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].

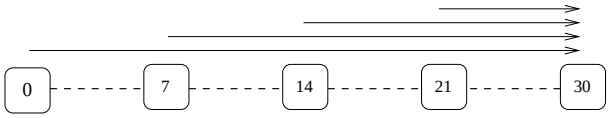


Figure 1: Nodes 0, 7, 14 and 21 all send traffic to node 30

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

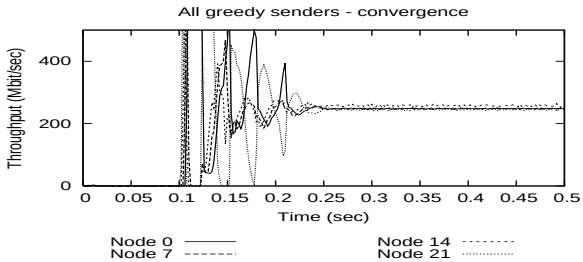
Fig. 1 illustrates our first scenario. For simplicity, there are 4 senders only, namely nodes 0, 7, 14 and 21 sending to node 30. Node 0 becomes the tail, and node 21 the head of the congestion domain.

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.

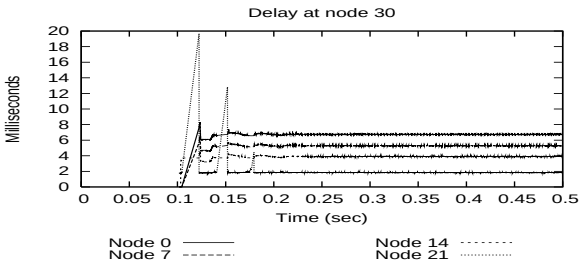
This can be observed in Fig. 2a in terms of reduced throughput for the head in the period [0.11,0.12] and in Fig. 2b by the peak in delay for traffic from node 21, at time 0.125. The effect of the increased transit delay (due to a higher *STQ* occupancy) can be observed for the other traffic flows as 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.

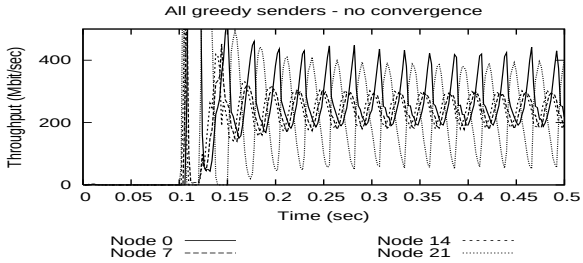
When the distance between the head and the tail is long, and the low-pass filter is not smoothing the add-rate measurements enough, the RPR fairness algorithm does not converge [10]. We run the exact same scenario as above once more, but this time we let the fairness algorithm use a slightly less smoothing (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 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.



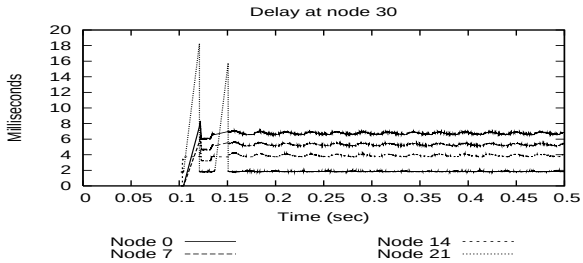
(a) Throughput using a slow low-pass filter (lpCoef=256)



(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.

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.

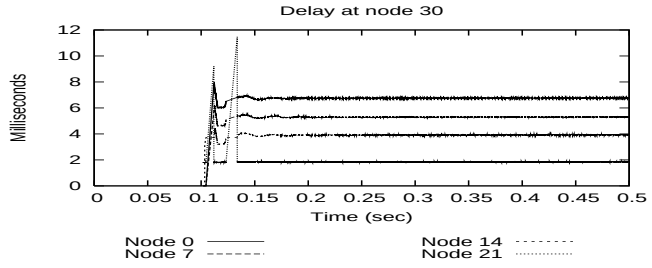


Figure 3: Latency per sender node measured at node 30 using the modified fairness algorithm

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.

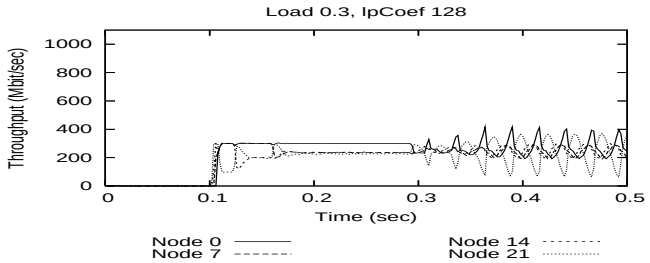
6. NON-GREEDY SENDERS

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).

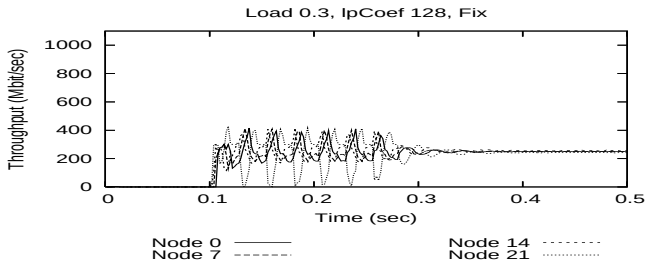
If we let each of the four active senders transmit at 30% of the full bandwidth, resulting in an aggregate demand of 120% of the full bandwidth, the system becomes congested. Some results are shown in Figs. 4a (throughput) and 4c (latency).

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.

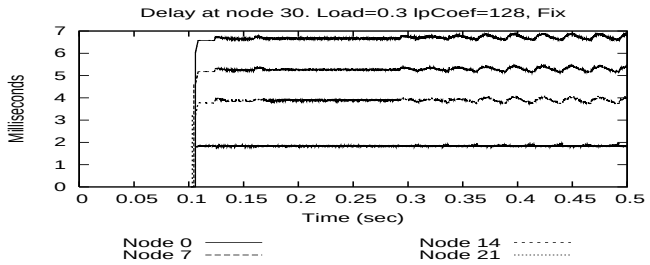
The use of our modified algorithm alleviates the problem of the tail sending too much. During the initial (transient) phase the throughput (Fig. 4b) and latency (Fig. 4d) of the individual flows varies. However, at time 0.3, when the fairness algorithm has converged,



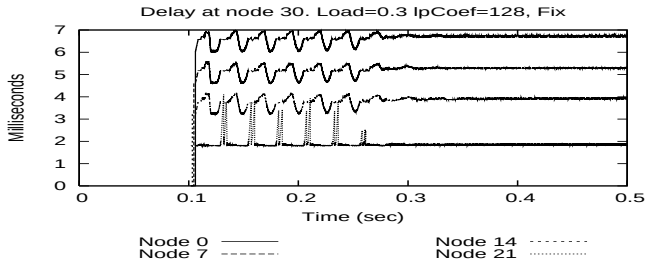
(a) Throughput Aggressive Fairness



(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.

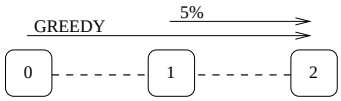


Figure 5: Unbalanced Scenario. The upstream node has an infinite demand (greedy sending behavior), while the downstream node has a modest sending behavior (sends at 5% of the line rate).

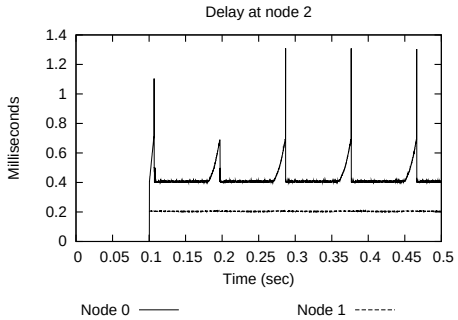


Figure 6: Modest congestion head node. The figure plots latency per sender node measured at node 2 using a quick low-pass filter (lpCoef=64)

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.