstream nodes at simulation startup in addition to a burst of control packets.

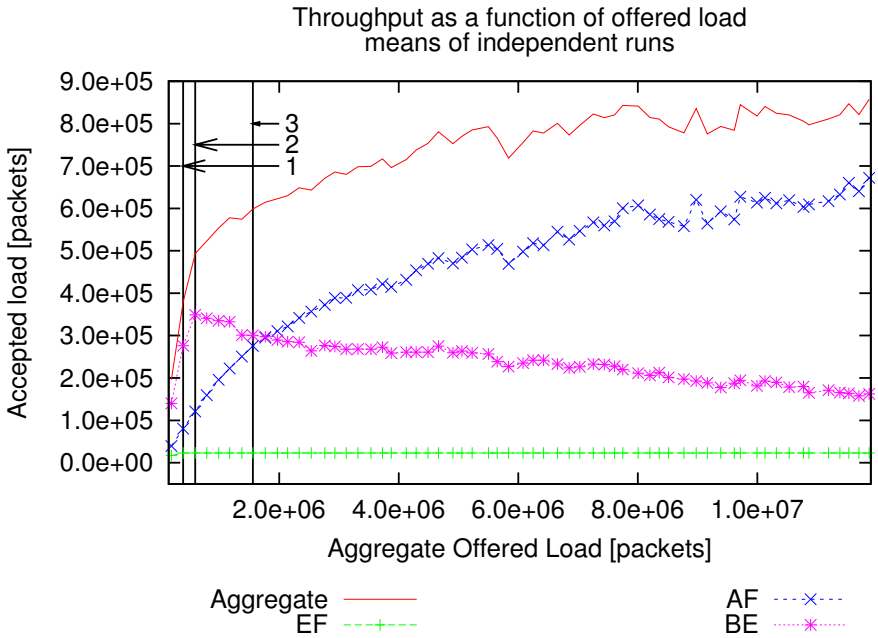


Fig. 9: Average throughput as a function of offered load for traffic of the three PHBs EF, AF and BE for the "Backbone/Metro Ring" scenario. The figure also shows the aggregate throughput. At point 1, the aggregate of accepted EF traffic has reached its maximum value. At point 2, some links start to become congested, thus to accept more AF traffic, the amount of BE traffic accepted must be reduced. Finally, at point 3, the ring starts to become saturated as more links become congested, thus the increase of accepted AF traffic is no longer a linear function with respect to the offered load. The "noise" on the curves is caused by the self-similar traffic generators which, depending on the seed value of the simulation run (and otherwise equal parameter settings), will produce an average amount of offered traffic which varies.

7.3 Random Fixed Pairs Communication

In the "Hot Receiver" and "Backbone/Metro Ring" scenarios, described in sections 7.1 and 7.2, we measured the time taken from a packet was available at the head of the *client* queue, until it was scheduled for transmission on the output link. This performance metric is essential in order to claim conformance to the DiffServ EF requirements.

However, given that an EF packet is transmitted onto the ring, it is also important that it is not delayed significantly on its transmit path from its ring ingress to its ring egress point.

In this experiment, we want to measure the transit path delay, starting the measurement as a packet is accepted by the MAC layer at the ingress point (i.e. put into the stage queue), and stopping the measurement as the packet is stripped from the ring at its egress point.

To ensure that there will be a competition for network resources, we use a scenario where for a ring of 16 nodes, we let each node transmit traffic to one other node on the ring. Additionally, we let each node receive traffic from one other node on the ring. Further, a node is not allowed to send traffic to itself. Thus we get a configuration with 16 pairs of communicating nodes. The communicating pairs are are randomly selected on beforehand, and 16 different configurations are generated.

For each simulation an offered load parameter is given. 20 load values are used, linearly distributed on a scale from 1 to 10 times the base load. In order to calculate confidence intervals for the delay, 12 independent simulation runs are performed per load-point for each configuration.

The base load uses the same 55/27/18 ratio for offered load of the EF/AF/BE PHBs as is used in section 6. The traffic is modelled in the same CBR-like fashion as is used in the two previous scenarios, where the inter-arrival time of packets are drawn independently according to a Poisson distribution. Thus resulting in an expected average throughput per relative load value, with some variation in the arrival time of the packets at the ingress nodes.

For the interested reader, detailed configuration information for this scenario can be found in Appendix C.

The delay measurements for this scenario are shown in Fig. 10. In the plot, we observe that the variations in delay values are relatively insignificant compared to their actual values. The worst case jitter is 33[μs], which constitutes 1.85% of the minimum delay value for a distance of 7 hops on the ring. One interesting observation is that with a lower offered load, the delay values for the 90th and 99th percentiles are slightly higher than for higher offered loads. We believe this is because at low values of offered loads, there is a larger chance that a congestion domain has not been established on the ring. Without one (or several) assigned node(s) (i.e. congestion domain head) working to constrain the amount of BE traffic accepted onto the ring, there is a greater chance for an EF packet at the ingress point or in the transit path having to await the completion of a BE packet being transmitted. However, the majority of variations are relatively small, as can be observed by the size and locations of the confidence intervals.

This confirms the expected behavior for EF delay, which is expected to be limited upwards, as the load increases. Both in terms of the access delay at the ingress point (with the analytical worst case expression provided in (5)), as well as the delay incurred on the packets as they propagates on their path between their ingress and egress points on the ring.

Fig. 11 plots the throughput for the three PHBs EF, AF and BE, as well as the aggregate of all PHBs. For each value of offered load, each point on the curves is the mean taken across all the 16 sender-receiver configurations and their independent simulation runs.

When looking at the throughput-performance of the ring, the behavior is not as clear-cut as that of a single link, as illustrated in Fig. 2. The general trend however follows the theoretical pattern, given by the invariants specified in section 4.1. Thus, as the offered load increases, the increase in accepted EF traffic is linear up to some point and then remains at a constant level. For both AF- and BE traffic, the accepted load increase linearly initially, and as some links get congested, the amount of BE traffic accepted has to be decreased to allow for an increase in accepted AF traffic. Up to an total offered load of $\sim 0.8 \cdot 10^6$ (marked with the vertical line labelled 1), the accepted traffic for all PHBs increases linearly as a function of offered load. At this point however, the amount of offered EF traffic has reached the point where a further increase in accepted EF traffic would violate invariant 8, thus no further increases in accepted EF traffic is allowed. For BE traffic, at an offered-load of $\sim 10^6$ (marked with the vertical line labelled 2), some links in the simulated sender-receiver pair configurations have become congested, resulting in a stop in the increase of accepted BE traffic (to give room for the increased amount of AF traffic and hence maintain invariant 12). From this point onwards, sufficiently many links have become congested, resulting in a clear decrease in the accepted amount of BE traffic while maintaining a linear increase in the accepted amount AF traffic.

As the offered load exceeds a level of $\sim 2.5 \cdot 10^6$ (marked with the vertical line labelled 3), to maintain invariant 10 as more links utilize all their capacity for the sending of EF and AF traffic, the growth rate of accepted AF traffic starts to decrease slowly towards 0.

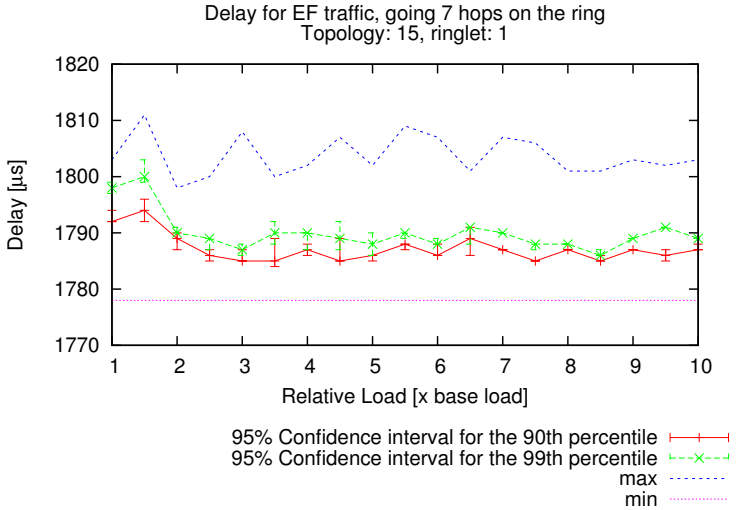


Fig. 10: Delay measurements for the "Random Pairs" scenario. The plot shows max and min delays as well as the 95% confidence intervals for the 90th and 99th percentiles. Worst case jitter (max-min) for Random Fixed Pair Scenario is measured to 33[µs] at relative load 1.5 for "topology" 15, for traffic going 7 hops on ringlet 1. Note that in this plot, the delay consists of the waiting time in the stage queue at the ingress node, as well as propagation and queueing delay in the transit path. With a lower offered load, there is a larger chance that a congestion domain has not been established on the ring. Thus without one (or several) assigned node(s) working to constrain the amount of BE traffic accepted onto the ring, there may be a greater chance for an EF packet at the ingress point or in the transit path having to await the completion of a BE packet being transmitted. Note however that the difference is in the order of 10-20[µs] and is insignificant compared to total delay experienced by a packet. Another interesting observation is that the delay does not follow a monotonically increasing behavior as the offered load increases. Rather, it varies around a "steady-state" value. The majority of variations are very small however, as can be observed by the size of the confidence intervals. This confirms the expected behavior for EF delay, which is expected to be limited upwards, as the load increases. Both in terms of the access delay at the ingress point (with the analytical worst case expression provided in (5)), as well as the delay incurred on the packets as they propagate on their path between their ingress and egress points on the ring.

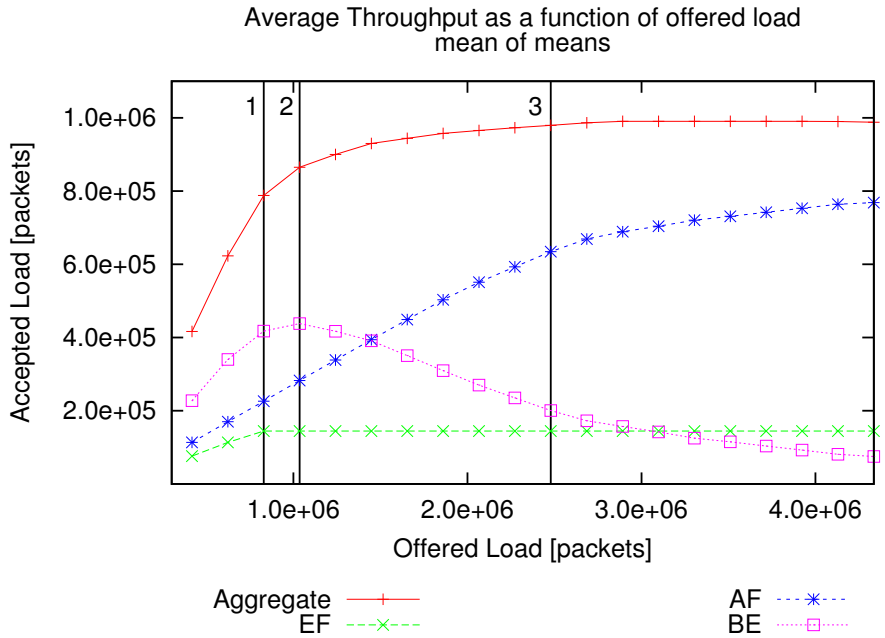


Fig. 11: Accepted vs. total offered load for traffic of the DiffServ PHBs EF, AF and BE for the "Random Pairs" scenario. The plot shows the mean of mean-values for all the independent sender-receiver configurations. That is, for each sender-receiver configuration, a number of independent simulations are run for each value of offered load. For each value of offered load, the mean is taken across all the 16 sender-receiver configurations and their independent simulation runs. Note the smoothness of this curve when compared to the throughput curves shown in figures 5 and 9. This is due to the type of traffic generators used. In this experiment, we use CBR-like traffic generators, producing a relatively similar aggregate offered load for different seed values. As a result of this, the accepted load varies more smoothly as the offered load is varied.

8 Conformance to DiffServ PHB Requirements

In section 3, the conformance requirements for the DiffServ PHBs EF, AF and BE (default) were presented. Having presented both analytical and simulation results evaluating throughput and delay performance of the proposed PHB mappings, we will now discuss how well our simulation results meet the DiffServ conformance requirements. We start by evaluating conformance to the DiffServ EF requirements in section 8.1, before evaluating conformance to the DiffServ AF requirements in section 8.2. Then in section 8.3, we end by evaluating conformance to the DiffServ BE requirements.

8.1 EF Conformance

For DiffServ EF traffic, we start by analyzing the results obtained for access delays, presented for the "Hot Receiver" scenario, presented in section 7.1 and the "Backbone/Metro Ring" scenario, presented in 7.2.

When considering the EF delay conformance requirements, one term in the expression provided in (2) is of particular interest:

In (2), the fraction $\frac{L_j}{R}$ accounts for the ideal per packet transmission delay for an EF PHB with a committed service rate of R, transmitting a packet j of length L_j. In our scenario, with a maximum packet size of $L_{max} = 500$[bytes], this accounts for a delay component of $\frac{500 \cdot 8}{R}$. The question then becomes, for an RPR network, what is the maximum committed DiffServ EF rate R that can be supported?

With a configuration where EF traffic is transported using RPR subclasses A0 and A1, at high loads when traffic is backlogged, the worst case access delay component is due to the accumulation of credits for the sending of a L_{max} sized packet. This component, which we term D_{ac}, is shown in (6) below.

$$D_{ac} = min\left(\frac{L_{max}}{R_{A0}}, \frac{L_{max}}{R_{A1}}\right) = L_{max} \cdot min\left(\frac{1}{R_{A0}}, \frac{1}{R_{A1}}\right) \tag{6}$$

Knowing that D_{ac} is not the only component of the access delay, D_a, clearly, we cannot commit to a service R, that requires a maximum value of the access delay, lower than D_{ac}. Thus we get a relation between the maximum committed EF service rate, R, and the credit accumulation component of the access delay as shown in (7) below.

$$R < \frac{L_{max}}{D_{ac}} = max(R_{A0}, R_{A1}) \tag{7}$$

Thus in our "Hot Receiver" scenario, with $R_{A0} = R_{A1} = 50$ [Mbit/s], we have $R < R_{A0} = 50$[Mbit/s].

In the same scenario, if we were to transport EF traffic using A0 capacity only (i.e. $R_{A0} = 100$[Mbit/s] and $R_{A1} = 0$) then clearly, $R < R_{A0} = 100$[Mbit/s].

Thus in the two configuration alternatives, we have the same aggregate bandwidth (100 [Mbit/s]) available for high-priority traffic. However, in the first configuration, the committed bandwidth R of a DiffServ EF conformant service is limited upwards to half that of the second configuration.

For the "Hot Receiver" scenario, serving EF traffic at a service rate $R = 50 + 50 = 100$[Mbit/s], the worst case access delay was found to be 101[μs]. Although slightly higher than the worst case D_{ac} of 80[μs], the results obtained indicates that by enforcing a somewhat stricter rate limitations on control traffic, it is possible to provide a DiffServ compliant EF PHB at a committed rate R, close to the A0 and A1 rate, that is 50Mbit/s.

Similarly, for the access delay results obtained for the "Backbone/Metro Ring" scenario, the worst-case value of 2121[μs] is clearly within the analytical worst-case of 2200[μs]. And as such, these results strengthen the above claim that it possible to provide a DiffServ compliant EF PHB offering a rate close to the A0 and A1 rate, which in the case of the "Backbone/Metro Ring" scenario was 2[Mbit/s].

However, the results for access delay alone, is not sufficient to conclude on the suitability of RPR for the provisioning of a DiffServ compliant EF PHB.

Thus we proceed to the study of throughput results obtained, starting with the "Hot Receiver" scenario.

In this scenario, the per-node aggregate rate of class A ($A0$ and $A1$) traffic is set to 10% of the line rate. The duration of the simulation runs constituting the throughput results in Fig. 5 is 0.9[s]. Thus, with $N = 1$ node, $R_L = 10^9$[bits/sec], $t_{sim} = 0.9$[s] and $L_p = 500$[bytes] , this corresponds to: $\sum_{A\ traffic} = \frac{N \cdot (R_{A0}+R_{A1}) \cdot R_L \cdot t_{sim}}{L_p} = \frac{1 \cdot 0.1 \cdot 10^9 \cdot 0.9}{4000} = 22500$[packets]. For all points of relative offered load, where the aggregate offered load exceeds 22500[packets], the aggregate accepted EF load is restricted to a value ~ 22530[packets].

Similarly, for the "Backbone/Metro Ring" scenario, the per-node aggregate rate of class A ($A0$ and $A1$) traffic is set to 0.4% of the line rate. The duration of the simulation runs constituting the throughput results in Fig. 9 is 0.9[s]. Thus, with $N = 25$ nodes, $R_L = 10^9$[bits/sec], $t_{sim} = 0.9$[s] and $L_p = 500$[bytes], this corresponds to: $\sum_{A\ traffic} = \frac{N \cdot (R_{A0}+R_{A1}) \cdot R_L \cdot t_{sim}}{L_p} = \frac{25 \cdot 0.004 \cdot 10^9 \cdot 0.9}{4000} = 22500$[packets]. For all points of relative offered load, where the aggregate offered load exceeds 22500[packets], the aggregate accepted EF load is restricted to a value ~ 22780[packets].

Lastly, for the "Random Fixed Pairs" scenario, the per-node aggregate rate of class A ($A0$ and $A1$) traffic is set to 4% of the line rate. The duration of the simulation runs constituting the throughput results in Fig. 11 is 0.9[s]. Thus, with $N = 16$ nodes, $R_L = 10^9$[bits/sec], $t_{sim} = 0.9$[s] and $L_p = 500$[bytes], this corresponds to: $\sum_{A\ traffic} = \frac{N \cdot (R_{A0}+R_{A1}) \cdot R_L \cdot t_{sim}}{L_p} = \frac{16 \cdot 0.04 \cdot 10^9 \cdot 0.9}{4000} = 144000$[packets]. For all points of relative offered load, where the aggregate offered load exceeds 144000[packets], the aggregate accepted EF load is restricted to a value ~ 144500[packets].

Thus in sum, all the throughput results support the claim, that the use of RPR's A0 and A1 subclasses meets the throughput conformance requirements of DiffServ's EF PHB.

8.2 AF Conformance

When studying the conformance requirements of the AF PHB, we see clearly from both the analytical results in Fig. 2 and the simulation results in Fig. 11, that as long as the ring is not saturated, the AF traffic is provided the requested throughput on the expense of BE traffic. Clearly, this is in conformance with the AF and default PHB requirements.

Also, although not supported by simulation results as this is not an issue in our simulation model, in-order delivery of AF packets is guaranteed by the RPR standard under normal operation of the ring.

However, as noted in section 5, the relative packet drop priorities have to be implemented in the *client*.

8.3 BE Conformance

Conformance to the default PHB is clearly obtained, as most of the bandwidth not utilized by the EF and AF traffic, is utilized by the BE traffic. Given the operation of the RPR fairness algorithm, 100% link utilization on all links in an RPR network is difficult to obtain (as is the case for any congestion control algorithm working concurrently to maximize throughput and minimize delay).

9 Related Work

IETF has created a working group named IP over Resilient Packet Rings (iporpr) chartered to investigate the problem area of *"developing the necessary standards for efficient interaction between L2 and L3"*. Currently, there is a preliminary Internet draft available (first version published July 10, 2005) from the IETF IPORPR Working Group[8]. The main focus of the draft however, is the detailed mapping proposal between various DiffServ PHBs (as well as mapping for MPLS traffic) and RPR service classes. The Internet draft has no discussions on the fulfillment of DiffServ PHB conformance requirements, neither does it address the implementation of various drop priorities for the AF PHB group.

Apart from the ongoing work in the IETF IPORPR working group, we have only found one other paper addressing the problem of using RPR in a DiffServ context. One of the problems with the work by Wang et al. [21] is that a proposal is made to map the DiffServ AF PHB to RPR's A1 service class and to map the DiffServ EF PHB to RPR's A0 service class. By this, the use of the 1TB node architecture is effectively prohibited. Further, for a node using the 2TB-architecture, the design of the RPR MAC is done so that the *client* never knows

[8] http://www.ietf.org/html.charters/iporpr-charter.html

whether a high-priority packet is sent as an A0 or an A1 packet. Thus with the mapping proposed, one will never know, whether an EF or AF packet will be transported using RPR service classes A0 and A1. In effect this makes it impossible to provide a clear service differentiation between DiffServ traffic of the EF and AF PHBs, traversing an RPR network.

10 Conclusion

In this paper we have evaluated the suitability for use of RPR in a DiffServ environment. We have discussed the fundamental mechanisms used in RPR to perform rate control according to given constraints, of which some are given by the network topology (link rates, propagation delays, number of nodes), some are statically configured (rate settings for traffic classes A and B) and some are configured dynamically (rate constraints for classes *B-EIR* and C).

We have also introduced a set of invariants which specify important parts of the RPR traffic class priorities and rate controls. Further, we proposed and discussed a simple mapping proposal between the RPR traffic classes and some standardized DiffServ PHB groups. Based on the formal invariants and the proposed mapping, an analytical model of a single RPR flow between two nodes was developed and analyzed.

We also performed a comprehensive test of the proposed mapping by use of three different simulation scenarios utilizing the RPR conservative fairness mode. In the first two, we focused on the evaluation of access delay for DiffServ EF traffic as well as throughput performance of DiffServ EF, AF and BE (default) traffic.

In the last simulation scenario, we evaluated delay incurred on DiffServ EF traffic, starting the measurement at the time an EF packet is delivered to the MAC layer at its ring ingress node, and stopping the measurement when the packet was stripped at its ring egress node. In this scenario, we also evaluated throughput performance of DiffServ EF, AF and BE traffic.

The results obtained were compared to and discussed by use of our established analytical expressions.

Finally, in section 8, we discussed how well the obtained results adhered to the conformance requirements for the DiffServ EF, AF and BE PHBs.

In concluding, we found that most of the achieved access delay measurements support our analytical worst case expressions. However, care must be taken in the handling of control traffic, to avoid large bursts of high priority control data. Further, we found that when utilizing both RPR subclasses A0 and A1 (with corresponding rate constraints R_{A0} and R_{A1}) for the transport of DiffServ EF traffic, the committed EF rate R is limited upwards so that $R < max(R_{A0}, R_{A1})$. In the case were only one of the two subclasses is used, $R < R_{AX},\ X \in \{A0, A1\}$.

In the last scenario, we also found that EF traffic was not significantly delayed on its path between its ingress and egress points on the ring.

Further, all our throughput results indicated that our proposed mapping conforms to the DiffServ conformance requirements for the DiffServ EF, AF and BE PHBs.

However, as noted in section 5, for the provisioning of an AF conformant DiffServ PHB in an RPR network, the relative packet drop priorities have to be implemented in the *client*.

As a side-node, we will mention that the most interesting lessons learned are related to the last scenario, were communications pairs were established in a randomized fashion. In this scenario, our simulator model was extensively tested, and many subtleties related to RPR conservative mode rate control were revealed. Indeed, we believe that this type of test scenario represents a scenario that may be used as a type of sanity check for the RPR conservative mode.

11 Further Work

A natural extension of this paper, would be to use the same simulation scenarios for the testing of RPR aggressive fairness mode.

As discussed above, the relative drop priorities within an Assured Forwarding PHB class is not supported by the RPR MAC, and thus has to be implemented in the *client*. A study on the implementation of this mechanism, to allow for full conformance to the AF PHB requirements would be interesting. Also, a study of admission control methods applicable for RPR networks used in a DiffServ environment seems reasonable.

12 Acknowledgements

We would like to thank Sven-Arne Reinemo, Tor Skeie and Olav Lysne for providing helpful insights into the DiffServ problem area.

References

1. Davie, B.: Deployment experience with differentiated services. In: Proceedings of the ACM SIGCOMM workshop on Revisiting IP QoS, ACM Press (2003) 131–136
2. Burgstahler, L., Dolzer, K., Hauser, C., Jahnert, J., Junghans, S., Macian, C., Payer, W.: Beyond technology: the missing pieces for QoS success. In: Proceedings of the ACM SIGCOMM workshop on Revisiting IP QoS, ACM Press (2003) 121–130
3. Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., Weiss, W.: An architecture for differentiated services (1998) IETF, RFC 2475.
4. El-Gendy, M.A., Bose, A., Shin, K.G.: Evolution of the Internet QoS and support for soft real-time applications. Proceedings of the IEEE **91** (2003)
5. IEEE Computer Society: IEEE Std 802.17-2004 Resilient packet ring (RPR) access method and physical layer specifications (2004)
6. OPNET Technologies, Inc: (OPNET Modeler)

7. Reames, C.C., Liu, M.T.: A loop network for simultaneous transmission of variable-length messages. In: Proceedings of the 2nd Annual Symposium on Computer Architecture. Volume 3. (1974)
8. Hafner, E., Nendal, Z., Tschanz, M.: A digital loop communication system. IEEE Transactions on Communications **22** (1974) 877–881
9. Davik, F., Yilmaz, M., Gjessing, S., Uzun, N.: IEEE 802.17 Resilient Packet Ring tutorial. IEEE Communications Magazine **42** (2004) 112–118
10. Gambiroza, V., Yuan, P., Balzano, L., Liu, Y., Sheafor, S., Knightly, E.: Design, analysis, and implementation of DVSR: a fair high-performance protocol for packet rings. IEEE/ACM Transactions on Networking **12** (2004) 85–102
11. Huang, C., Peng, H., Yuan, F., Hawkins, J.: A steady state bound for Resilient Packet Rings. In: Global Telecommunications Conference, (GLOBECOM '03). Volume 7., IEEE (2003) 4054–4058
12. Davik, F., Kvalbein, A., Gjessing, S.: An analytical bound for convergence of the Resilient Packet Ring Aggressive mode fairness algorithm. In: Proceedings of the 40th annual IEEE International Conference on Communications, Seoul, Korea (2005)
13. Nichols, K., Blake, S., Baker, F., Black, D.: (Definition of the Differentiated Services field (DS field) in the IPv4 and IPv6 headers)
14. Davie, B., Charny, A., Bennett, J., Benson, K., Boudec, J.L., Courtney, W., Davari, S., Firoiua, V., Stiliadis, D.: An Expedited Forwarding PHB (Per-Hop Behavior) (2002) IETF, RFC 3246.
15. Heinanen, J., Baker, F., Weiss, W., Wroclawski, J.: Assured Forwarding PHB group (1999) IETF, RFC 2597.
16. Tamir, Y., Frazier, G.L.: High-performance multi-queue buffers for VLSI communications switches. In: Proceedings of the 15th Annual International Symposium on Computer architecture, IEEE Computer Society Press (1988) 343–354
17. Karol, M.J., Hluchyj, M.G., Morgan, S.P.: Input vs. output queueing on a space-division packet switch. IEEE Transactions on Communications **35** (1987) 1347–1356
18. Mondragón, R.J., Arrowsmith, D.K., Pitts, J.M.: Chaotic maps for traffic modelling and queueing performance analysis. Performance Evaluation **43** (2001) 223–240
19. Horn, G., Kvalbein, A., Blomskøld, J., Nilsen, E.: An empirical comparison of generators for self similar simulated traffic. Submitted to Performance Evaluation (2004)
20. Jain, R., Routhier, S.: Packet trains–measurements and a new model for computer network traffic. IEEE Journal on Selected Areas in Communications **4** (1986) 986–995
21. Wang, X., Huang, B., Yu, X., Zhang, F.: Edge QoS study of RPR equipment. Proceedings of the SPIE The International Society for Optical Engineering **5281** (2004) 396–403

A Hot-Receiver Configuration

Parameter Name	**Value**
Fairness Mode	Conservative
Line Rate	1 [Gbit/s]
Packet size	500 [B] (fixed)
Packet Generator Settings:	Relative load: Offered load $\in [1, .., 30] \cdot (Baseload)$ Nodes 1-26 sending behavior: Poisson Node 15 Base load (average rate), EF: 20[Mbit/s] Nodes 1-26 Base load (average rate), AF: 12[Mbit/s] Nodes 1-26 Base load(average rate), BE: 20[Mbit/s] Node 0 sending behavior: self-similar Hurst parameter: 0.85 Base load, BE: $E(\overline{Rate}) = 20$[Mbit/s]
Rate Constraints (shaper settings)	Per Station AF Rate (nodes 1-26) : 12[Mbit/s] Node 15 EF Rate (subclass A0): 50[Mbit/s] Node 15 EF Rate (subclass A1): 50[Mbit/s]
STQ Thresholds	low: $125 \cdot 10^3$[bytes] medium: $185 \cdot 10^3$ [bytes] high: $250 \cdot 10^3$ [bytes]
rampUpCoef	64
rampDnCoef	64
ageCoef	4
lpCoef	32
link-delay	250 [μs]
Start of traffic	1.1s
Simulation Duration (simulated time)	2s
Number of independent simulations	12

Table 1: Detailed configuration information for "Hot-Receiver" simulation experiment.

B Backbone/Metro Ring Configuration

Parameter Name	**Value**
Fairness Mode	Conservative
Line Rate	1 [Gbit/s]
Packet size	500 [B] (fixed)
Packet Generator Settings for nodes 0-24:	Note that we have a 10/20/70 ratio of offered traffic for the EF/AF/BE PHBs
Relative load:	Offered load $\in [1, .., 60] \cdot (Baseload)$
EF Traffic	sending behavior: Poisson Base load (average rate) = 2.86[Mbit/s]
AF Traffic	sending behavior: self-similar Hurst parameter: 0.85 Baseload ($E(\overline{Rate})$) = 5.71[Mbit/s]
BE Traffic	sending behavior: self-similar Hurst parameter: 0.85 Baseload ($E(\overline{Rate})$ = 8[Mbit/s] $min = 1, max = 30$ (limited upwards so that traffic is not sent beyond node 30)
Rate Constraints (shaper settings for nodes 0-24)	 Per Station AF Rate: 8[Mbit/s] Per Station EF Rate (subclass A0): 2[Mbit/s] Per Station EF Rate (subclass A1): 2[Mbit/s]
STQ Thresholds	low: $125 \cdot 10^3$[bytes] medium: $185 \cdot 10^3$ [bytes] high: $250 \cdot 10^3$ [bytes]
rampUpCoef	64
rampDnCoef	64
ageCoef	4
lpCoef	32
link-delay	100 [μs]
Start of traffic	1.1s
Simulation Duration (simulated time)	2s
Number of independent simulations	12

Table 2: Detailed configuration information for "Backbone/Metro Ring" scenario.

C Random Fixed Pairs Configuration

Parameter Name	**Value**
Fairness Mode	Conservative
Line Rate	1 [Gbit/s]
Packet size	500 [B] (fixed)
Packet Generator Settings for all nodes:	Note that we have a 55/27/18 ratio of offered traffic for the BE/AF/EF PHBs
Relative load:	Offered load $\in [1, .., 10] \cdot (Baseload)$
AF Traffic	sending behavior: Poisson $Baseload$ (average rate) = 30[Mbit/s]
BE Traffic	sending behavior: Poisson $Baseload$ (average rate) = 60[Mbit/s]
EF Traffic	sending behavior: Poisson $Baseload$ (average rate) = 20[Mbit/s]
Rate Constraints (shaper settings for all nodes)	 Per Station AF Rate: 40[Mbit/s] Per Station EF Rate (subclass A0): 20[Mbit/s] Per Station EF Rate (subclass A1): 20[Mbit/s]
STQ Thresholds	low: $100 \cdot 10^3$[bytes] medium: $400 \cdot 10^3$ [bytes] high: $700 \cdot 10^3$ [bytes]
rampUpCoef	64
rampDnCoef	64
ageCoef	4
lpCoef	32
link-delay	250 [μs]
Start of traffic	1.1s
Simulation Duration (simulated time)	2s
Number of independent simulations	12

Table 3: Detailed configuration information for "Random Fixed Pairs" simulation experiment.