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.

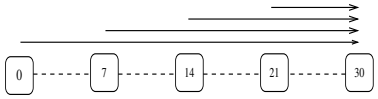


Fig. 1: Nodes 0, 7, 14 and 21 all send traffic to node 30.

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].

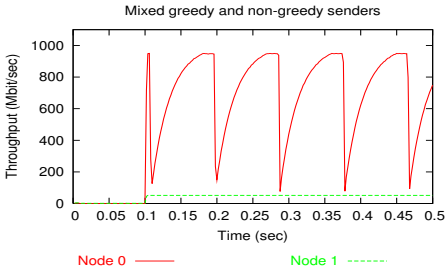


Fig. 2: Modest congestion head node. The figure plots throughput per sender node measured at node 2 using a quick low-pass filter (lpCoef=64) and 2 ms sampling intervals

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.

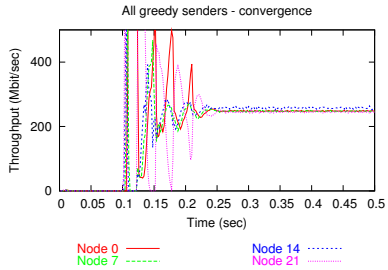
11 Further Work

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.

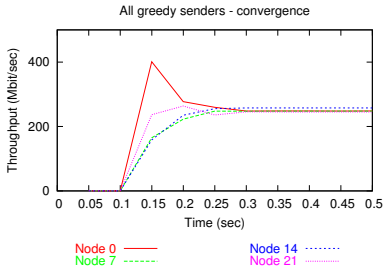
References

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)

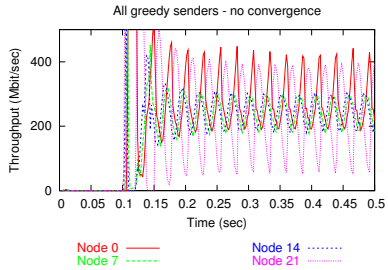
16. Knightly, E., Balzano, L., Gambiroza, V., Liu, Y., Yuan, P., Sheafor, S., Zhang, H.: Achieving High Performance with Darwin's Fairness Algorithm. http://grouper.ieee.org/groups/802/17/documents/presentations/mar2002 (2002) Presentation at IEEE 802.17 Meeting.
17. Yue, P., Liu, Z., Liu, J.: High performance 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



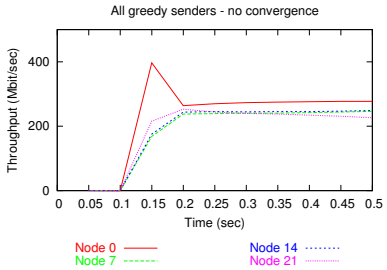
(a) lpCoef=256, 2 ms sampling interval



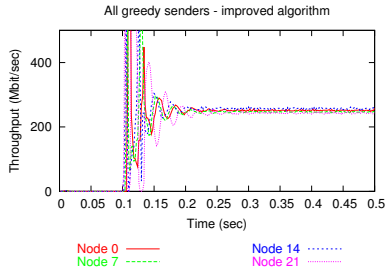
(b) lpCoef=256, 50 ms sampling interval



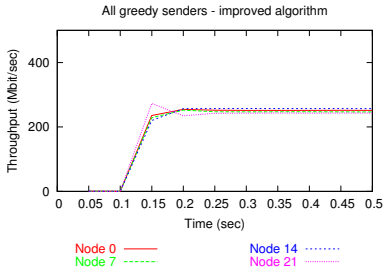
(c) lpCoef=128, 2 ms sampling interval



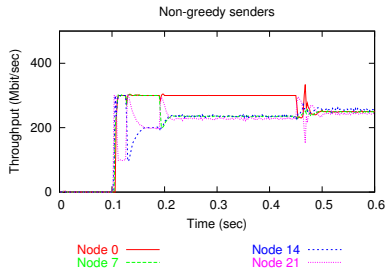
(d) lpCoef=128, 50 ms sampling interval



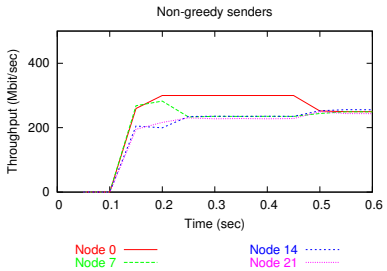
(e) lpCoef=128, 2 ms sampling interval



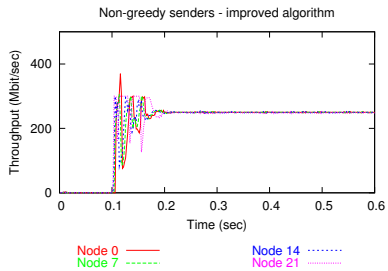
(f) lpCoef=128, 50 ms sampling interval



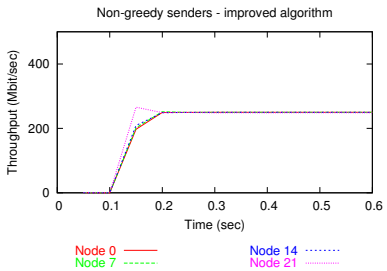
(g) lpCoef=128, 2 ms sampling interval



(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.