The Stability of the Resilient Packet Ring Aggressive Fairness Algorithm

Fredrik Davik[1,2] *and Stein Gjessing*[1]
[1]*Simula Research Laboratory*
[2]*Ericsson Research Norway*
{bjornfd,steing}@simula.no

19th March 2004

Abstract

Resilient Packet Ring (RPR) is currently being standardized by the Institute of Electrical and Electronics Engineers (IEEE), within the 802.17 working group. RPR is a dual ring based medium access protocol based on the "buffer insertion" principle. Instead of controlling the access to the ring using a circulating token, each station on the ring executes its part of a distributed fairness algorithm. The task of the fairness algorithm is to ensure that all stations gets their fair share of the bandwidth, when there is more demand than supply. This article discusses how the RPR fairness algorithm is able to divide the bandwidth fairly among contending stations and also, what are the requirements for this to work correctly. We discuss how and when the fairness algorithm fails to converge to a fair value, and what is needed in order to make it converge. Finally we show that the setting of the RPR $lpCoef$ *configuration parameter is of crucial importance for the convergence of the fairness algorithm, and some guidelines for setting this correctly are provided. To do this, we first present some analytical work, of which some was contributed to the RPR working group and is now incorporated in the standard. We explain our approach using an informal "proof by induction".*

I Introduction

Resilient Packet Ring (RPR) [1, 2] is currently being standardized by the Institute of Electrical and Electronics Engineers (IEEE), within the LAN/MAN 802.17 working group. RPR is a dual ring based medium access protocol based on the "buffer insertion" principle [3, 4]. Instead of using a token to access the shared ring, any station can in principle send frames onto the ring as long as there are no frames passing the station (no frames in transit). If a station is sourcing a frame onto the ring, and another frame (a frame in transit) arrives to the station, then the transiting frame is stored in the *insertion buffer* as long as it takes the station to source its frame. In RPR, the insertion buffer is called the Transit Queue (see figure 1b). When, however, a sending station has to wait for the transit path to be empty before it can source a packet, an upstream station can easily starve the downstream ones. In general, when the bandwidth on a ring segment is less than the demand, this scheme will give priority to the station furthest away from the congestion point. The IEEE RPR working group decided that when the demand for bandwidth on a segment of the ring, is greater than the supply, the bandwidth shall be divided fairly among the stations sending frames over that part of the ring. Every station on the ring have an associated weight, so that a fair division of bandwidth is not necessarily an equal one. This principle, where the division of bandwidth applies to the individual stations aggregate of ingress traffic, is the same principle as is stated by the RIAS fairness principle [5]. In order to achieve this, a distributed fairness algorithm is specified. This paper discusses how the RPR fairness algorithm divides the bandwidth fairly among contending stations, and also, what are the requirements for the algorithm to work correctly. The aim of the fairness algorithm is to compute the bandwidth that each of the contending stations can use. According to the RPR specification, this can achieved in any of two different ways and the two options are called respectively the aggressive and the conservative version of the algorithm. The conservative version has a more damped performance, making changes to the fair bandwidth only very slowly. It also needs to send some extra controls messages in order to execute the algorithm. The aggressive version reacts must faster to change, and needs less control information to work. In this article we are going to concentrate on the aggressive version of the fairness algorithm. After a change in the traffic pattern, the fairness algorithm should, after a transient period, converge to the fair rate. In this paper we assume that the new traffic pattern is stable. However, analyses show that the fairness algorithm not always converges to the new fair rate. The main goal of this article is to illustrate how and when the aggressive fairness algorithm fails to converge in its computation of the fair bandwidth value, and explain what is needed in order to make the algorithm converge to a fair value. In chapter II we also show that the setting of a smoothing parameter, called the $lpCoef$ configuration parameter, is of crucial importance for the convergence of the fairness algorithm. We provide some guidelines for setting this parameter correctly. To do this, we first present some analytical work, of which some was contributed to the RPR working group, and is now incorporated in the standard. We explain our approach in chapter II using an informal "proof by induction". In chapter III we show show some additional simulation results to illustrate our findings. And finally, in chapter IV we conclude and give some directions for future work. Note that an underlying assumption for this presentation is that the size of the congestion domain's stations transit queue(s) and associated threshold levels allows storage of excess traffic during the fairness algorithm's convergence to the fair rate (i.e. in figure 3, $t_4 < t_5$). If this is not the case, the operation of the fairness algorithm is additionally constrained, and may never converge to the fair rate.

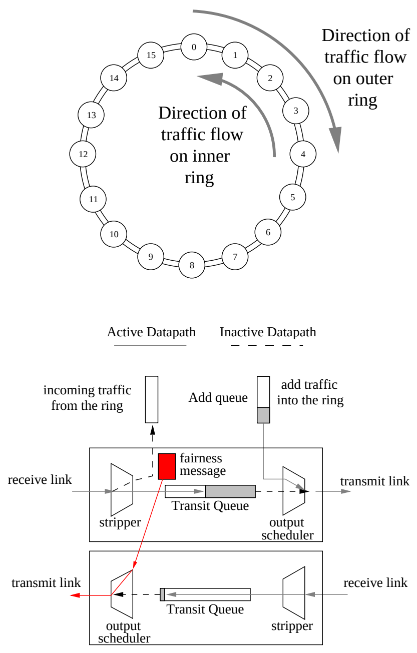


Figure 1: a) 16 node dual ring topology, b) A node's attachment to both rings and the sending of a fairness message upstream on the opposite ring.

II Informal Proof by Induction

The induction basis is trivial; one station sending at full speed over its downstream link. In order not to make the mathematics the focus of this informal article, we use an example in the induction step.

Assume that stations 1, 3 and 5 already are sending traffic to station 8 (on the outer ring), that they all try to add as much traffic as possible to station 8, and that the fairness algorithm have been able to divide the bandwidth fairly among them. Now station 7 also tries to send as much as possible.

When the RPR transit queue is lightly filled, the scheduler on the transmit link divides the downstream bandwidth equally between the add queue and the transit queue. Hence, when station 7 starts sending, its output scheduler will initially transmit at equal rates from the add queue (see figure 1b) and the transit queue. Hence, station 7 will initially add traffic at half of the link rate. This will result in the transit queue of station 7 starting to fill. When the fill level crosses a threshold called "Low" (the default value for Low is 1/8 of the queue size), the station is by definition congested.

When a station becomes congested, it calculates an approximation to a fair division of the bandwidth, called "the fair rate", by using its own current add rate. This approximate value is "advertised" (sent on the opposite (inner in our example) ring) to all upstream stations contributing to the congestion, and these stations have to adjust their add rate accordingly. This segment of the ring, that receives a fairness message with the same value originating from the congested station, is called a congestion domain. The congested station (station 7) is called the head of the congestion domain, while the upstream station furthest away, contributing to the congestion (station 1), is called the tail of the congestion domain.

In the aggressive mode, a congested station advertises its approximation to the fair rate every "aging interval", where the duration of an aging interval is 100μs (for line rates $\geq$ 622Mbit/s). When a station periodically advertises new approximations to the fair rate, not waiting for the effect of the previous one to be noticed, the calculations of the fairness algorithm may not converge to the fair rate, resulting in an unstable system. (In the conservative mode the rates are controlled more conservatively).

Let the amount of traffic added by the congestion domain head (station 7), measured as a byte-count the output of the output scheduler, during aging interval n be denoted Δ_n. Let two low-pass filtered series of Δ_n be denoted $addRate_n$ and $lpAddRate_n$. The subscript n indicates that these are distinct, discrete values, corresponding to the value at the expiration of aging interval n. The value of $lpAddRate_n$ is the fair rate approximation transmitted to upstream stations at the expiration of aging interval n.

$addRate_n$ are the values taken on by a counter called $addRate$ ([1], clause 9,6). It is computed by:

$$addRate_n = \frac{addRate_{n-1} \cdot (ageCoef - 1)}{ageCoef} + \Delta_n \quad (1)$$

, where $ageCoef \in [1, 2, 4, 8, 16]$

This crudely means that $addRate_n$ is a slightly smoothed value of Δ_n multiplied by $ageCoef$.
The RPR stations calculates an even more smoothed value of Δ_n by:

$$lpAddRate_n = \frac{1}{lpCoef} \cdot$$
$$(lpAddRate_{n-1} \cdot (lpCoef - 1) + addRate_n) \quad (2)$$

where $lpCoef \in [16, 32, 64, 128, 256, 512]$

Let:

$$k = \frac{ageCoef - 1}{ageCoef} \quad (3)$$

And:

$$k_2 = \frac{lpCoef - 1}{lpCoef} \quad (4)$$

If Δ_n is constant and its value equals Δ, $addRate_n$ can be rewritten in terms of a geometric series:

$$addRate_n = \Delta \cdot \frac{k^n - 1}{k - 1} \quad (5)$$

By substituting 4 into 2 and rearranging, we get 6.

$$lpAddRate_n = lpAddRate_{n-1} \cdot k_2 + \frac{addRate_n}{lpCoef} \quad (6)$$

Assuming Δ_n is constant and equals Δ, this can be rewritten as the difference between two geometric series as shown in 7.

$$lpAddRate_n = \frac{\Delta}{lpCoef \cdot (k-1)} \cdot \quad (7)$$
$$\left(k_2^{n-1} \cdot k \cdot \frac{\left(\frac{k}{k2}\right)^n - 1}{\frac{k}{k2} - 1} - \frac{k_2^{n-1} - 1}{k_2 - 1} \right), n > 2$$

Given a constant amount of traffic, added at rate Δ, the values for both $addRate_n$ and $lpAddRate_n$ will approach asymptotic values, given by:

$$\begin{aligned} \lim_{n \to \infty} & \quad addRate_n & (8) \\ = & \quad \lim_{n\to\infty} lpAddRate_n \\ = & \quad ageCoef \cdot \Delta \end{aligned}$$

Thus:

$$addRate_n \in [0, \ldots, ageCoef \cdot \Delta], \quad (9)$$
$$lpAddRate_n \in [0, \ldots, ageCoef \cdot \Delta]$$

As explained above, when station 7 starts to send, the add rate, Δ_n, will be constant at half of the link capacity. Only when the transit queue in station 7 is filled above a threshold called "High" (with a default value of 1/4 of the transit queue size), will the output scheduler refrain from selecting traffic from the add queue. While the add traffic is stopped, the values of Δ_n are 0. Stalling of add traffic when the transit queue level exceeds the High threshold is due to preservation of delay bound guarantees for high priority traffic. This issue is not discussed in this presentation.

Assuming Δ_n is constant at half of the link speed, and that $\Delta_n = \Delta$, figure 2 plots the value of equations 1 and 6 as well as the maximum value ($ageCoef \cdot \Delta$) and the line for 95% of the maximum value.

The dataset for $lpAddRate_n$ measured during simulation at the congestion head is also included in the same plot. As can be seen, the analytical values correspond well to the values obtained from simulation. In figure 2, as the number of aging intervals n exceeds 95, the simulated value for $lpAddRate_n$ starts to ramp down. This is the point where the add traffic is stopped and Δ_n becomes 0.

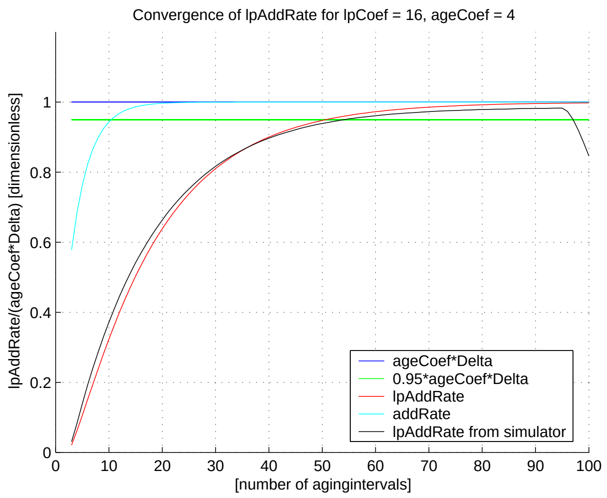


Figure 2: Convergence of addRate and lpAddRate, including lpAddRate obtained from simulation

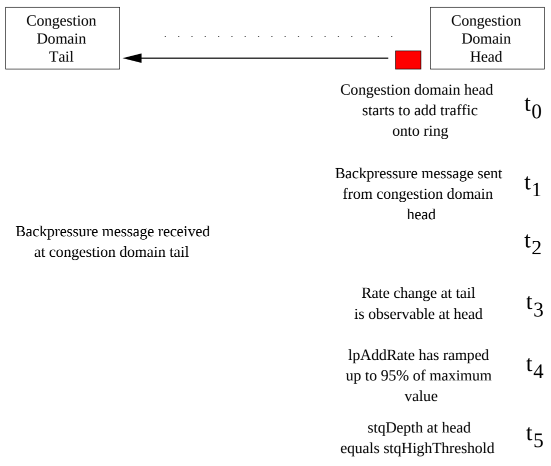
Let t_0 denote the time when the congested station (i.e. the head of the congestion domain, station 7) starts to add traffic onto the ring. The delay between the time (denoted t_1) this station advertises an adjustment, and the time (denoted t_3) the result is observable by the same station (in the form of a change in the rate of received transit traffic), is controlled mainly by the hop-count and physical distance between the head and the upstream senders. Let t_4 denote the time required for $lpAddRate_n$ to ramp up from 0 to 95% of its maximum value (when adding traffic at half of the link bandwidth) and let t_5 denote the time when the transit buffer occupancy exceeds the "High" threshold. When Δ_n is constant during the interval $[t_0, t_4]$, it can be shown that the rise-time $(t_4 - t_0)$ of the $lpAddRate_n$ function to 95% of the max value, is approximately proportional to the value of the $lpCoef$ parameter. Thus, if $lpCoef$ doubles, the rise-time of $lpAddRate_n$ to the 95% level, also doubles.

The course of events from the point when congestion is detected (t_1) until the fairness algorithm converges (at the fair division of rates), or does not converge, consist of a set of adjustment cycles. Each cycle is characterized by a monotonic increase of the advertised rate until the point, when the fill level of the transit queue exceeds the "High" threshold (t_5). Following this, Δ_n becomes 0 and hence the advertised rate is monotonically decreased, until the fill level of the transit queue falls below the same threshold. By this, the cycle is concluded. For a system that converges to the fair division of rates, the magnitude of the oscillations (the difference between the highest and lowest advertised rate during a cycle) will have an exponential decreasing envelope, while for an unstable system, the magnitude of the oscillations are almost stable. Figure 3 shows the relative ordering of the events occurring from t_0 to t_5.

From a study based on an analytical evaluation and simulations we conclude that in order to obtain a stable system, the propagation delay between the head and tail nodes in the congestion domain needs to be smaller than the size of the interval $[t_1, t_4]$.

For any given scenario, the values of t_1 *and* t_4 may be calculated analytically. To ensure correct setting of the $lpCoef$ parameter for a ring of arbitrary size, the propagation delay between any two nodes on the ring must not exceed the bound $t_4 - t_1$. Hence, for a given value of $lpCoef$, the circumference of the largest possible topology can be calculated.

For the scenario discussed in this presentation, $t_0 = 1.2s$ and $t_1 = 1.2046s$. The calculated values



Congestion Domain Tail
Congestion Domain Head
Congestion domain head starts to add traffic onto ring
t_0
Backpressure message sent from congestion domain head
t_1
Backpressure message received at congestion domain tail
t_2
Rate change at tail is observable at head
t_3
lpAddRate has ramped up to 95% of maximum value
t_4
stqDepth at head equals stqHighThreshold
t_5

Figure 3: Monotonic Increasing Part of Congestion Detection and rate adjustment cycle.

of t_4 for the possible values of $lpCoef$ are shown in equation 10. The equation also shows the largest propagation delay supported for a given setting of $lpCoef$.

$$\begin{aligned}
&t_4(lpCoef = 16)\\
&\quad \approx t_0 + 5ms\\
&\quad \Rightarrow t_{propagation} < 5 - 4.6 = 0.4\ [ms]\\
&t_4(lpCoef = 32)\\
&\quad \approx t_0 + 9.8ms\\
&\quad \Rightarrow t_{propagation} < 9.8 - 4.6 = 5.2\ [ms]\\
&t_4(lpCoef = 64)\\
&\quad \approx t_0 + 19ms\\
&\quad \Rightarrow t_{propagation} < 19 - 4.6 = 14.4\ [ms]\\
&t_4(lpCoef = 128)\\
&\quad \approx t_0 + 39ms\\
&\quad \Rightarrow t_{propagation} < 39 - 4.6 = 34.4\ [ms]\\
&t_4(lpCoef = 256)\\
&\quad \approx t_0 + 77ms\\
&\quad \Rightarrow t_{propagation} < 77 - 4.6 = 72.4\ [ms]\\
&t_4(lpCoef = 512)\\
&\quad \approx t_0 + 154ms\\
&\quad \Rightarrow t_{propagation} < 154 - 4.6 = 149.4\ [ms]
\end{aligned} \tag{10}$$

A possible improvement to the *Aggressive mode* fairness algorithm would be to leave the setting of the $lpCoef$ parameter to the RPR topology discovery mechanism. In their topology database, all stations on the ring have information about the size of the ring (in the form of propagation delays around the ring). Thus the setting of the $lpCoef$ parameter could be calculated during topology discovery upon ring initialization and again when nodes are added to or removed from the ring. This would be somewhat analogous to the operation of the RPR *Conservative mode* fairness algorithm, where the propagation delay (including transit queue delays) between the head and tail stations in a congestion domain is measured, and the resulting value decides how fast to perform rate adjustments. For the *Aggressive mode* fairness algorithm however, the $lpCoef$ parameter would be stable as long as the network topology is stable, while for the *Conservative mode* fairness algorithm, the interval between rate adjustments is recomputed periodically. A more computing intensive option for the *Aggressive mode* fairness algorithm could be to let a node, at the time it becomes the head of the congestion domain, recompute $lpCoef$ and again when the location of the assumed congestion domain tail changes. This could be done without the passing of control messages, using node local information only.

III Simulation Results

For the scenario discussed in this presentation, we have performed simulations using our OPNET Modeler [6] simulator model implementing RPR as described in [1]. In the simulated scenario, the link propagation delay is 250μs, resulting in a propagation delay from station 1 to station 7 and back (12 hops) of 3ms. Looking at equation 10 above, for $lpCoef = 16$, the maximum allowed propagation delay is 0.4ms, which is clearly below the actual propagation delay of the scenario which is 3ms. Thus the system should not converge to the fair rate. On the other hand, for $lpCoef \geq 32$, the allowed propagation delay is greater than the actual propagation delay of 3ms. Thus the system should converge to the fair rate. Two plots are included below. Figure 4 shows how the congested station fails to converge to the fair rate when $lpCoef$ is set to 16. Figure 5 shows how the congested station succeeds in its attempt to adjust to the fair rate

($\frac{840 \cdot 10^6}{4} bit/s$) when $lpCoef$ is set to 32. In figure 5, we can also observe the exponential decreasing envelope as the add rates converges in cycles towards the fair rate.

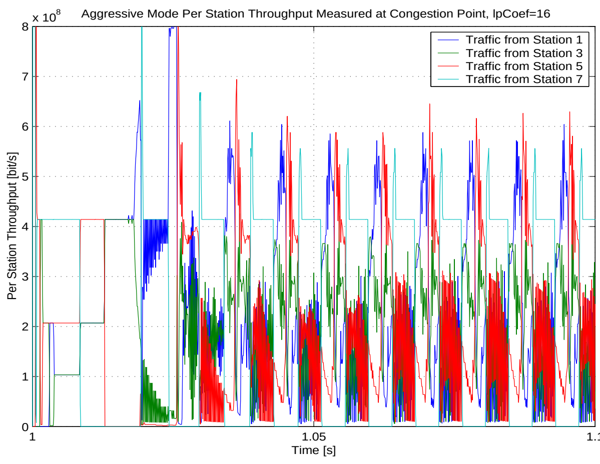


Figure 4: Aggressive mode scenario does not converge to fair rate for contending stations, lpCoef set to 16.

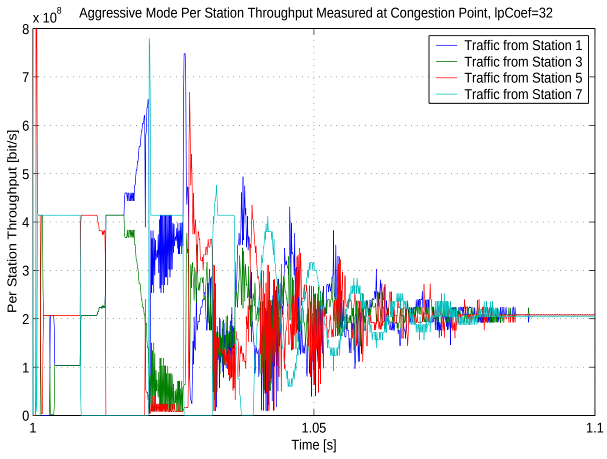


Figure 5: Aggressive mode convergence to fair rate for contending stations, lpCoef set to 32.

IV Conclusions And Further Work

In this presentation we have shown analytically and by simulation how the stability of the RPR *Aggressive mode* fairness algorithm depend on the correct setting of the $lpCoef$ configuration parameter. For the set of allowed values of $lpCoef$, we have shown which ring sizes that can be supported. We have also discussed possible improvements to the *Aggressive mode* fairness algorithm. Directions of further work could be along the lines discussed at the end of chapter II. It would also be interesting to see how such modifications would perform for node/link-failure scenarios.

References

[1] IEEE P802.17 Working Group. Media Access Control (MAC) Parameters, Physical Layer Interface, and Management Parameters, Draft 3.0, November 2003.

[2] Fredrik Davik, Mete Yilmaz, Stein Gjessing, and Necdet Uzun. IEEE 802.17 Resilient Packet Ring Tutorial. *IEEE Communications Magazine*, volume 42(3):112–118, March 2004.

[3] E.R. Hafner, Z. Nendal, and M. Tschanz. A Digital Loop Communication System. *IEEE Transactions on Communications*, volume 22(6), June 1974.

[4] Cecil C. Reames and Ming T. Liu. A Loop Network for Simultaneous Transmission of Variable-length Messages. In *ACM SIGARCH Computer Architecture News, Proceedings of the 2nd annual symposium on Computer architecture*, volume 3(4), December 1974.

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

[6] OPNET Modeler. http://www.opnet.com.