A Fairness Algorithm for High-speed Networks based on a Resilient Packet Ring Architecture

Stein Gjessing and Arne Maus

Simula Research Laboratory and Department of Informatics

University of Oslo, POB 1080 Blindern, Oslo, Norway

Email: steing@simula.no, arnem@ifi.uio.no

Abstract. **IEEE is currently standardizing a spatial reuse ring topology network called the Resilient Packet Ring (RPR, IEEE P802.17). The goal of the RPR development is to make a LAN/MAN standard, but also WANs are discussed. A fairness algorithm will regulate each stations access to the ring. The RPR fairness algorithm is being developed with mostly long distances between stations in mind. In this paper we discuss the feedback aspects of this algorithm and how it needs to be changed to give good performance if and when RPR is used for high-speed networks and LANs with shorter distances between stations. We suggest the use of triggers instead of timers to meet the response requirements of high-speed networks. We have developed a discrete event simulator in the programming language Java. The proposed improvements are evaluated using a ring network model that we have built using our simulator.**

Keywords. **High Speed Communication, Communication Networks and Protocols, Discrete Event Simulation, Resilient Packet Rings, Flow control, Fairness algorithm.**

I. INTRODUCTION

The IEEE Resilient Packet Ring (RPR) standardization group works under the LAN/MAN umbrella, and is designated IEEE P802.17. Its goal is to define a standard that can be used for high-speed LANs and MANs, but many companies also sees it as a WAN technology. Another goal is to be able to use physical layers of different kinds, e.g. high-speed point-to-point Ethernet.

RPR will define a full duplex ring topology. The nodes on the ring are called stations. A subnet that connects all the stations and moves traffic in one direction around the ring is called a ringlet. RPR will spatially reuse the ring bandwidth by letting the destinations strip the packets. Hence one packet may flow on one segment of a ringlet while another packet flows on another part of the same ringlet at the same time. Figure 1 shows a ring with 16 stations where spatial reuse is illustrated on the inner ringlet. Notice that each station is connected to two ringlets, and has a full duplex connection to the outside. Figure 2 shows one ringlet interface with *ingress*, e*gress* and p*assthru* buffers. Packets on their way into the ring are stored in the ingress buffer, while packets that are stripped from the ring are stored in the egress buffer. Packets that are traveling by a station on the ring, while this station sends out a packet from its ingress buffer, will have to wait in the stations passthru buffer.

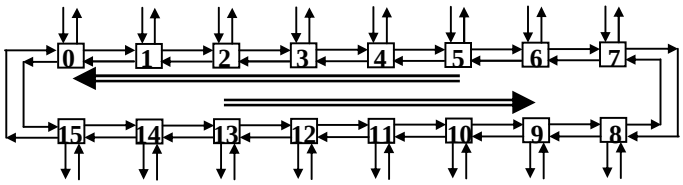


Figure 1. *A ring built from two ringlets with spatial reuse shown on the inner ringlet where station 5 sends to station 0 at the same time as station 13 sends to station 9.*

Ring networks have been designed and built for a long time, and are also extensively studied in the literature[18] [10] [21]. The first ring networks used a token to regulate the access to the communication medium. Later destination stripping with spatial reuse was exploited in systems like SRP [23], MetaRing[6], DQDB [11], ATMR [13], CRMA-II [15] and SCI [12]. Access to the ring must be controlled by a fairness or media access control (MAC) algorithm, and several have been proposed and evaluated [2,5,9,16,17,20].

The rest of this paper is organized as follows: In the next section we discuss the RPR fairness algorithm and some factors that could make RPR more suitable for high-speed networks and LANs. We also outline a new version of our fairness algorithm that we believe is better for high-speed networks. Then in section 3 we describe our experimental platform (the RPR discrete event simulator written in Java), and in section 4 we discuss and evaluate our proposals for improvements using this platform. In section 5 we conclude and point out further work.

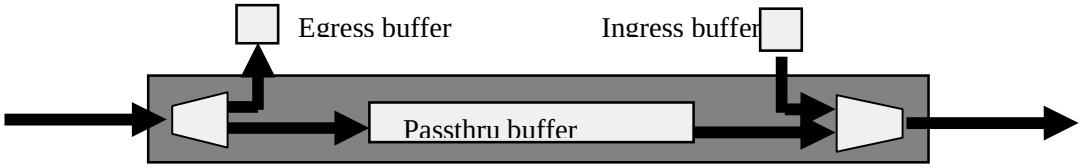


Figure 2. *Single ringlet network interface*

II. PROBLEM STATEMENT

A naïve algorithm for access to the ring is that a station is allowed to send whenever the passthru buffer is empty, or at least almost empty. However, a station that is sending a lot can effectively starve a downstream station if this simple access principle is used. A control system with feedback to upstream stations is used in order to avoid starvation and achieve fair access to the ring. The basis for such a control mechanism is flow control packets that a station sends upstream when it does not get its fair share of the bandwidth (it is about to be starved). Such a packet contains the starving stations current send rate. When the upstream stations receive flow control packets they adjust their sending rates to the value advertised in the flow control packets. When a station does not receive any more flow packets, it gradually increases its send rate again.

In this way all stations get to send the same amount after a while. In a WAN the control loop is long because of the long links between stations. Then it will take some time for the stations to converge to the same send value. In a high-speed network convergence should be much faster. In the sequel we discuss this convergence and the latency of the packets involved. In the SRP algorithm it is suggested to run the starving stations own send rate thru a low pass filter before the value is sent upstream. In the sequel we discuss the values used in this low pass filter.

Obviously a high-speed network has tighter latency constraints than a MAN or a WAN that has long distances between stations. In the SRP, the stations check their states about every 100 microseconds in order to see if a flow control notification has arrived or should be sent. In this paper we propose that in a high-speed network it is important to discover these two situations as early as possible. We propose the use of triggers so that the correct actions can be taken immediately and we discuss and evaluate how this improves the behavior of the network.

III. THE EXPERIMENTAL PLATFORM

We have developed a discrete event simulator in the programming language Java. This simulator includes an event queue that is served based on the service time of the queued events. Eg. when a packet is sent on a link, the arrival time of the packet on the receiving end of the link will be the time at which the packet should be taken out of the event queue and serviced.

We have also developed a model of RPR in Java . The model is built on top of our Java simulation kernel. Hence we can run the model and see how it behaves under different architectural parameters and different traffic patterns. The model is flexible; we can change it or extend it using the Java language.

In MANs and in WANs the distance between stations are measured in kilometers. In LANs and high-speed networks the distance is measured in meters. The difference in propagation delay and the expected packet latency is a key difference between these kinds of networks. This report focuses on high-speed networks and we will hence use a relatively short distance, mostly about 25 meters, between the stations.

The link speed used is one Gbyte/sec. We use data packets that are low priority and 500 bytes long, while the flow control packets are 32 bytes high priority packets. In some of the experiments we vary the passthru buffer size, but the size is mostly 5000 bytes, with a threshold (starvation) value of 1500 bytes.

IV. THE EXPERIMENT

We illustrate and discuss our suggested improvements, including the proposed triggers in the fairness algorithm, with an experiment, pictured in figure 4. Station 0 sends at full speed (2 packets per microsecond) to station 6 for the duration of the experiment (22 milliseconds). Station 2 starts a full speed flow to station 4 at time 2 milliseconds, and terminates this flow after 16 milliseconds, i.e. at time 18 milliseconds.

When station 2 starts to send (after 2 milliseconds), both the passthru buffer and the ingress buffer starts to fill up. When the size of the passthru buffer reaches the threshold, the station is congested and sends a flow control notification packet upstream (upstream links are not showing in figure 4). In the SRP algorithm, the size of the passthru buffer is checked at regular intervals. In a MAN/WAN network, this can typically be done every 100 microseconds [23]. The plot marked “Timer 100 microsec” in figure 5 shows that in a high-speed network, this is not a good idea. We first explain the nature of figure 5.

Figure 5 shows what happens in the first 125 microseconds after station 2 has started to send. We have run the experiment with four different versions of the fairness algorithm; hence there are four plots in this figure. The latency that packets experience *from the moment they are ready to be put onto the ring* by station 2, until they reach station 4, are plotted on the y-axis

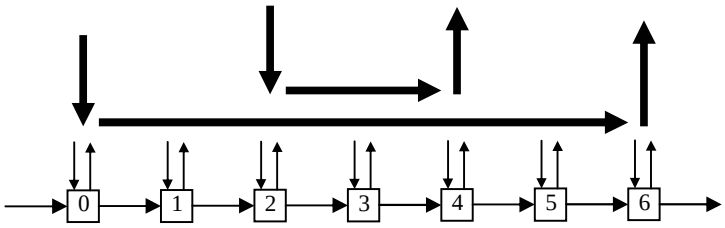
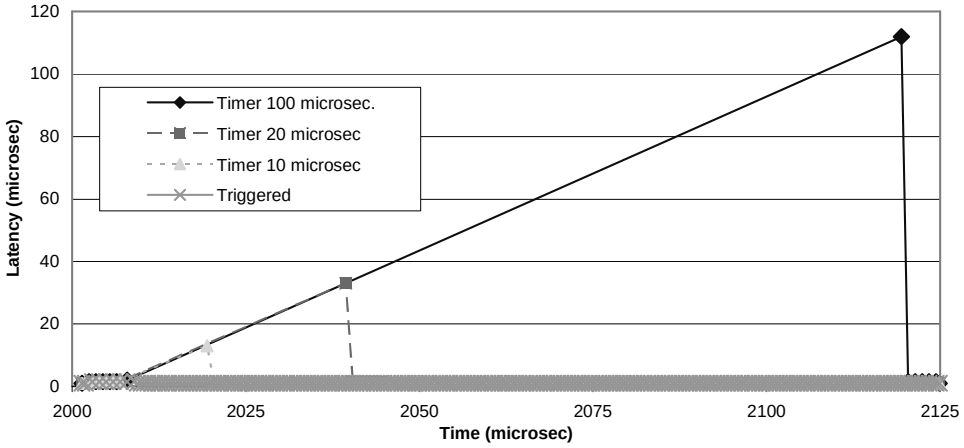


Figure 4. *A ringlet segment with two competing flows.*

. Only packets from station 2 to station 4 are shown. Each point in the plot represents one packet. The time the packet is accepted by station 4 is shown on the x-axis. Only those packets arriving during the first 125 microseconds after time 2 ms. are shown on the figure. When the passthru buffer in station 2 has reached the threshold value (1500 bytes), only packets from the passthru buffer are forwarded downstream. For each run the top latency value is the latency of the packet sitting at the head of the ingress buffer in station 2 when the passthru buffer reaches the threshold, and station 2 is not allowed to send any more. This packet has to wait for the flow control notification signal to reach station 0, and also wait for station 0 to send less traffic. Then the passthru buffer will shrink below the threshold, and the packet that has waited can finally be transmitted



.

Figure 5. *The latency of the first packets in the new flow from station 2 to station 4. The new flow starts at time 2000 us, and the figure shows the latency of the packets arriving during the first 125 us thereafter.*

The slower the system reacts to the starvation, the longer the packet at the head of the ingress queue in station 2 has to wait. We have tried with 100 microsecond timers, 20 microsecond timers and 10 microsecond timers. In all these cases we see that the latency might get unacceptably high (between 14 and 115 microseconds).

The latency values shown for the timers in figure 5 might be acceptable for a MAN/WAN, but not for a high speed network. In order to discover the possible upcoming starvation as soon as possible, we suggest the use of triggers. We have implemented the fairness algorithm so that whenever there is a line of packets in the bypass fifo, and a new packet arrives, we test for a possible starvation. If the passthru buffer is filled above a certain value, the station immediately sends out a flow control packet.

When a flow control notification arrives at an upstream station, the send value in the packet is remembered, but in the timer scheme it is not reacted upon until the timer goes off. Hence it may take a while until the station starts sending less traffic. Again this might be acceptable in a MAN or a WAN, but not when packet latency should be minimized. Our trigger scheme will act upon the received flow control notification immediately, and adjust the stations send rate to the received value at once. The latency of the packets from the run with the trigger scheme is almost not visible in figure 5. The trigger scheme latency is, however, also shown in figure 6 (250ns links – 1500B threshold).

Figure 6 shows three results using the trigger scheme. Only packets traveling from station 2 to station 4 during the first 20 microseconds after the flow has started, are plotted. The axes mean the same as in the previous figure. Notice however that the scales are different. The bottom curve shows the same trigger result as in figure 5. The figure also illustrates what happens if the links between the stations increase in length so that the one way link latency goes up to 400 and 500 ns. It not only takes longer for the packets to travel from station 2 to station 4, but it also takes longer for the flow notification packet from station 2 to reach the source (station 0).

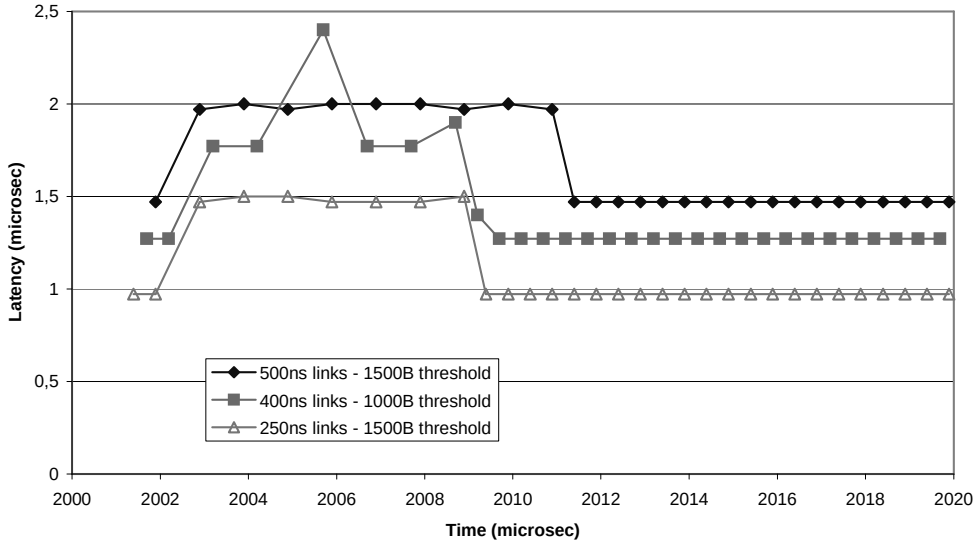


Figure 6. *The latency of the first packets in the new flow from station 2 to station 4. The trigger scheme is used for all three runs. The new flow starts at time 2000 us, and the figure shows the latency of the packets arriving during the first 20 us thereafter.*

In the bottom and the top plots the flow control notification is sent immediately when the passthru buffer size reaches 1500 bytes. Being emptied at half speed, six more 500 byte packets can then arrive before the station is congested. During this time the flow control packet has had enough time to reach station 0, order it to stop transmitting and for the stop effect to be noticed at station 2, i.e. a full round trip time between stations 2 and 0.

In the middle plot, however, we have decreased the threshold value to 1000 bytes, and then the passthru buffer more than fills up to the threshold while the flow notification packet takes effect. In order for the plots not to come on top of each other, we ran that experiment with a link latency of 400ns. In this middle plot we see that the passthru buffer threshold is so small that before the flow control notification has had any effect, station 2 must serve the passthru buffer only, giving the first packet in the ingress buffer a really long latency (about 2.4 microseconds).

We now turn to a more holistic view of the experiment. In figures 7 and 8 we see the stability and the adaptation to changing traffic load of the timer and the trigger scheme. Results (the y axis) are the number of packets received at the destination every 100 ms, shown both for the flow from station 0 to station 6 and for the flow from to station 2 to station 4. The complete experiment takes 26 ms (x axis).

Figure 7 shows the algorithm with a 100 microseconds timer and the original low pass filter. Figure 8 shows the trigger scheme and a filter that accepts 16 times faster oscillations. Because station 0 has been sending for some time when station 2 starts sending, the fairness algorithm at first gives priority to station 2. In figure 7, station 2 gets to send alone for the longest time. Figure 7 also shows that the run with the timer scheme has fluctuations as long as both flows are active. This scheme is designed for MANs and WANs and will not stabilize so quickly. However we see that the "waves" are getting shorter with time, indication that would smooth out if the two flows had kept on competing.
Figure 8 show great instabilities the first two milliseconds after the point where the two flows start competing. Then the run flattens out with almost an exactly 50-50 division of the bandwidth for a while. Then there are some temporary instability, before the bandwidth is divided equally again.

When the flow from station 2 to station 4 terminates, the speed at which station 0 is increasing its send rate is quite different in the two figures. Here it is easy to see the effect of the low pass filter. The lowest filter explains the very slow rise of the sending rate after time 18 milliseconds in figure 7. Because of the higher frequency filter, in figure 8 it seems like station 0 starts to send at full speed almost immediately after time 18 milliseconds.

In order to investigate long-term stability, we changed the experiment and let the two flows continue to compete also after 18 milliseconds. With the timer and the original filter, the flow from 0 to 6 had taken a total of 47 % of the bandwidth after 100 ms., 49 % after 150 ms. and 50% after 200 ms. We ran this experiment with the trigger version of the algorithm and the higher frequency filter. Then the flow from 0 to 6 had taken 48% of the bandwidth after 30 ms, 49% of the bandwidth after 50 ms, and 50 % of the bandwidth after 100ms. Hence, both schemes converge; but the trigger scheme with aggressive aging values does so much quicker.

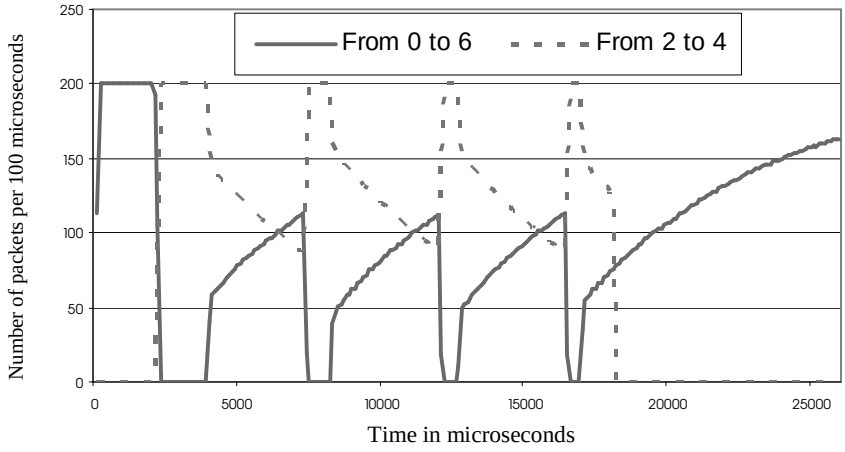


Figure 7. *Packets received during the duration of the experiment with two competing flows – Timer scheme with original aging values*

V. CONCLUSIONS AND FURTHER WORK

We have analyzed and discussed aspects of the flow control or fairness algorithm of the Resilient Packet Ring architecture. Included in this work is an implementation of an RPR model in our discrete event simulator .

We have shown how a revised version of the RPR fairness algorithm can be tuned so that it is suitable for a high speed network. In particular our implementation of triggers to handle fairness seems to be very promising. Also the use of a higher frequence filter seems to be necessary in order to handle quick flow changes in high-speed networks with short links. The triggers and the new filter introduce some instability. The duration of these instabilities are short, and have no significance in the long run. They however deserve further study, because they could result in unexpected long communication delays between processes on different parts of the network.

We have investigated and explained the relation between passthru buffer threshold values and station-to-station latency. In particular we have seen that in LANs and high-speed networks with short distances between stations, passthru buffers that hold eight to twelve packets are large enough.

The total latency that a packet experience is the propagation delay on the links and the combined latency of waiting in the ingress buffer and all the passthru buffers under way. We would like to understand the trade off between the sizes of these buffers. We will also look at the smoothing effect of passthru buffers.

ACKNOWLEDGEMENT

The authors thanks Olav Lysne for making the initial simulation kernel and for help in building the discrete event simulator.

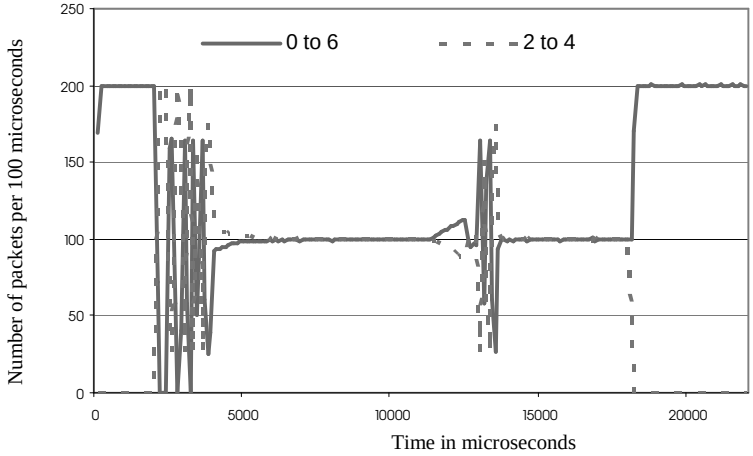


Figure 8. *Packets received during the duration of the experiment with two competing flows – Trigger scheme with one sixteenth of the original aging values*

REFERENCES

1. ANSI T1.105.01-2000: Synchronous Optical Network (SONET) - Automatic Protection.
2. H.R. van As: Major Performance Characteristics of the DQDB MAC Protocol. Telecommunications Symposium, 1990. ITS'90 Symposium Record, SBT/IEEE 1990
3. S. Breuer, T.Meuser: Enhanced Throughput in Slotted Rings Employing Spatial Slot Reuse. INFOCOM '94. Networking for Global Communications. IEEE. 1994
4. I. Cidon, L. Georgiadis, R. Guerin, Y. Shavitt: Improved fairness algorithms for rings with spatial reuse. INFOCOM '94. Networking for Global Communications. IEEE, 1994
5. I. Cidon, Y. Ofek: Distributed Fairness Algorithms for Local Area Networks with Concurrent Transmissions. In: Lecture Notes in Comp. Sci., Vol. 392, Springer, 1988
6. I. Cidon, Y.Ofek: MetaRing - A Full-Duplex Ring with Fairness and Spatial Reuse. IEEE Trans on Communications, Vol. 41, No. 1, January 1993.
7. K.C. Claffy: Internet measurements: State of DeUnion. http://www.caida.org/outreach/presentations/Soa9911
8. M.W. Garrett, S.-Q. Li: A study of slot reuse in dual bus multiple access networks**.** IEEE Journal on Selected Areas in Communications, Vol. 9 Issue 2, Feb. 1991
9. A. Grebe, C. Bach: Performance comparison of ATMR and CRMA-II in Gbit/s-LANs. SUPERCOMM/ICC '94, IEEE Int. Conf. on Serving Humanity Through Communications, 1994
10. IEEE Standard 802.5–1989, IEEE standard for token ring
11. IEEE Standard 802.6–1990, IEEE standard for distributed queue dual bus (DQDB) subnetwork
12. IEEE Standard 1596–1990, IEEE standard for a Scalable Coherent Interface (SCI)
13. ISO/IECJTC1SC6 N7873: Specification of the ATMR Protocol (V. 2.0), January 1993
14. I. Kessler, A. Krishna: On the cost of fairness in ring networks. IEEE/ACM Trans. on Networking, Vol. 1 No. 3, June 1993
15. 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.
16. M.J. Marsan et al.: Slot Reuse in MAC Protocols for MANs. IEEE J. on Selected Areas in Communications. Vol. 11, No. 8, October 1993.
17. H.R. Muller et al: DQMA and CRMA: New Access Schemes for Gbit/s LANs and MANs. INFOCOM '90, Ninth Annual Joint Conference of the IEEE Computer and Communication Societies. IEEE , 1990
18. R.M. Needham, A.J. Herbert: The Cambridge Distributed Computing System. Addison-Wesley, London, 1982.
19. T. Okada, H. Ohnishi, N. Morita: Traffic control in asynchronous transfer mode. IEEE Communications Magazine , Vol. 29 Issue 9, Sept. 1991
20. D. Picker, R.D. Fellman: Enhancing SCI's fairness protocol for increased throughput. IEEE Int. Conf. On Network Protocols. October, 1993.
21. F.E. Ross: Overview of FDDI: The Fiber Distributed Data Interface. IEEE J. on Selected Areas in Communications, Vol. 7, No. 7, September 1989.
22. I. Rubin, H.-T. Wu: Performance Analysis and Design of CQBT Algorithm for a Ring Network with Spatial Reuse. IEEE/ACM Trans on Networking, Vol. 4, No. 4, Aug. 1996.
23. D. Tsiang, G. Suwala: The Cisco SRP MAC Layer Protocol. IETF Networking Group, RFC 2892, Aug. 2000
24. S. Gjessing and B.F. Davik: Avoiding Head of Line Blocking using an Enhanced Fairness Algorithm in a Resilient Packet Ring. Proceedings 2002 International Conference on Telecommunications (ICT 2002).
25. Gjessing, S.; Maus, A.; Strøm, T.; Huse, L.P.: Running the Synthetic Aperture Radar (SAR) Application on a switched SCI cluster -- Gustavson, D.B.; Li, Q. (ed.): Proceedings The SCIzzL-12 Low-Cost High-Performance Computing Workshop Santa Clara University , CA. 1999-08-21.
26. Strøm, T.; Maus, A.; Halfen, B.; Gjessing, S.: A HIC Based SCI switch - implementation and performance -- Gustavson, D.; Li, Q. (ed.): SCIzzL-10/11 Workshop Proceedings The SCIzzL-11 Low-Cost High-Performance Computing Workshop Santa Clara University , CA. 1999-03-23.