# Improvement of Resilient Packet Ring Fairness

February 2005
Revised August 2005

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

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

***Abstract*— Resilient Packet Ring (RPR) is a recent networking standard developed by the IEEE LAN/MAN working group. RPR is an insertion buffer, dual ring technology, utilizing a back pressure based fairness algorithm to distribute bandwidth when congestion occurs. The fairness algorithm has two modes of operation, called respectively the *aggressive* and the *conservative* fairness modes. For some scenarios, the *aggressive* fairness mode suffers from severe performance deficiencies.**

**In this paper, we propose two novel contributions. The first is a measurement method which enables a node to determine its operating context. The second contribution is a fair rate calculation method, termed the *moderate* fairness mode, which resolves the *aggressive* mode performance deficiencies while retaining several other properties provided by the *aggressive* mode fairness.**

**We compare the performance of the *moderate* fairness mode to that of the *aggressive* and the *conservative* modes by simulations, and find that for some scenarios the *moderate* mode outperforms the *aggressive* and the *conservative* modes. For some other scenarios, the convergence time of the *moderate* mode is somewhat longer than that of the *aggressive* mode.**

Keywords: Resilient Packet Ring, Fairness, Performance evaluation, Simulations, Next generation protocol design and evaluation, Communications modeling, Next Generation Networks Principles, High-speed Networks.

## I. Introduction and motivation

Resilient Packet Ring (RPR) is a new networking standard developed by the IEEE 802 LAN/MAN Standards Committee, assigned standard number IEEE 802.17-2004 [1], [2]. Although RPR was developed by the LAN/MAN committee, it is designed mainly to be a standard for metropolitan and wide area networks.

RPR is a ring topology network. By the use of two rings (also called ringlets), resilience is ensured; if one link fails, any two nodes connected to the ring still have a viable communication path between them. When a node wants to send a packet to another node on the ring, it adds (sends) the packet onto one of the two ringlets. For bandwidth efficiency, the ringlet that gives the shortest path is used by default, but a sender can override this (on a per packet basis) if it for some reason has a ringlet preference. When the packet travels on the ring, it *transits* all nodes between the sender and the receiver. When it reaches the destination, the packet is removed (stripped) from the ring. Hence the bandwidth that would otherwise be consumed by the packet on its way back to the sender (as is the case in the Token Ring [3]), can be used by other communications. Such destination stripping of packets leads to what is commonly known as *spatial reuse*.

RPR uses *insertion buffer(s)* for collision avoidance [4], [5]. When a packet in transit arrives at a node that is currently adding a packet to the ring, the transiting packet is temporarily stored in an insertion buffer, called a *transit queue* in RPR. In order to get some flexibility in the scheduling of link bandwidth resources between add- and transit traffic, the transit queues may be in the order of hundreds of kilobytes large.

In a buffer insertion ring like RPR, a *fairness algorithm* is needed in order to divide the bandwidth fairly[1] between contending nodes, when congestion occurs [6], [7].

The RPR fairness algorithm runs in one of two modes, termed respectively the *conservative* and the *aggressive* modes. The *aggressive* mode of operation is simpler (i.e. requires less code and configuration) than the *conservative* mode, and is used by e.g. Cisco Systems. In this paper, the main focus is on some of the performance deficiencies found in the *aggressive* fairness mode. We do however evaluate the performance of the *conservative* mode fairness for some selected scenarios.

The feedback control system nature of the RPR fairness algorithm makes the amount of add traffic from each sending node oscillate during the transient phase where the feedback control system tries to adjust to a new load [8]. Several papers have reported that in some cases the oscillations decreases and (under a stable traffic pattern) converges to a fair distribution of add rates, while under other conditions, the algorithm diverges, and oscillations continues [8]–[11].

The main contribution of this paper is twofold. First, we propose a novel context determination function, to enable a congested node to determine whether it is utilizing its fair share of bandwidth or not. Secondly, we propose an alternative fair rate calculation mode, termed the *moderate* fairness mode, with the goal of improving on the properties of the *aggressive*

[1]RPR nodes may have different weights, so a fair division does not have to be an equal one. In this paper, however, we assume all nodes have the same weight.

mode fairness. The proposed fairness mode, is designed to fit within the framework given by the current RPR standard.

To evaluate the properties of the three fairness modes, we use performance evaluation by simulations. We have implemented the RPR standard within the the J-Sim and OPNET modeler discrete event simulator frameworks [12], [13]. For this paper however, we use our OPNET implementation.

The rest of this paper is organized as follows: In section II, we present a brief introduction to the RPR fairness algorithm. Then, in section III, we present a class of scenarios where a so-called *modest* head node in a congestion domain induces permanent oscillations in the throughput experienced by the head's upstream neighbors [14]. Then in section IV, we discuss some possible solutions to these problems. Then in sections V and VI, we present our novel contributions handling these problems. In section VII, we evaluate our contributions by simulations and compare the performance to that of the original RPR fairness modes. Finally in sections VIII, IX and X, we present related work, conclude and present possible directions for further works.

## II. The RPR Fairness Algorithm

When several sending nodes try to send over a congested link concurrently, the objective of the RPR fairness algorithm is to divide the available bandwidth fairly between the contending nodes. RPR has three traffic classes: high, medium and low priority. Bandwidth for high and medium traffic is pre-allocated, so the fairness algorithm distributes bandwidth to low priority traffic only. In this paper all data traffic is low priority.

The fairness algorithm is a closed-loop control system [8]. The goal of the fairness algorithm is to arrive at the "Ring Ingress Aggregated with Spatial Reuse" (RIAS) fair division of rates over the congested link [9]. For a congested link, over which each active node has an infinite demand and all nodes have equal weights, the RIAS fair bandwidth will be the capacity of the link divided by the number of active nodes. Fairness for ring segments having active nodes with different weights is not covered by the RIAS reference model. In this paper, all nodes have equal weights, thus we can use the RIAS reference model.

The control system encompasses all nodes that send over the same congested link, known in RPR as a *congestion domain* [15]. The node directly upstream of the most congested link is called the *head* of the congestion domain. The node in the congestion domain that is furthest away from the head is called the *tail* of the congestion domain. Later in this paper we are going to use a scenario depicted in figure 1. Here nodes 0, 1, 2 and 3 all send traffic to node 4. When these nodes in total want to send more than the available bandwidth, the most congested link will be the link immediately downstream of node 3. Thus, the congestion domain will consist of the 4 nodes from node 0 to node 3. Node 0 will be the tail of the domain, node 3 the head.

When a node's total transit and add traffic amounts to more than the full bandwidth, the transit queue of the node with a congested out-link will fill up[2]. When the transit queue occupancy is above a threshold called *low*, the node enters a state called *congested* and if it is has not observed that there are downstream nodes that are more congested, it becomes head of a congestion domain. As head, it starts sending fairness messages, which is the feedback mechanism of the control system. These feedback messages instruct the upstream nodes to restrict their add rate to that of the head's fair rate estimate.

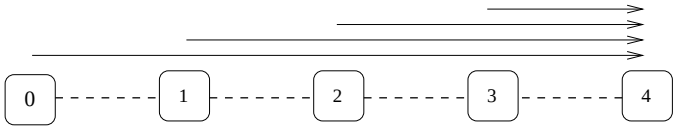



*Figure 1: A congestion domain consisting of 4 active nodes: node 0, 1, 2 and 3. The output link of node 3 is the most congested link in the domain, thus node 3 is the* head-, *while node 0 is the* tail *of the congestion domain.*

When a fairness message is received by upstream nodes in a congestion domain, these nodes restrict their add rate to the value of the encapsulated fair rate estimate. The head estimates and advertises new fair rates estimates periodically (every aging interval (termed $agingInterval$)). The default duration of an $agingInterval$ is 100 $\mu$s.

The time it takes from the head advertises a new fair rate estimate, until it sees the effect of this action, is the time it takes for a fairness message to reach an upstream node, and then the time it takes for the packets from this node, generated in accordance with the newly received fair rate estimate, to reach the head. Hence, in general there is a considerable feedback latency in the control system. This latency, which can be considered the time-constant of the congestion domain, combined with the configuration of the algorithm in the head for calculating fair rate estimates, decides the stability of the RPR fairness algorithm.

The RPR standard has defined two algorithms for the calculation of the fair rate estimates. In this paper, although the main focus is on the improvement of deficiencies found in the *aggressive* fairness mode, in section II-A, we provide a brief description of the *conservative* fairness mode. For a more detailed overview of the *conservative* fairness mode, refer to [2], [16]. Following the brief overview of the *conservative* fairness mode, we provide a brief overview of the *aggressive* fairness mode in section II-B. For a detailed overview of the *aggressive* fairness mode, refer to [8].

### A. Conservative Fairness Mode

In the *conservative* fairness mode, when using the node design with two transit queues, the goal of the fairness algorithm is to maximize the throughput of the congested link, while at the same time keeping the occupancy of the

[2]RPR nodes may have one or two transit queues. In the case of a node with two transit queues, the highest priority traffic will use one transit queue, while the two lower priority classes will use the other transit queue. In the RPR model used in this paper there are two transit queues, but since all data traffic will be of the lowest priority, the high priority transit queue will be empty.

transit queue for low and medium priority traffic (termed the Secondary Transit Queue (*STQ*)) between two thresholds termed *low* and *medium*. The layout of the *STQ* with associated queue thresholds and their default level relative to another, higher threshold, termed *high*, is shown in Fig. 2.

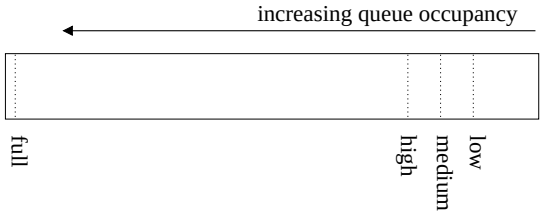



*Figure 2: Secondary Transit Queue and associated queue thresholds. Note that in the figure, the relative location of the thresholds w.r.t. the high threshold is according to the default settings specified by the RPR standard. This layout applies to the* aggressive *fairness mode as well. In the* aggressive *fairness mode however, the* medium *threshold is not used.*

In this fairness mode, to allow for a more modestly sized *STQ* than is used for the *aggressive* fairness mode, several techniques are used: i) once the *STQ* occupancy exceeds the *medium* threshold, the actions taken to reduce the queue occupancy are more *aggressive* (fair rate estimate reductions are done faster) than the corresponding actions taken to increase the queue occupancy once the occupancy falls below the *low* threshold; ii) to prevent that the fair rate estimate is adjusted faster (or slower) than the effect can be observed, the time-constant of the congestion domain is measured and used to set a timer termed Fairness Round Trip Time (FRTT).

Thus, once the fair rate estimate has been adjusted, the FRTT timer is reset and no additional rate adjustments are performed before the FRTT timer expires.

To provide a means of congestion recovery, should the *STQ* occupancy exceed the *high* threshold, the fair rate estimate is reduced every $agingInterval$, regardless of the status of the FRTT timer.

### *B. Aggressive Fairness Mode*

In the *aggressive* fairness mode, once a node utilizing the two-transit buffer design has become congested, it continues to add traffic until the transit queue occupancy reaches the *high* threshold (shown in Fig. 2). At this point, the head stops its add traffic, until the upstream nodes have reduced their send rate so much that the head's transit queue occupancy decreases below the high threshold.

We have previously shown that for the *aggressive* fairness mode, the value used by the head as its fair rate estimate, is the head's own add rate run through a 2nd order low-pass filter. Below, we will give the reader a short summary of the properties of this filter. A detailed analysis of the *aggressive* fairness mode with associated stability properties can be found in [8].

The low-pass filter is shown in Fig. 3. In the figure, the box with the marking $z^{-1}$ denotes that the value on the output of the box is delayed one sampling period (i.e. one $agingInterval$) as compared to the value on the input of the box. The filter input and output-values, denoted respectively $X(z)$ and $Y(z)$ are the Z-domain representations of the discrete time-domain signals $x(n)$ and $y(n)$, where $x(n)$ is the add rate of the head and $y(n)$ is the low-pass filtered version, $lpAddRate(n)$.

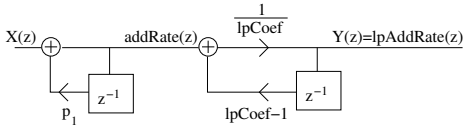



*Figure 3: Block diagram of the second-order low-pass filters yielding the fair rate estimate based on the congestion head's own send rate* $x(n)$

In the figure below, the value of the constants $p_1$ and $p_2$ comes from the RPR configuration parameters *ageCoef* and *lpCoef* as shown below:

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

$$p_2 = \frac{lpCoef - 1}{lpCoef},\ lpCoef \in \{16, 32, 64, 128, 256, 512\} \tag{2}$$

This gives a transfer function, $H(z)$, of the filter as shown in (3) below.

$$H(z) = \frac{Y(z)}{X(z)} = \frac{1}{lpCoef} \cdot \frac{z^2}{(z - p_1)(z - p_2)} \tag{3}$$

Under the assumption that the input to the filter, $x(n)$ is a step-function, the output of the filter consists of a constant part and two first-order filter functions. Further, when $lpCoef > ageCoef$, the filter time-constant is dominated by the second filter stage. In this case, the time-constant, $\tau$, of the filter, can be approximated by the expression:

$$\tau \approx \frac{-1}{ln(p_2)} = \frac{-1}{ln(\frac{lpCoef-1}{lpCoef})} \approx lpCoef\ [agingIntervals] \tag{4}$$

It has been shown that the *aggressive* mode fairness algorithm does not always reach a stable state [8]. In general, the calculated fair rate estimate always varies (oscillates) initially in response to transient load conditions. If the (new) traffic load is stable, these oscillations should ideally decay as the fairness algorithm converges to the new fair division of sending rates. For some scenarios however, even under (new) stable load conditions, the *aggressive* mode fairness algorithm does not converge, and the rate at which each different node is allowed to send, continues to oscillate.

In the next section, we will present and discuss a class of scenarios, for which the *aggressive* mode fairness algorithm does not converge.

## III. Unbalanced Traffic Scenario

As described above, congestion occurs when several senders at the same time want to transmit more data than the link capacity (bandwidth) over the same link can sustain (all links in RPR have the same capacity). Some senders may be greedy, i.e. they want to send as much as possible. Other senders send at a limited rate. For modest senders, i.e. senders sending less than their fair share, RPR should not impose any rate restrictions. For nodes having more to send than their fair share, RPR should restrict their sending rate to their fair share.

In the case where the head of a congestion domain is a modest sender, utilizing the **aggressive** fairness mode, this induces permanent oscillations in the throughput of its upstream greedy senders. An example of this, using 1Gbit/s links, is shown in figure 4 below.

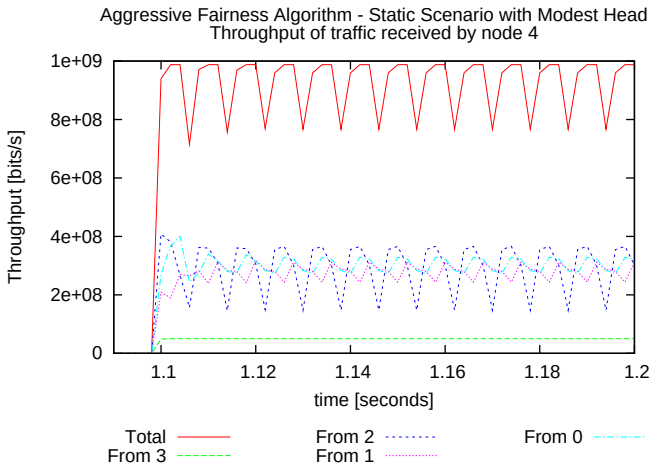



*Figure 4: Modest congestion head node. The figure plots throughput per sender node measured at node 4.*

As seen in the figure, the induced oscillations in the throughput of the individual nodes upstream of the head causes the aggregate throughput of the congested link to oscillate. This leads to a reduction in link utilization of the congested link. For illustrative purposes, we have checked the actual numbers of this scenario. When the congestion domain head is configured as modest sender, adding traffic at 5% of the link rate, the link-utilization converges toward a value of 92.8%. For the same scenario, if we configure the head as a greedy sender, the link utilization for the congested link converges to a value of 99.8%. I.e., for a 1Gbit/s link-speed, we incur a throughput loss of 70 Mbit/s. For scenarios where the head node is a modest sender, the actual throughput loss depends on several factors, notably the node configurations and the size of the congestion domain.

Another effect of the throughput oscillations, is that the occupancy of the *STQ* in the head will vary more, thus leading to greater delay variations for medium and low-priority traffic.

The origin of this oscillatory behavior comes from the method used by the *aggressive* fairness mode to control the send rate of its upstream neighbors. In the event of a congestion situation, the head of the congestion domain uses its own add rate as its fair rate estimate, deciding how much its upstream neighbors should be allowed to send over the congested link. As we already know, the (modest) head adds traffic over the congested link at a rate that is below the fair rate. Thus as long as the *STQ* occupancy of the head exceeds the *low* threshold, the rate values encapsulated in the fairness messages will be the head's own modest add rate. The effect of these fairness messages, when received by the upstream neighbors, is that for a period o f time, the aggregate of traffic crossing the congested link will be less than the capacity of the same link. As noted above, this results in a reduction in the link utilization of the congested link.

Before we proceed to the description of our contributions, in the next section, we present the goals we seek to achieve.

## IV. Improving on Stability and Throughput Performance for Modest Head Scenarios

As discussed and shown in section III, the presence of a modest head in a congestion domain where the head utilizes the *aggressive* fairness mode, leads to throughput loss and consequently reduced link utilization for the most congested link in the congestion domain. Now the question becomes – is there anything that can be done to improve the performance of the *aggressive* fairness mode in this type of scenarios?

**One alternative** would be to implement an alternative fairness algorithm where this type of problem is not an issue [9], [10], [17].

**A second alternative** is to modify the current RPR *aggressive* fairness mode in order to improve its performance in the presence of a modest head [11]. An inherent constraint for this alternative is that the proposed modification must fit within the given RPR standard framework w/o yielding undesirable side-effects, such as performance degradations for other traffic scenarios.

**A third alternative** would be to design an alternative fairness mode, that fits within the RPR framework, resolves the above performance deficiencies and can be used regardless of the sending behavior of the congestion head.

Our proposed solution belongs to the third alternative above, but can also be used as outlined for the second alternative (i.e. as an alternative to the *aggressive* fairness mode, when the congestion head has a modest sending behavior).

In the context of the second alternative, it is clear that there must be a way for a congestion head to determine its operating context, to know whether it is operating as a modest head or not (i.e. to decide which fairness "mode of operation" to use).

Based on the above discussion, we are ready to introduce a set of Design Objectives (DOs) which we seek to meet when designing our contributions.

*DO 1:* Remove the oscillatory behavior in the presence of a head with a modest sending behavior.

*DO 2:* Retain the behavior of the original algorithm in the presence of a head with a demand greater than or equal to its fair share.

*DO 3:* Minimize the changes (state information, code and processing requirements) to the current algorithm.

*DO 4:* Fit our modifications into the framework given by the RPR standard.

*DO 5:* Allow for interoperability of nodes in a congestion domain regardless of the fairness mode used.

In the next section (section V), to allow for the fulfillment of the above Design Objectives in the context of alternative 2 outlined above, we propose a context determination method. Then in section VI, we describe our *moderate* fairness mode.

## V. Monitoring of Congestion Head's Bandwidth Utilization

We can envision at least two strategies for providing the context determination method described in the previous section.

One way is to compare the head's own add rate, *lpAddRate*, to the total available bandwidth divided by the number of active nodes (the fair rate for a congestion domain with all greedy senders having equal weights). If the head is sending at a rate close to or above this ratio, one can assume that the head's own add rate is a good rate estimate that can be used for distribution to its upstream neighbors. If the head is sending at a rate below this ratio, we can assume that the head's capacity demand is below the head's fair share and that the head's own add rate should not be used as a fair rate estimate.

This is the method used by Zhou et al. in [11].

There are several problems with this method. **Firstly**, this requires knowledge about the number of nodes sending traffic over the congestion point. The calculation of the number of nodes sending traffic over the congestion point is not mandatory for nodes running the *aggressive* fairness mode. **Secondly**, and more seriously; the use of a rate threshold for switching between two different rate calculation methods may lead to instabilities. This is especially true when the rate threshold is set to the fair rate. As the rate algorithm converges towards the fair rate, it is expected that the add rate of the congestion head will oscillate around the fair rate [8]. Thus this will cause the algorithm to constantly switch between two different rate calculation methods. As seen in their paper, the oscillations are reduced by a factor 2 as compared to the original RPR fairness algorithm, but the magnitude of the oscillations are still 50% of the line rate. The throughput improvement is 13.4%, leading to a link-utilization of ≈87%.

**Thirdly**, below, we will argue that the basic underlying assumption, that the head is adding traffic at a rate $R_{head} \geq \frac{C}{|\mathcal{X}|}$ (where $C$ is the link capacity and $\mathcal{X}$ is the set of nodes sending traffic over the congested link), is not an indication of the correctness of the head's fair rate estimate.

Let us assume that the above assumption was true, and that we have a scenario, with the set $\mathcal{X}$ of active senders, sending traffic over the congestion point.

With a head adding traffic at a rate of $\frac{C}{|\mathcal{X}|}$, then the head's fair rate estimate ($R_{fair\ estimate}$) would be given by (5) below.

$$R_{fair\ estimate} = \frac{C}{|\mathcal{X}|} \tag{5}$$

Further, let us assume that the congestion domain has a set of nodes $\mathcal{Y}$, where each node $i \in \mathcal{Y}$, is sending traffic at a rate $R_i$, lower than the rate of the head ($R_{head}$), while the remaining nodes, $|\mathcal{X}| - |\mathcal{Y}| - 1$, send as much as possible. We will now show the error in the fair rate estimate, given by (5) and the theoretical fair rate, shown in (6) below.

$$R_{fair} = \frac{C - \sum_{i \in \mathcal{Y}} R_i - \frac{C}{|\mathcal{X}|}}{|\mathcal{X}| - |\mathcal{Y}| - 1} \tag{6}$$

Thus the error, $E$, in the fair rate estimate is given by (7) below.

$$E = R_{fair} - R_{fair\ estimate} = \frac{C - \sum_{i \in \mathcal{Y}} R_i - \frac{C}{|\mathcal{X}|}}{|\mathcal{X}| - |\mathcal{Y}| - 1} - \frac{C}{|\mathcal{X}|} \tag{7}$$

As shown in (8) below, we can make some general observations for this function.

$$\sum_{i \in \mathcal{Y}} R_i \to 0 \text{ and } |\mathcal{Y}| \to (|\mathcal{X}| - 2) \Rightarrow E \to C \cdot \frac{|\mathcal{X}| - 2}{|\mathcal{X}|} \tag{8}$$

I.e. as the demand of the nodes in the set $\mathcal{Y}$ decreases and the number of nodes in the set approaches the remaining number of nodes in the congestion domain except for one (the worst-case), the error approaches $C \cdot \frac{|\mathcal{X}|-2}{|\mathcal{X}|}$. Thus as the size of the congestion domain ($|\mathcal{X}|$) increases, $E \to C$. I.e. the error of the fair rate estimate approaches the link-rate.

In conclusion, it is clear that the use of the node's own add rate as a fair rate estimate results in large (fair rate estimation) errors for some scenarios. Furthermore, it is clear that it is not possible for the head to reason about the correctness of the use of its add rate as a fair rate estimate, purely based on its knowledge of the link-rate, its own add rate and the number of active senders over the congested link

A second, and novel strategy, for monitoring whether the congestion head is utilizing its fair share of the bandwidth, is to measure the fraction of time when the client is allowed to send traffic but chooses not to do so. By this, a congested node is able to intelligently reason about the usability of its own add rate as a fair rate estimate. Given the design of the RPR MAC layer, this can be easily accomplished. By monitoring the signal (denoted *sendC* in the standard) sent from the MAC layer to the MAC client we can measure the aggregate of the fraction of the time that i) this signal indicates that traffic can be transmitted on the ring, regardless of the destination

address and ii) the MAC layer does not receive a packet from the MAC client. This aggregate represents the link capacity the local node chooses not to use for local traffic. For a node with a traffic demand equal to or greater than its fair share, this aggregate will be zero (as it always have something to transmit when it is allowed to).

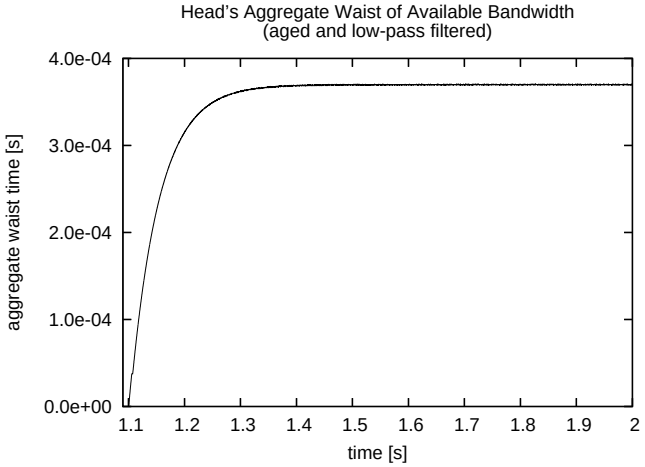



*Figure 5: Modest congestion head node. The figure plots the aged and low-pass filtered measured aggregate of the time the head node chooses not to utilize the available bandwidth for transmission of local traffic.*

For the scenario shown in figure 1, with a modest head, we have measured this (aged and low-pass filtered) aggregate. This is shown in figure 5. By use of the same aging and low-pass filtering as is used on other measured variables in the RPR standard, we get a value which will converge towards $ageCoef \cdot agingInterval = 400\ [\mu s]$ as the amount of added traffic by the head approaches 0 (a very modest head). For the specific scenario shown here, we see that the long term average waist (denoted *lpWaistTime*) of the head is $\approx 370\ [\mu s]$. Thus clearly, the value of $lpWaistTime > 0$ and by inspecting the value of *lpWaistTime*, the head will know that it is operating in a modest head context and that it should not use its own add rate as its fair rate estimate.

Given this context determination function, we are ready to discuss the problem of calculating a fair rate estimate for distribution by the head to its upstream neighbors.

## VI. Alternative Fair rate estimate calculation

Given that the head of a congestion domain is not utilizing fully the bandwidth available for local traffic, we know that the use of its local add rate as a fair rate estimate, results in permanent throughput oscillations for its upstream active neighbors. This also leads to a reduction in link utilization of the congested link.

From previous work, we know that the way rate adjustments are performed are critical for the stability of an RPR ring [8]. If rate adjustments are made too fast, not allowing enough time to observe the effect of previously issued fair rate estimates, the fairness algorithm will not converge to the fair rate. Similarly, if rate adjustments are made too slowly, the algorithm will converge, but may result in unfair division of bandwidth and/or reduced link-utilization during the transient period as well as prolong the duration of the transient period.

If the new rate estimation method differs greatly from the method currently used, this may alter the convergence/stability properties of the fairness method significantly. Thus warranting a complete study of all aspects of the stability and fairness properties of the proposed algorithm.

As stated in the introduction, the goal of the control system is to provide a fair sharing of the bandwidth of the congested link. In the case of a modest head, this means a fair sharing of the available bandwidth minus the fraction used by the head. We know that the theoretical fair rate value to be distributed to the upstream neighbors reside somewhere in the region $\langle lpAddRate, bandwidth - lpAddRate \rangle$ (where $lpAddRate$ is the add rate of the congestion domain head). The exact value depends on the number of upstream active nodes, their weights and their demand. In the case of all greedy senders having equal weights, the fair rate value to distribute to upstream node would be $\frac{bandwidth - lpAddRate}{active\ senders - 1}$. The head node is regarded as an active sender, but does not have any additional bandwidth demands, thus we subtract 1 from the active sender count.

Further, let us assume that an additional requirement to the fairness algorithm is to maintain the *STQ* occupancy at a relatively constant level above 0 and below the *high* threshold. By doing this, we obtain three things: i) we keep the transit delay (for low and medium priority traffic) through the congested node at a relatively fixed level; ii) we ensure that the node always have traffic to send once the output-link becomes available, thus avoiding under-utilization of the link; and iii) we ensure that local traffic gets its fair share (or a fraction of it in the case of a modest head) as the RPR scheduling algorithm ensures the scheduling of both local (within the estimated fair rate bound) and transit traffic while the *STQ* occupancy is below the *high* threshold.

Now, what can be done to achieve this? For the simplicity of the discussion, we assume that the demand of the upstream nodes is stable and that they always have more to send than the capacity of the congested link allows. When the head of the congestion domain calculates a fair rate estimate to be distributed to its upstream neighbors, we know that at some future time, the resulting aggregate traffic from the upstream nodes is either too high (the *STQ* occupancy will increase), too low (the *STQ* occupancy will decrease) or correct[3] (the *STQ* occupancy remains constant).

To achieve a fair division of bandwidth, we monitor the **occupancy** of the *STQ*, just like in the original algorithm. In addition to this, we also monitor the **growth-direction** of the *STQ* occupancy.

[3] A *STQ* occupancy that remains constant at 0 or at the *high*-threshold is not an indication of a correct rate-estimate. But these are special cases that requires special handling.

### A. Operation of the Moderate fairness mode

The *aggressive* fairness mode declares a node as *congested* when the *STQ* occupancy exceeds the *low* threshold. From this point onwards, as long as the *STQ* occupancy remains above the *low* threshold, the node sends back-pressure messages to its upstream neighbors. When the *STQ* occupancy falls below the *low* threshold again, the node reenter the *uncongested* state and the node signals to its upstream neighbor(s), that they are allowed to increase their add rate. This rate increase is done according to their own configuration settings.

In the *moderate* fairness mode, a node (that was previously not congested) becomes congested when the *STQ*-occupancy exceeds the *low*-threshold. Once a node has become congested however, the transition back to the uncongested state is done first once i) the *STQ*-occupancy has fallen below the *low*-threshold, and ii) the fair rate estimate equals the maximum allowed rate. The slight modification associated with the transition from the congested back to the uncongested state is to avoid loosing control over the sending behavior of upstream nodes, during transient periods where the *STQ* occupancy falls below the *low* threshold.

In the absence of a locally available and usable fair rate estimate (remember that for a modest head scenario, the use of the node's own add rate, $lpAddRate$, prevents the *aggressive* fairness mode to stabilize), we introduce a locally maintained **rate estimate**. Let us denote this estimate *mRate* (moderate Rate estimate). We also maintain aged and low-pass filtered versions of this variable, denoted respectively *ageMRate* and *lpMRate*. The value of ***lpMRate* is the fair rate estimate**, which is distributed to the upstream neighbors in the fairness messages.

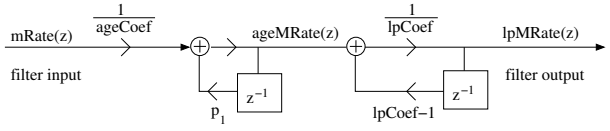



*Figure 6: Second order low-pass filter as applied to the rate estimate $mRate$. The output of the filter, $lpMRate$, is the fair rate estimate, distributed to upstream neighbors by a congestion head.*

We use the same[4] two-stage second-order low-pass filter construct, used by both the *aggressive* and the *conservative* modes as shown in Fig. 6. In section II, we showed that for the values used for the configuration settings $ageCoef$ and $lpCoef$ ($p_1 = \frac{ageCoef-1}{ageCoef}$), this filter can be approximated by a first-order low-pass filter with a time-constant, $\tau \approx lpCoef \cdot agingInterval\ [s]$. In our *moderate* mode, the input to this filter is the variable, *mRate*, while the output of the filter, $lpMRate$, is the fair rate estimate.

The basic idea of our fairness mode is simple. If the *STQ* occupancy of the congested node increases, the fair rate estimate is too high and must be decreased. Correspondingly, if the *STQ* occupancy of the congested node decreases, the fair rate estimate is too low and must be decreased.

[4]With the exception of a divisor $ageCoef$ applied at the input.

To aid the convergence process, we use a rate interval, limited by a maximum and a minimum value, denoted respectively $mRateMin$ and $mRateMax$. The purpose of this rate interval is to ensure that, during increasing *STQ* occupancy, the use of the minimum rate value as the fair rate estimate, guarantees (at some future time) a decreasing *STQ* occupancy. Correspondingly, the use of the maximum rate value as the fair rate estimate, guarantees (at some future time) an increasing *STQ* occupancy. The challenge then, is to maintain and use this interval, so that the *STQ* occupancy can be kept at a relatively constant level (i.e. at the *low* threshold) and at the same time avoid that the *STQ* occupancy falls to 0 or exceeds the *high* threshold. Note that the size and location of the rate interval, as given by the values of $mRateMin$ and $mRateMax$, will change during the convergence process as well as in the case of a change in the applied load.

Upon transition from the uncongested to the congested state, we initialize $mRateMin$ to the value of the node's own send rate, $lpAddRate$. The value of $mRateMax$ is set to half of the maximum allowed rate. We also set the fair rate estimate, *lpMRate*, to $mRateMin$, while the low-pass filter input (see Fig. 6), *mRate*, is set to *mRateMax*.

Below follows a description of one iteration of the cyclic convergence process, starting as we have an increasing *STQ* occupancy at the head, and the *STQ* occupancy is below the *low* threshold.

When the *STQ* occupancy increases, once the *STQ* occupancy exceeds the *low* threshold, we decrease the fair rate estimate towards the minimum value of the rate interval ($mRateMin$) by setting the input of the low-pass filter to $mRateMin$. Once the fair rate estimate has been sufficiently reduced, the *STQ* occupancy will start to decrease. At this point for the given load, we know that in the future, to ensure a decreasing *STQ* occupancy, the fair rate estimate need not be set lower than its current value. Thus, we set $mRateMin$ to the current fair rate estimate.

Next, to oppose the decreasing *STQ* occupancy, we increase the fair rate estimate towards the maximum value of the rate interval ($mRateMax$).

Depending on the *STQ* occupancy, this is done in one of two ways. If the *STQ* occupancy is above the *low* threshold, to avoid increasing the rate estimate too much and too fast, we set the input of the low-pass filter to $\frac{mRateMin+mRateMax}{2}$. If this is not enough, and the *STQ* occupancy falls below *low* threshold, the input of the low-pass filter is set to $mRateMax$.

Once the fair rate estimate has been sufficiently increased, the *STQ* occupancy will, as a result, start to increase. At this point for the given load, we know that in the future, to ensure an increasing *STQ* occupancy, the fair rate estimate need not be set higher than its current value. Thus, we set $mRateMax$ to the current fair rate estimate. By this, the cycle is concluded and we are back to the starting point. For each iteration of this cycle, the size of the rate interval is improved (reduced).

The actual increase/decrease behavior follows an exponential ramp function, given by the properties of the second-order

low-pass filter shown in Fig. 6. During the periods[5] of increasing *STQ* occupancy, the filter-input is set to $mRateMin$, thus ensuring a monotonic decrease of the fair-rate estimate towards $mRateMin$. Correspondingly, during the periods of decreasing *STQ* occupancy, the filter-input is set to $mRateMax$, thus ensuring a monotonic increasing of the fair-rate estimate towards $mRateMax$.

The exponential ramp function ensures that the time, used to adjust the fair-rate estimate between the max/min values, remains constant, regardless of the size of the rate-interval. Thus, during the convergence process, the narrower the rate interval gets around the RIAS fair rate, the slower the fair rate estimate is adjusted. This way, the variations in throughput during steady-state are minimized.

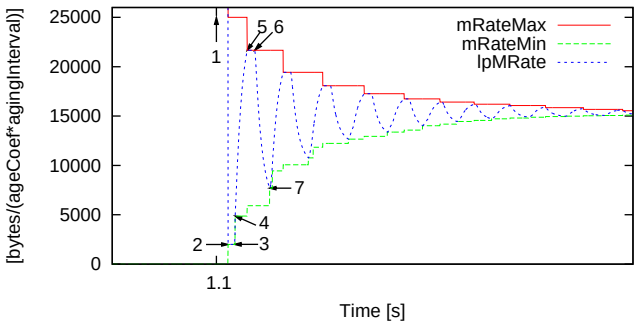
For the simple scenario shown in Fig.1, we will show the convergence process of our *moderate* fairness mode. In the scenario, we assume a stable demand by the individual nodes. The example illustrates, without loss of generality, the convergence for the transition between a no-load situation, and a max-load situation (i.e. a worst-case situation). For a dynamic scenario, the load-change is typically much smaller. Thus in this case, the task of the fairness mode is to shift the established rate interval higher or lower, so that the new fair rate is included in the interval. This is done by expanding the rate interval on one side before continuing the convergence cycle.

Below, the convergence towards the fair rate is illustrated for the scenario shown in Fig. 1, using 410$\mu$s (82km) link-delays and a value of 32 for the lpCoef parameter. The convergence of the fair rate estimate, *lpMRate*, is shown in Fig. 7a while the corresponding *STQ* occupancy for the *head* during the same period is shown in Fig. 7b. Significant points in the plots and the discussion have been marked with labels.

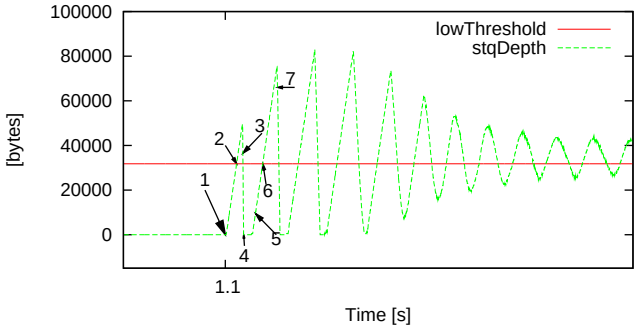
Let us assume there are currently no active nodes on the ring. At time $t_1$ (the point labelled 1), nodes 0, 1, 2 and 3 all start to transmit traffic to node 4. This will cause the *STQ* occupancy of node 3 to start to fill.

At the point labelled 2, node 3 becomes congested, thus we perform the initial variable assignments as described above. From this point, the value of *lpMRate* will increase towards *mRateMax*. The effect on the *STQ* occupancy is that at first, it will have a transient initial increase, before the effect of the reduced rate value is observable at the head (at point 3). As long as the *STQ* occupancy is above the *low* threshold and decreasing, to oppose the decreasing *STQ* occupancy, we set the input to the low-pass filter to $\frac{mRateMax+mRateMin}{2}$. If this is not sufficient to oppose the negative *STQ* growth. Once the *STQ* occupancy falls below the *low* threshold, we set the input to the low-pass filter to *mRateMax*. The result is seen at point 5, where the increased output-value of the low-pass filter results in an increase in the *STQ* occupancy. At this point, the value of *mRateMax* is replaced by the current output value from the low-pass filter, *lpMRate*. Thus effectively reducing

[5]The description in this paragraph is a slightly simplified version of that provided above, and is thus not entirely correct.




(a) Fair Rate Estimate Convergence




(b) STQ Occupancy

*Figure 7: Illustration of fair rate estimate and STQ occupancy during rate convergence process, using the simulated scenario with a value of 32 for lpCoef and 410$\mu$s link-delays (82 km links).*

the interval of rates, $\langle mRateMin, mRateMax\rangle$, by improving (decreasing) the maximum value. From this point, until the *STQ* occupancy exceeds the *low* threshold, we keep the input to the low-pass filter constant at *mRateMax*, thus keeping the rate value sent to upstream nodes constant. At point 6, the *STQ* occupancy once again exceeds the *low* threshold, and to prevent *STQ* overflow, we set the input to the low-pass filter to *mRateMin*. This results in a decrease in the output-value of low-pass filter. The effect of this is seen at point 7, where the growth-direction of the *STQ* occupancy once again becomes negative. At this point, we have a new estimate for the minimum rate-value, *mRateMin*, thus effectively reducing the interval of rates, $\langle mRateMin, mRateMax\rangle$, by improving (increasing) the minimum value. At point 7, the current rate adjustment cycle is concluded and the next cycle starts. By this, we are back to the starting point. At point 7, the only difference is that when entering the cycle starting at 7, the interval of rates have been (significantly) reduced compared to the size of the interval when entering the previous cycle (at 3).

The reader may have noticed that an explanation of the point labelled 4 has been omitted. At this point, the *STQ* has become empty (because the value of *mRateMin* is currently too low). To rectify this, and to minimize underutilization of the link, the value of *mRateMin* is corrected slightly based on its current value and the value of *mRateMax*. A similar corrective action is performed if the *STQ* occupancy reaches

the *high* threshold. A second corrective action we perform to avoid underutilization of the congested link, as long as it stays within rate restrictions (if such exist) received from downstream nodes, is to allow the congested node (the head) to add traffic to the output-link as long as the transit queue is empty.

Another observation the reader may have done, is that the points (i.e. 3, 5 and 7) where the *STQ* growth-direction changes is not exactly at the local minimum/maximum points for *STQ* occupancy. This is to avoid making any rate adjustments based on accidental variations in *STQ* occupancy (i.e. measurement "noise").

### *B. Comparison of the Moderate and the Aggressive Fairness Modes*

The *moderate* fairness mode may seem overly complex compared to the *aggressive* fairness mode. Added complexity however, is difficult to overcome. For a greedy head, running the *aggressive* fairness mode, whenever the fair rate estimate is too low, resulting in decreasing *STQ* occupancy, the *head* may quickly (almost immediately) compensate for this using locally added traffic. And vice versa, when the fair rate estimate is too high, local traffic can be held back. This makes it possible to quickly establish self-adjusting rate intervals where the lower rate limit is established once the *STQ* occupancy exceeds the *high* threshold and the higher rate limit is established once the *STQ* occupancy falls below the *high* threshold. The rate limits are not explicitly stored in separate variables, but the value of the local *addRate* rate counter, which is the output signal from the first low-pass filter for the *aggressive* fairness mode, is quickly (in a matter of a few aging intervals) updated to the new rate limit for both the increasing and decreasing rate periods.

The *moderate* fairness mode can in some sense, be considered a generalized version of the *aggressive* fairness mode. In the case of *moderate* fairness, we do not (and cannot) rely purely on the capability of the head to compensate for bad fair rate estimates by the increase or throttling of local traffic. Thus we establish and maintain explicit rate intervals to aid the convergence of the fair rate calculation process. Further, we cannot use the node's measurements of locally added traffic as the fair rate estimate, as the correlation between the node's own add rate and the congestion state is rather weak. Thus we have to introduce a separate rate variable (much the same way as is done for the *conservative* fairness mode). Once a congestion state is established, we regulate the value of the rate estimate within the established rate interval, based on the *STQ* occupancy level and growth direction.

As discussed in section III and shown in Fig. 4, the establishment of a rate interval based on measurements of local traffic only, results in large oscillations in throughput and reduced utilization of the congested link. This is because the rate interval which is established by measurements of local traffic is incorrect – it is not centered around the fair rate. In an interval of rates, where the fair rate should be the center, the center of the interval is effectively placed at a negative offset, where the offset is decided by the amount of traffic transmitted by the *head*.

For the *moderate* fairness mode, once a fair rate estimate has been calculated, we do not see the the full effect of it before a system time-constant later. Further, the establishment and maintenance of maximum and minimum rate values can only be done by observing local maxima and minima points of the *STQ* occupancy. Thus when using the *moderate* fairness mode, the establishment of a rate interval takes more time than for the *aggressive* mode.

However, as we shall see in section VII, the performance of the *moderate* fairness mode, when used in the presence of a greedy head, is comparable to that of the *aggressive* fairness mode.

## VII. Performance Evaluation

In this section, we present results for a set of carefully selected simulation scenarios. The goal of this selection is to evaluate and compare the performance of our proposed *moderate* fairness mode to that of the *aggressive* and the *conservative* fairness modes for some fundamental load-patterns. Having done this, it is should be possible to predict the behavior of our *moderate* fairness mode for many other load-patterns.

Thus in section VII-A, we start by comparing the behavior of the *moderate* fairness mode to that of the *aggressive* and *conservative* fairness modes when the head has a greedy sending behavior. Next, in section VII-B we compare the performance of the *moderate* fairness mode to that of the *aggressive* and the *conservative* fairness modes when the head has a modest sending behavior. In section VII-C, we demonstrate the ability of the *moderate* fairness mode to adapt to a dynamic load scenario. Finally, in section VII-D, we have run an extensive series of simulations, where we for each test scenario and each allowed value of the *lpCoef parameter*, test the ability of the *aggressive* and our *moderate* fairness modes to converge to the fair division of rates as the size of the tested network is increased.

In our paper "Congestion Domain Boundaries in Resilient Packet Rings" [15], we analyzed a problem, where the the tail of a congestion domain stops the propagation of fairness messages received from a downstream head. The negative side-effect of this behavior is that a node upstream of the tail may send excessive amounts of traffic, thus preventing the convergence of the fairness algorithm. Thus we incur both unfairness as well as non-convergence of the fairness algorithm. In the same paper, we proposed and evaluated a modification, termed the *tail-fix*, that solved these problems by a conditional propagation of the fair rate estimate beyond the tail of a congestion domain. In the performance evaluation experiment that follows, we make use of the *tail-fix* when evaluating the performance of our proposed *moderate* fairness mode.

### *A. Convergence when head has a greedy sending behavior*

In this experiment, we compare the performance of the *aggressive*, the *moderate* and the *conservative* fairness modes

in the presence of a greedy head node. Further, in the first part of the experiment, we use the context determination function introduced in section V to allow for the selection between the *aggressive* and the *moderate* rate calculation modes, in accordance with a node's sending behavior and congestion status.

We use the same scenario as before (depicted in Fig. 1), but for this test, all nodes send as much traffic as possible. The per node configuration is shown in Table I.

| **Parameter Name** | **Value** |
|---|---|
| Line Rate | 1 [Gbit/s] |
| Packet size | 500 [B] (fixed) |
| All nodes (in congestion domain) sending behavior | GREEDY |
| STQ Thresholds<br>- low<br>- high | <br>31812 [bytes]<br>120000 [bytes] |
| rampUpCoef | 64 |
| rampDnCoef | 64 |
| ageCoef | 4 |
| lpCoef | 64 |
| link-delay | 410 $[\mu s]$ |
| Start of traffic | 1.1s |

*Table I: Configuration for comparison of the three fairness modes in the presence of a greedy head.*

First, we let the context determination function (i.e. the value of the *lpWaistTime* variable), decide whether the head should use the *aggressive* or the *moderate* fairness modes. In the case of a greedy head. The value of the *lpWaistTime* variable remains at 0, thus the *aggressive* fairness mode is used. Next, we force the use of the *moderate* fairness mode, regardless of the value of the *lpWaistTime* variable. Finally, we force the use of the *conservative* fairness mode, regardless of the value of the *lpWaistTime* variable.

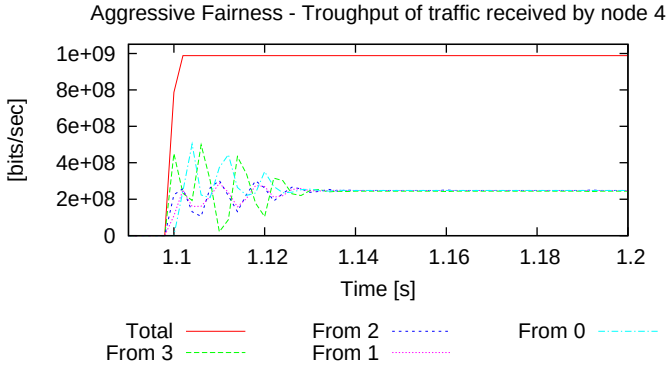



*Figure 8:* Aggressive *fairness mode with a greedy head. This type of scenario is optimal for the* aggressive *fairness mode, providing faster convergence than the two other fairness modes. At time 1.028s throughput convergence is achieved.*

We start by comparing the throughput convergence of the three fairness modes. From the measurements performed, we calculate the time it takes for the throughput of all flows to stabilize at a level within $\pm 5\%$ of the steady-state (fair) value. For the *aggressive* fairness mode throughput convergence, shown in Fig. 8, we found as expected that its associated

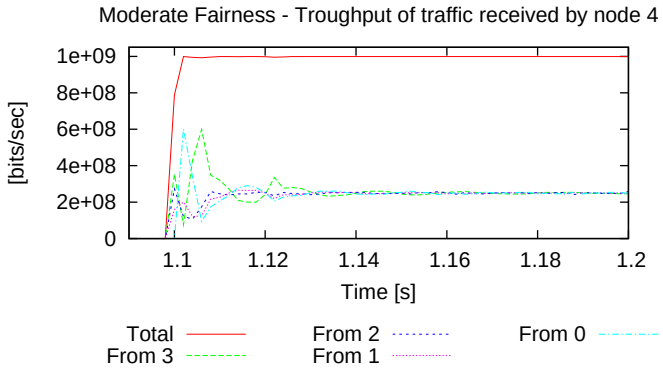



*Figure 9:* Moderate *fairness mode with a greedy head. The convergence time is 8ms (steady-state is obtained at 1.136s) longer than for the* aggressive *fairness mode for this scenario. Full link-utilization is achieved as fast as for the* aggressive *fairness mode.*

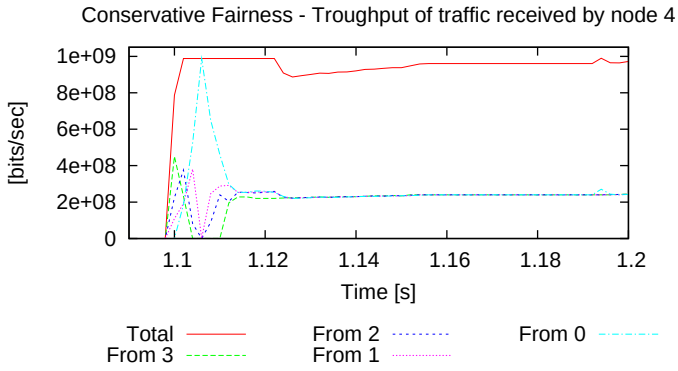



*Figure 10:* Conservative *fairness mode with a greedy head. The convergence time for the* conservative *fairness mode is much longer than for the other two fairness modes. Additionally, once steady state is achieved (shown in Fig. 11), we incur some reduced link-utilization because of the periodic event of an empty* STQ *buffer in the head.*

convergence time of 28 ms is the shortest (stable state achieved at time 1.128s) of the three fairness modes. Full link-utilization is achieved almost immediately, as the head will utilize any available bandwidth over the congested link, not in use by transit traffic.

The throughput convergence for the *moderate* fairness mode for the same scenario and using the same configuration is shown in Fig. 9. The convergence time here is somewhat higher. Steady state is achieved at time 1.136s, after 36ms. Full link-utilization however, is achieved immediately (once the sum of transit and add traffic is sufficient to fill the output link).

For the *conservative* fairness mode, the throughput convergence for this scenario is much slower. As seen from Figures 9 and 10, at the time where the *moderate* fairness mode has reached steady-state (1.136s), the *conservative* fairness mode has only reached 91% link-utilization. As seen in Fig. 11, full throughput for the *conservative* fairness mode is not achieved until 1.222s. Furthermore, as seen from Fig. 11, the *conservative* fairness mode suffers from sustained oscillations in total throughput, leading to a minor reduction in link-

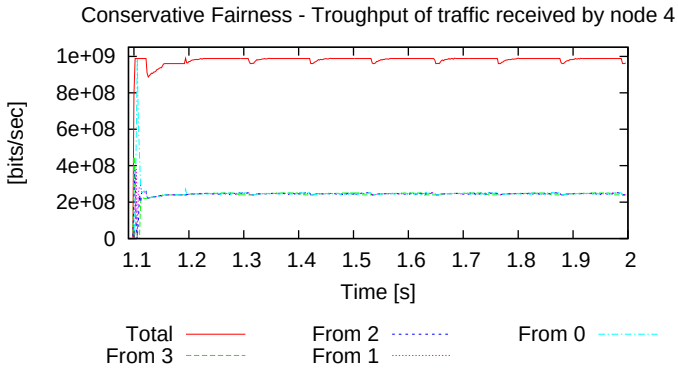



*Figure 11:* Conservative *fairness mode with a greedy head. The plot shows the reduced link-utilization (reduction in aggregate/total throughput) caused by the periodic under-run of the head's* STQ *buffer.*

utilization.

In addition to the throughput convergence and link-utilization characteristics of the three fairness modes, we can make another observation on the throughput performance of the three fairness modes. In the *aggressive* fairness mode, when the head is greedy, the *STQ* occupancy oscillates around the *high* threshold. The RPR scheduling rules block local low-priority traffic whenever the *STQ* occupancy reaches this threshold. This causes the throughput of the head to fall somewhat below the throughput of its upstream greedy neighbors. For the *moderate* and *conservative* modes, where the *STQ* occupancy oscillates around the *low* threshold (*moderate* mode) or between empty and the *medium* threshold (*conservative* mode), this is not the case. In this case, the head will achieve its (full) fair share of the available bandwidth. This effect can be seen when comparing figures 12, 13 and 14.

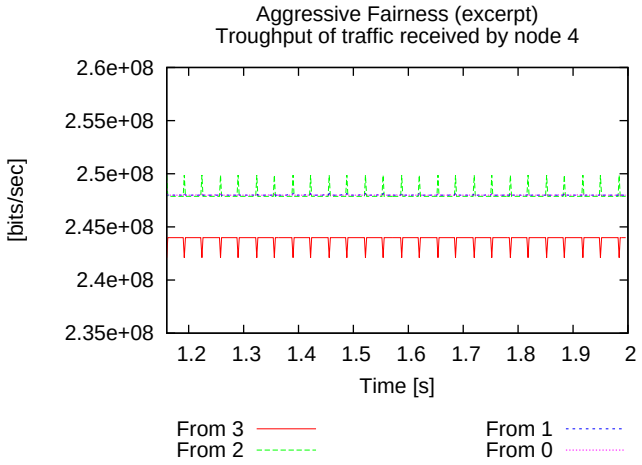



*Figure 12: Excerpt of* aggressive *fairness mode throughput with a greedy head. As seen in the figure, the traffic from the head (node 3) does not get its full fair share of bandwidth over the congested link. This is caused by the blocking of local traffic once the* STQ *occupancy exceeds the* high *threshold.*

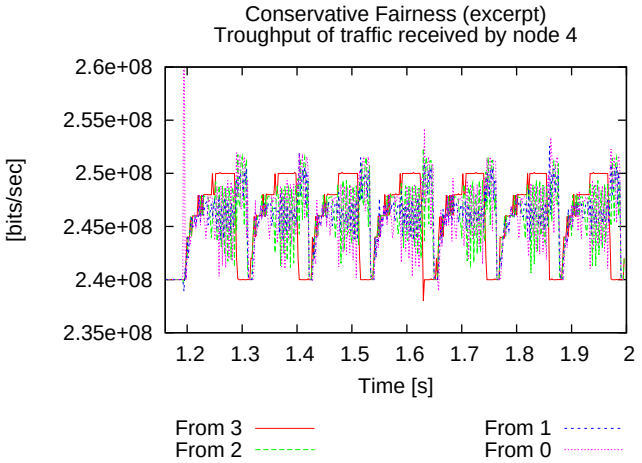



*Figure 13: Excerpt of* conservative *fairness mode throughput with a greedy head. As seen in the figure, the throughput of traffic from the head is not penalized. This is because in steady-state, the* STQ *occupancy oscillates varies between 0 and the* medium *threshold.*

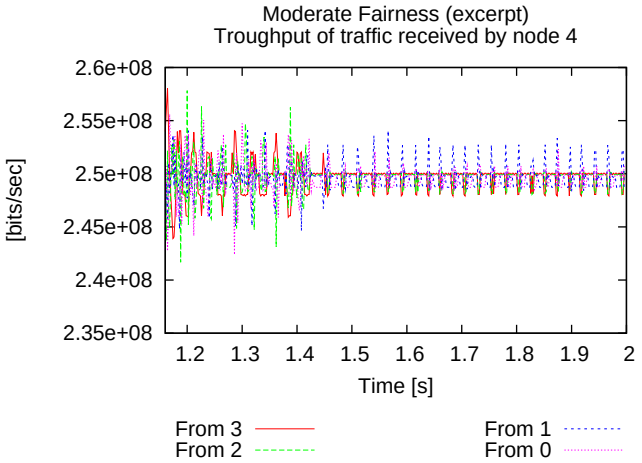



*Figure 14: Excerpt of* moderate *fairness mode throughput with a greedy head. As seen in the figure, the throughput of traffic from the head is not penalized. This is because in steady-state, the* STQ *occupancy oscillates around the* low *threshold. The variations in throughput may seem large, but are within $\pm 2$[packets/averaging interval] of the mean (the fair rate). The averaging period is 2ms.*

*B. Throughput Convergence for an Unbalanced Traffic Scenario*

In this test, we use the configuration shown in Table II. Notice that the head is configured as a modest sender, sending CBR traffic at 5% of the line-rate.

| **Parameter Name** | **Value** |
|---|---|
| Line Rate | 1 [Gbit/s] |
| Packet size | 500 [B] (fixed) |
| Head's Sending Behavior | CBR traffic at 5% of the line rate |
| All other active nodes (in congestion domain) sending behavior | GREEDY |
| STQ Thresholds<br>- low<br>- high | <br>31812 [bytes]<br>120000 [bytes] |
| rampUpCoef | 64 |
| rampDnCoef | 64 |
| ageCoef | 4 |
| lpCoef | 64 |
| link-delay | 410 $[\mu s]$ |
| Start of traffic | 1.1s |

*Table II: Configuration for comparison of the three fairness modes in the presence of a modest head.*

For this configuration, as expected (discussed in section III), the throughput convergence of the *aggressive* fairness mode (shown in Fig. 15) does not converge to the fair division of sending rates. Furthermore, the magnitude of the induced oscillations results in reduced link-utilization (i.e. the total throughput falls below the maximum obtainable).

The throughput convergence for the *conservative* fairness mode is shown in Fig. 16. As seen from the figure, it takes approximately 72ms just to reach a level of 90% of the available throughput on the congested link. Furthermore, the *conservative* mode of operation causes the *STQ* occupancy to fall to 0 at periodic intervals (just as for the *aggressive* mode). For the *conservative* fairness mode, this leads to a reduction in link-utilization for the congested link. For this particular scenario, the average reduction in link-utilization is $\approx 0.5\%$. Given the small magnitude of the *conservative* fairness mode's throughput-loss during steady-state, this can be neglected for most practical cases.

Finally, in Fig. 17, we show the throughput convergence of our *moderate* fairness mode. As seen from the figure, the convergence time is 52ms. During the transient phase, as the rate control algorithm is working on improving the size and position of the rate interval $\langle mRateMin, mRateMax \rangle$ (discussed in section VI-A), we have some brief periods where the *STQ* becomes empty, thus resulting in a temporary reduction in total throughput. Because of the low demand of the head, there are no local packets available for transmission every time the *STQ* becomes empty. Once steady-state is obtained however, the *STQ* does not become empty, and we do no longer incur any reductions in the link utilization. Furthermore, as seen from the figure, we have a fair sharing of the capacity of the congested link.

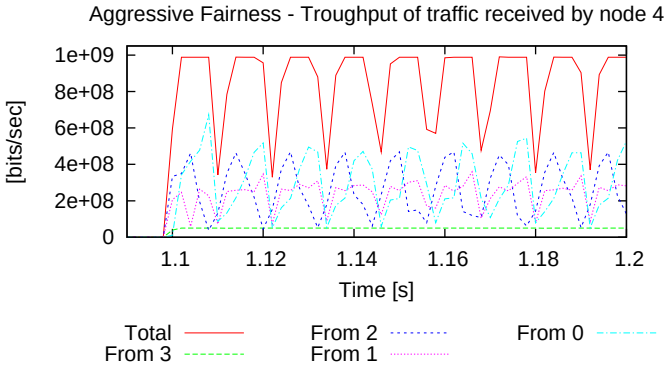



*Figure 15:* Aggressive *mode throughput convergence when head is modest. As seen, the* aggressive *fairness mode does not converge for this scenario. Additionally, the link utilization (as seen in the reduction on total/aggregate throughput) is reduced.*

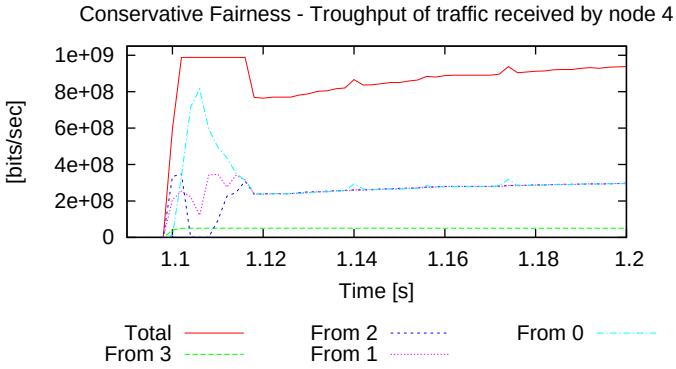



*Figure 16:* Conservative *mode throughput convergence when head is modest. As seen, the* conservative *fairness mode converges slowly. At time 1.172s, the total throughput has only reached 90% of full link-utilization.*

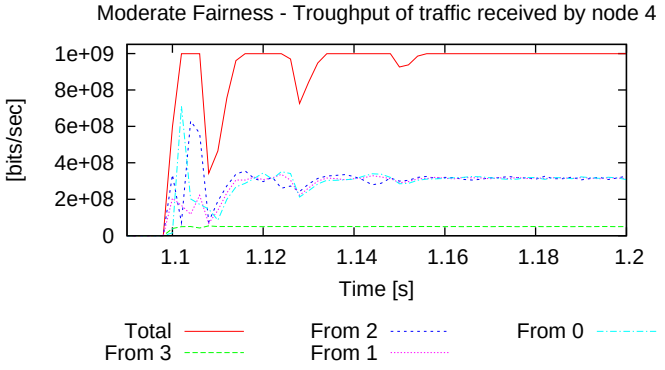



*Figure 17:* Moderate *mode throughput convergence when head is modest. The* moderate *fairness mode converges within 52ms and after this point, there is no loss in link-utilization.*

### C. Convergence for a Dynamic Traffic Scenario

In this test, we use the scenario shown in Fig. 18. The per node configuration is shown in Table III. The purpose of this test is to show that our *moderate* fairness mode is able to adapt to changing load conditions. Thus during the simulation run, we start by, at time 1.4s, decreasing the load, before we, at time 1.6s, increase the load.

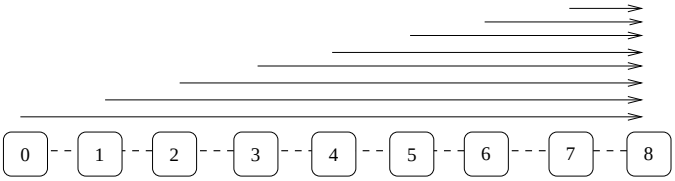



*Figure 18: A congestion domain consisting of various active nodes. All active nodes send to node 8. Thus the link between nodes 7 and 8 is the most congested link in the domain.*

| Parameter Name | Value |
|---|---|
| Line Rate | 1 [Gbit/s] |
| Packet size | 500 [B] (fixed) |
| Head's Sending Behavior | CBR traffic at 5% of the line rate |
| All other active nodes (in congestion domain) sending behavior | GREEDY |
| STQ Thresholds<br>- low<br>- high | <br>31812 [bytes]<br>120000 [bytes] |
| rampUpCoef | 64 |
| rampDnCoef | 64 |
| ageCoef | 4 |
| lpCoef | 128 |
| link-delay | 410 $[\mu s]$ |
| Start of traffic | Per node settings specified below |

*Table III: Configuration for dynamic traffic scenario.*

The resulting throughput for the various nodes is shown in Fig. 19. As seen from the figure, our algorithm converges nicely as the number of active senders increases as well as when the number of active senders decreases.

When the load decreases, the rate interval specified by the fairness mode variables $\langle mRateMin, mRateMax \rangle$ (discussed in section VI-A) is located too low. Thus, as a result of this and the modest demand of the head, the *STQ* will become empty during the transient phase, where the fairness mode works to shift the position of the rate interval higher. Thus at time 1.4s, the link-utilization is temporarily reduced (seen by the dip in total throughput), before reaching its maximum value again, following the convergence of the fairness algorithm.

Similarly, when the load is increased at time 1.6s, the rate interval specified by the fairness mode variables $\langle mRateMin, mRateMax \rangle$ is located too high. Thus, as a result the *STQ* occupancy will exceed the *high* threshold. Thus, local traffic from the head, will be blocked during portions of the transient phase, where the fairness mode works to shift the position of the rate interval lower. We also notice that following this, the STQ occupancy falls briefly to 0 (seen by the short reduction in aggregate throughput). Notice that at this

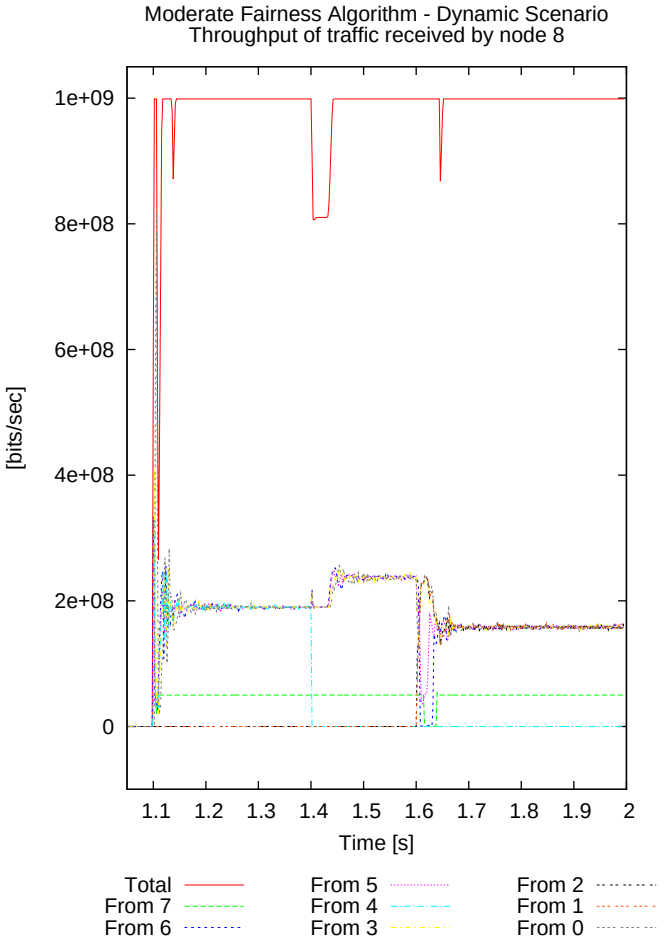



*Figure 19: The figure shows the convergence for the* moderate *fairness mode for a dynamic traffic scenario. In this scenario, nodes 0, 3, 4, 5, 6 and 7 transmit traffic from time 1.1s. At time 1.4s, node 4 stops transmitting traffic. Finally, at time 1.6s, nodes 1 and 2 start to transmit traffic. All nodes operate as greedy senders, expect for node 7, the head, which transmits traffic at 5% of the link-rate.*

time (1.6s), although the load change is larger (2 additional nodes start sending) than at the previous time (1.4s) (1 node stopped transmitting), the convergence time is approximately equal to that of the previous load change.

### D. Convergence Time as a Function of Aggregate Propagation Delay

In this section, we want to compare the performance of our *moderate* fairness mode to that of the *aggressive* fairness mode as the size of the network (congestion domain) to be controlled increases. We use the scenario shown in Fig. 1. In this experiment, for each allowed value of the *lpCoef* parameter, we run a series of simulations. For each simulation run, we increase the network size. That is, we increase the per link propagation delay, thus increasing the time-constant, $\tau_{system}$, of the congestion domain. After the simulation has been executed, we post-process the throughput data obtained, to determine the convergence time of the fairness mode used. A non-converging simulation-execution is represented by an infinite value of the convergence time.

| **Parameter Name** | **Value** |
|---|---|
| Line Rate ($unreservedRate$) | 1 [Gbit/s] |
| Packet size | 500 [B] (fixed) |
| Upstream Nodes Sending Behavior | GREEDY |
| STQ Thresholds<br>- low<br>- high | <br>30000 [bytes], fixed size STQ<br>120000 [bytes], fixed size STQ<br>otherwise given by (11) and (12) |
| rampUpCoef | 64 |
| rampDnCoef | 64 |
| ageCoef | 4 |
| lpCoef | $\{16, 32, 64, 128, 256, 512\}$ |
| link-delay | free variable<br>(all links are of equal length) |
| Start of traffic | 1.1s |

*Table IV: Configuration for scenario where we investigate the relation between convergence of fairness mode and the aggregate propagation delay of the congestion domain.*

The experiment is repeated for various load-configurations of the head (greedy and modest) for the two fairness modes as well as using a fixed *STQ* size and a *STQ* size that is a function of the aggregate propagation delay.

The configuration of our experiment is summarized in Table IV and Fig. 20. The results are shown in Fig. 21.

For fast convergence of the fairness mode, let us assume that the region between the *STQ* thresholds *low* and *high* must be able to buffer data for a period given by the system time constant – $\tau_{system}$. For simplicity, we express $\tau_{system}$ in terms of the round-trip propagation delay of the congestion domain to be controlled as shown in (9) (disregarding transit path and transmission delays).

$$\tau_{system} \approx (|congestionDomain| - 1) \cdot 2 \cdot t_{lp} \tag{9}$$

In the formula, $|congestionDomain|$ is the size (number of nodes) in the congestion domain and $t_{lp}$ is the length (propagation delay) of a single link in the network (all links are of equal length).

Thus for a period of duration $\tau_{system}$, the head should be able to buffer an amount of transit traffic equal to the maximum amount of local traffic added during the same period. When receiving transit data at the maximum allowed rate, *unreservedRate*, as long at the *STQ* occupancy stays below the *high* threshold, the add rate of the head is limited upwards by the RPR scheduling rules to $addRate \leq \frac{unreservedRate}{2}$.

Thus for the configuration of the *STQ* size and associated thresholds, we get the formula as follows:

$$(high - low) \geq \frac{unreservedRate}{2} \cdot \tau_{system}\ [bits] \tag{10}$$

Furthermore, if we use the RPR standard's default value of the *low* threshold and convert to units of bytes:

$$low = \frac{high}{4} \tag{11}$$

We get:

$$\begin{aligned} high - \frac{high}{4} &\geq \frac{unreservedRate}{16} \cdot \tau_{system}\ [bytes] \\ \Rightarrow high &\geq \frac{unreservedRate}{12} \cdot \tau_{system}\ [bytes] \end{aligned} \tag{12}$$

For the *aggressive* fairness mode, we know that the fairness mode does not converge when the head has a modest sending behavior (represented by the cross in Fig. 21), thus for this mode, we present results for a greedy head sending behavior only.

If we consider the results for the *aggressive* fairness mode, we find that larger size of the *STQ* does not translate directly[6] to the stable operation of larger networks. This is illustrated by Fig. 21 a) and b). Fig. 21 a) shows the convergence result for the scenario, where the size (and associated thresholds) of the *STQ* is a function of the round-trip propagation delay (i.e. $\tau_{system}$) of the congestion domain.

Fig. 21 b) on the other hand, shows the convergence result for the scenario, where the size (and associated thresholds) of the *STQ* is kept fixed according to the settings shown in Table IV. When we compare the two figures, we see that the experiment where we use a fixed *STQ* size, the fairness algorithm converges for networks of a larger size than for the experiment where the *STQ* size is a function of $\tau_{system}$.

The reason for this is that an increased size of the *STQ* buffers (i.e. when the *STQ* size is a function of $\tau_{system}$) contributes to increasing the queueing delay in the transit path of the congestion domain. This leads to an earlier onset (i.e. for networks of smaller size) of the problem described in [15], where node 1 will periodically assume the role as tail of the congestion domain. During the periods where node 1 operates as tail of the congestion domain, node 0 is allowed to transmit traffic at excessive rates, prohibiting convergence of the fairness algorithm.

[6]This does not mean the the *STQ* buffer can be arbitrarily small.

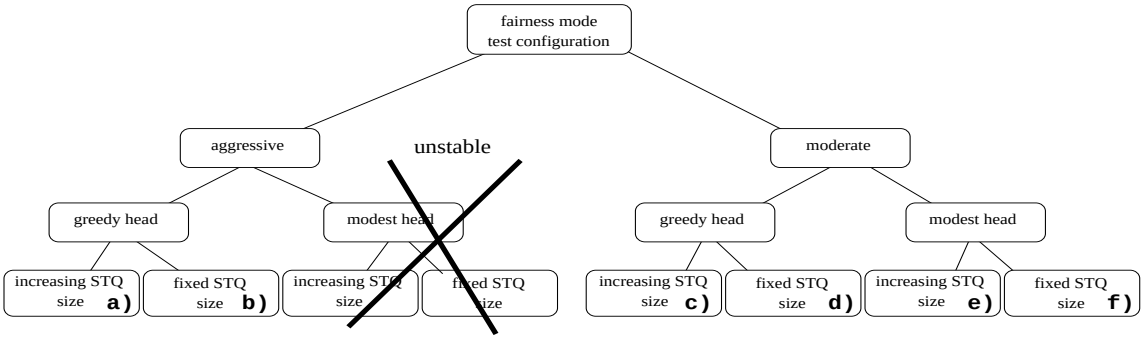



*Figure 20: Test configuration overview for section "Convergence Time as a Function of Aggregate Propagation Delay". Notice the letters a-f in the lower right-hand corner of the leaf-nodes in the tree. These point to the corresponding sub-figure number in Fig. 21 on the next page, showing the simulation results for this experiment. The part of the tree with a cross over, corresponds to so called unbalanced traffic scenarios discussed in section III. For these scenarios, the* aggressive *fairness mode does not converge, so we do not show any results for these tests.*

If we consider the *moderate* fairness mode (Fig. 21 c-f), we see that it is of importance that the *STQ* and associated thresholds are set large enough. The reason for this, is that the observation[7] of local optima (i.e. changes in growth direction) of the *STQ* occupancy, should be done without the *STQ* becoming empty or exceeding the *high* threshold. While this is OK in the initial phase (it may actually speed up the convergence in many cases) of the convergence process, the process will not converge if the *STQ* thresholds are set so that these thresholds are encountered at every cycle of the convergence process. Equation (12) provides a conservative estimate of the required threshold setting of the *STQ*. The fixed value of the *STQ* $high$ threshold used for experiments d) and f) translates to a limitation of $120 \cdot 10^3 \geq \frac{10^9}{12} \cdot \tau_{system} \Rightarrow \tau_{system} \leq 1.44ms$ for the system time-constant. For the *moderate* fairness mode and a head sending at 5% of the line-rate (Fig. 21 f)), we see that this *STQ* setting provides adequate performance for a network of a size $\leq 10$ times this. At this point (i.e. when $\tau_{system} \approx 14000$) however, for $lpCoef = 512$, we see that although the fairness mode converges, it takes consideratebly more time than if we increase the sizing of the *STQ* (shown in Fig. 21 e)).

Another property worth noting for the *moderate* fairness mode, is that convergence may take longer if the head is modest than if the head is greedy. This is in large part due to the fact that when the head is greedy, the head will be able to observe the resulting effect of a rate change much faster than when the head is modest. In particular, a change in *STQ* occupancy in response to the head's change in locally transmitted traffic may be observable almost immediately.

Finally, if we compare the performance of the *aggressive* fairness mode to that of the *moderate* fairness mode, we see that convergence for the *moderate* fairness mode takes somewhat longer for the greedy head scenarios. However, provided sufficiently sized and configured *STQ* buffers, the *moderate* fairness mode provides stable operation for a larger range of network sizes, as well as for unbalanced traffic (i.e. modest head) scenarios. While the operation of the *aggressive* mode fairness mode can be modified to support networks of larger size (e.g. by using our modification proposed in [15]), we are not aware of any modifications providing stable operation in unbalanced traffic scenarios.

In concluding this section, it appears that the *moderate* fairness mode provides a stability property where, the convergence of the fairness mode is a function of the network size and setting of the lpCoef parameter. We have previously demonstrated this property for the *aggressive* fairness mode [8].

[7]Remember that it is the finding of the local optima of the *STQ* occupancy that enables the fairness mode to improve the size and position of the dynamic rate range used to limit the adjustment of the *moderate* mode fair rate estimate, $lpMRate$.

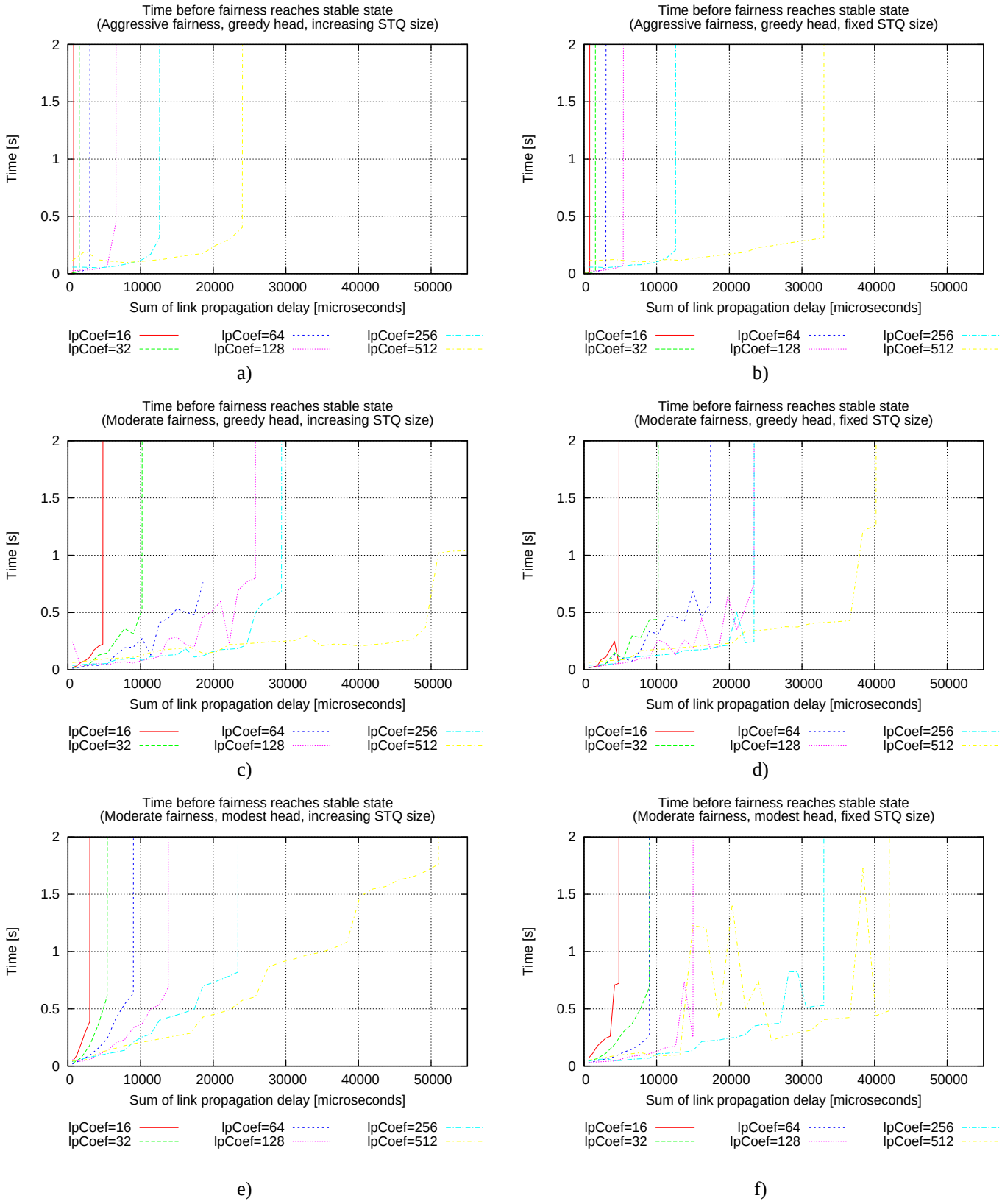



*Figure 21: Throughput convergence for RPR* Aggressive *and* Moderate *fairness modes.*

## VIII. Related Work

Other groups having looked into the problems of sustained large oscillations and resulting loss of link-utilization for unbalanced traffic scenarios, have mostly solved the problem by proposing alternate fairness algorithms that would have to replace the current RPR *aggressive* mode fairness algorithm [9], [10]. Some work have presented results that seems to fit within the framework given by the RPR standard [11]. In that work however, the effect of the proposed modifications is to reduce the symptoms (oscillations) rather than removing them.

## IX. Conclusion

In this paper we have proposed two novel contributions. The first contribution is a context determination function, to enable the head of congestion domain to decide whether it is using its fair share of bandwidth over the congested link. The method is based on local information only and does not require knowledge on the number of nodes sending traffic over the congestion point. The second contribution is a novel fairness mode, which we have called the *moderate* fairness mode. The *moderate* fairness mode can be regarded as a generalized version of the *aggressive* fairness mode, where we by use of an explicit, self-adjusting rate interval and the RPR 2-stage low-pass filter construct, control the convergence of the (*moderate*) fair rate calculation process. By use of this fairness mode, the problems reported for so called unbalanced load-scenarios is avoided and we do not incur the large throughput oscillations and resulting reduced link-utilization.

For the tested scenarios, our *moderate* fairness mode outperforms[8] the *conservative* fairness mode in terms of convergence time and link-utilization. For the *moderate* fairness mode, regardless of the sending behavior of the head, the average steady-state *STQ* occupancy in the head equals the *low* threshold. Thus providing a relatively low transit delay for all traffic classes. This is clearly better than for the *aggressive* fairness mode, where for a greedy head, the average *STQ* occupancy equals the *high* threshold.

In the *moderate* fairness mode however, the head cannot compensate for errors in its fair rate estimates purely by the increase or throttling of local traffic. Thus, for some scenarios, the convergence time may be somewhat longer than that of the *aggressive* fairness mode.

As seen in sections VII-A-VII-D, our *moderate* fairness mode converges to the fair division of sending rates, as well as maintaining full link-utilization for a broad range of tested scenarios, thus we claim conformance to DO1 and DO2.

There is no clear definition of a "minimized set of changes", neither is it clear that it makes sense to try to formulate such a definition. Thus conformance to DO3 can be argued. We have however, by use of existing statistics data, introduction of a set of new state variables and a new state machine (for the fair rate calculation) introduced a new fairness mode. Thus, we claim conformance to DO3 and DO4.

[8]Once in steady-state, the link-utilization of the *conservative* mode is only slightly lower than that of the *moderate* fairness mode.

Furthermore, in section VII-A, we have demonstrated that the use of the *moderate* fairness mode in the head, interoperates well with the use of the *aggressive* fairness mode in upstream nodes. We have however not tested interoperability with the *conservative* fairness mode, thus we can only claim partial conformance to DO5.

In conclusion, we will claim that our Design Objectives (DOs) presented in section IV have been partially fulfilled.

## X. Further Work

In further work, it would be interesting to test our *moderate* fairness mode for an even broader set of test scenarios.

## References


[1] IEEE Computer Society, "IEEE Std 802.17-2004," September 24 2004.

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

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

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

[5] C. C. Reames and M. T. Liu, "A loop network for simultaneous transmission of variable-length messages," in *Proceedings of the 2nd Annual Symposium on Computer Architecture*, vol. 3, December 1974.

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

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

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

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

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

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

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

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

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

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

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

[17] H. Kong, N. Ge, F. Ruan, and C. Feng, "Congestion control algorithm for Resilient Packet Ring," *Tsinghua Science and Technology*, vol. 8, no. 2, April 2003.