High Performance Fair Bandwidth Allocation for Resilient Packet Rings *

V. Gambiroza, Y. Liu, P. Yuan, and E. Knightly
ECE Department, Rice University
Houston, TX 77005, USA
email:{violeta, yhliu, pyuan, knightly }@rice.edu

Abstract

The Resilient Packet Ring (RPR) IEEE 802.17 standard is under development as a new high-speed backbone technology for metropolitan area networks. A key performance objective of RPR is to simultaneously achieve high utilization, spatial reuse, and fairness, an objective not achieved by current technologies such as SONET and Gigabit Ethernet nor by legacy ring technologies such as FDDI. The core technical challenge for RPR is the design of a bandwidth allocation algorithm that dynamically achieves these properties. The difficulty is in the distributed nature of the problem, that upstream ring nodes must inject traffic at a rate according to congestion and fairness criteria downstream. Unfortunately, the proposed algorithms in the current draft standards have a number of critical limitations. For example, we show that in a two-flow two-link scenario with unbalanced and constant-rate traffic demand, a draft RPR algorithm will suffer from dramatic bandwidth oscillations within nearly the entire range of the link capacity. Moreover, such oscillations hinder spatial reuse and decrease throughput significantly. In this paper, we introduce a new dynamic bandwidth allocation algorithm called Distributed Virtual-time Scheduling in Rings (DVSR). The key idea is for nodes to compute a simple lower bound of temporally and spatially aggregated virtual time using per-ingress counters of packet (byte) arrivals. We show that with this information propagated along the ring, each node can remotely approximate the ideal fair rate for its own traffic at each downstream link. Hence, DVSR flows rapidly converge to their ring-wide fair rates while maximizing spatial reuse. To evaluate DVSR, we bound the deviation in service between DVSR and an idealized reference model, thereby bounding the unfairness. With simulations, we find that compared to current techniques, DVSR's convergence times are an order of magnitude faster (e.g., 2 vs. 50 msec), oscillations are mitigated (e.g., ranges of 1% vs. up to 100%), and nearly complete spatial reuse is achieved (e.g., 0.1% throughput loss vs. 14%).

1 Introduction

Current technology choices for high-speed metropolitan ring networks provide a number of unsatisfactory alternatives. A SONET ring can ensure minimum bandwidths (and hence fairness) between any pair of nodes. However, use of circuits prohibits unused bandwidth from being reclaimed by other flows and results in low utilization. On the other hand, a Gigabit Ethernet (GigE) ring can provide full statistical multiplexing and high utilization, but suffers from unfairness. For example, in the topology of Figure 1, nodes will obtain different throughputs to the core or hub node depending on their spatial location on the ring. Finally, legacy ring technologies such as FDDI do not employ spatial reuse. That is, by using a rotating token such that a node must have the token to transmit, only one node can transmit at a time.[1]

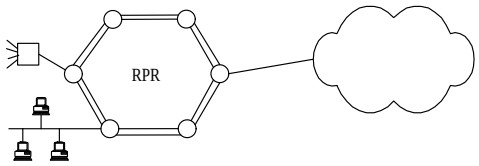


Figure 1: Illustration of Resilient Packet Ring

The IEEE 802.17 Resilient Packet Ring (RPR) working group was formed in early 2000 to develop a standard for bi-directional packet metropolitan rings. Unlike FDDI, the protocol supports destination packet removal so that a packet will not traverse all ring nodes and spatial reuse can be achieved. However, allowing spatial reuse introduces a challenge to ensure fairness among different nodes competing for ring bandwidth. Consequently, the key performance objective of RPR is to simultaneously achieve high utilization, spatial reuse, and fairness.[2]

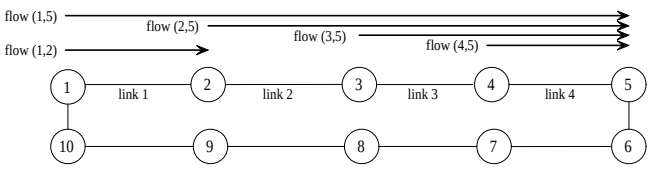


Figure 2: Parallel Parking Lot

To illustrate spatial reuse and fairness, consider the depicted scenario in Figure 2 in which four infinite demand flows share link 4 in route to destination node 5. In this

*This research is supported by NSF Grants ANI-0099148 and ANI-0085842, and by a Sloan Fellowship.

[1] For the reader asking "why a ring?", we note that rings are the most prevalent metro technology primarily for their protection and fault tolerance properties.

[2] Additional RPR goals beyond the scope of this paper include fast fault recovery similar to that of SONET.

"parallel parking lot" example, each of these flows should receive 1/4 of the link bandwidth. Moreover, to fully exploit spatial reuse, flow (1,2) should receive all excess capacity on link 1, which is 3/4 due to the downstream congestion.

The key technical challenge of RPR is design of a bandwidth allocation algorithm that can dynamically achieve such rates. Note that to realize this goal, some coordination among nodes is required. For example, if each node performs weighted fair queueing [18], a local operation without coordination among nodes, flows (1,2) and (1,5) would obtain equal bandwidth shares at node 1 so that flow (1,2) would receive a net bandwidth of 1/2 vs. the desired 3/4. Thus, RPR algorithms must throttle traffic at ingress points based on downstream traffic conditions to achieve these rate allocations.

Two draft proposals, Gandalf [6] and Aladdin [5], have been under development and debate since the inception of RPR. Additionally, a third consensus proposal termed Darwin [7] has been recently adopted as a compromise, incorporating most of its algorithmic properties from Gandalf as well as others from Aladdin. Unfortunately, we have found that the proposed algorithms have a number of important performance limitations. First, they are prone to severe and permanent oscillations in the range of the entire link bandwidth in simple "unbalanced traffic" scenarios with unbalanced input traffic rates, or unbalanced congestion on bottleneck links. Second, they are not able to fully achieve spatial reuse and fairness. Third, they require numerous control signal updates to converge to their fair bandwidths thereby hindering fast responsiveness.

In this paper, we develop a new dynamic bandwidth allocation algorithm termed Distributed Virtual-time Scheduling in Rings (DVSR). Like current implementations, DVSR has a simple FIFO (or Strict Priority) transit path. However, with DVSR, each node uses its per-destination byte counters to construct a simple lower bound on the evolution of the spatially and temporally aggregated virtual time. That is, using measurements available at an RPR node, we compute the minimum cumulative change in virtual time since the receipt of the last control message, as if the node were performing weighted fair queueing at the granularity of ingress-aggregated traffic. By circulating such control information throughout the ring, we show how nodes can perform simple operations on the collected information and throttle their ingress flows to their ideal fair rates as defined in the RIAS reference model of [12].

Next, we study the performance of DVSR and RPR algorithms in general using a combination of theoretical analysis and simulation. In particular, we analytically bound DVSR's unfairness due to use of delayed and time-averaged information in the control signal. We perform *ns-2* and OPNET simulations to compare RPR algorithms and obtain insights into problematic scenarios and sources of poor algorithm performance. For example, we show that while DVSR can fully reclaim unused bandwidth in scenarios with unbalanced traffic, Gandalf suffers from permanent oscillation and significant utilization losses. We also show how DVSR's fairness mechanism can provide performance isolation among nodes' throughputs. For example, in a Parking Lot scenario (Figure 9) with even moderately aggregated TCP flows from one node competing for bandwidth with non-responsive UDP flows from other nodes, all ingress nodes obtain nearly equal throughput shares with DVSR, quite different from the unfair node throughputs obtained with a GigE ring.

Finally, we have developed a 1 Gb/sec network processor implementation of DVSR. Details of the testbed and results of our measurement study on an eight-node ring can be found in reference [2].

2 Background on IEEE 802.17 RPR

In this section, we describe the objectives of the Resilient Packet Ring (RPR) standard and briefly review a proposal under discussion in the IEEE 802.17 working group, Gandalf [6] (a successor to SRP [22]).[3]

The goal of all fairness algorithms is to achieve RIAS fairness (Ring Ingress Agreggated with Spatial Reuse), a property that we defined in [2, 12]. RIAS Fairness has two key components. The first component defines the level of traffic granularity for fairness determination at a link as an ingress-aggregated (IA) flow, i.e., the aggregate of all flows originating from a given ingress node. The second component of RIAS fairness ensures maximal spatial reuse subject to this first constraint. That is, bandwidth can be reclaimed by IA flows when it is unused either due to lack of demand or in cases of sufficient demand in which flows are bottlenecked elsewhere. Thus, RIAS allocations are quite different than max-min fairness and proportional fairness [3, 11, 15].

2.1 RPR Node Architecture

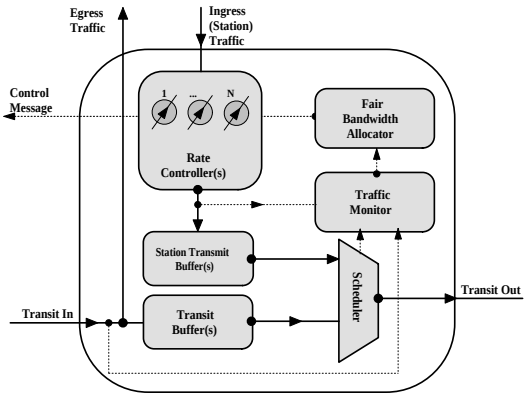


Figure 3: Generic RPR Node Architecture

The architecture of a generic RPR node is illustrated in Figure 3. First, observe that all station traffic entering the ring is first throttled by rate controllers.[4] In the example of

[3]The Darwin proposal [7], introduced in January 2002, uses most mechanisms from the Gandalf algorithm with additional features from the Aladdin algorithm [5]. We compare DVSR with Aladdin in [2].

[4]In RPR terminology, "station" traffic is referred to as "transmit" traffic; we use the former term to more easily differentiate from "transit" traffic.

the Parallel Parking Lot, it is clear that to fully achieve spatial reuse, flow (1,5) must be throttled to rate 1/4 at its ring ingress point. Second, these rate controllers are at a per-destination granularity. This allows a type of virtual output queueing analogous to that performed in switches to avoid head-of-line blocking, i.e., if a single link is congested, an ingress node should only throttle its traffic forwarded over that link. Next, RPR nodes have measurement modules (byte counters) to measure demanded and/or serviced station and transit traffic. These measurements are used by the fairness algorithm to compute a feedback control signal to throttle upstream nodes to the desired rates. Nodes that receive the control message use the information in the message, perhaps together with local information, to set the bandwidths for the rate controllers. The final component is the scheduling algorithm that arbitrates service among station and transit traffic. This scheduler is typically FIFO with buffer thresholds or Strict Priority (SP) of transit traffic over station traffic. Both ensure hardware simplicity and SP ensures that the transit path is lossless, i.e., once a packet is injected into the ring, it will not be dropped at a downstream node.

The dynamic bandwidth control algorithm that determines the station rate controller values, and hence the basic fairness and spatial reuse properties of the system is the key component of RPR protocols and is the focus of the discussion below as well as throughout the paper.[5]

2.2 Gandalf

There are two key measurements for Gandalf's bandwidth control, *forward_rate* and *my_rate*. The former represents the service rate of all transit traffic and the latter represents the rate of all serviced station traffic [6].[6] In both cases, it is important that the rates are measured at the output of the scheduler so that they represent serviced rates rather than offered rates.

A node n is said to be congested whenever *forward_rate*[n] + *my_rate*[n] $>$ $low\_threshold$, where $low\_threshold$ is a fixed value less than the link capacity. When a node is congested, it transmits a control message to upstream nodes containing the normalized service rate of its own station traffic, *my_rate*[n]. When an upstream node i receives a congestion message advertising *my_rate*[n], it reduces its rate limiter values, termed *allowed_rate*[i][j], for all values of j on the path from i to n. The objective is to have upstream nodes throttle their own station rate controller values to their minimum received value of the collected *my_rate*[n] values along the path to the destination. Thus, station traffic rates will not exceed the advertised *my_rate* value of any node in the downstream path of a flow. If upstream node $n-1$ receiving the congestion message from node n is also congested, it will propagate the message upstream using the minimum of the node's newly computed *allowed_rate* and its own measured *my_rate*[n-1]. If node $n-1$ is not congested, it sets *my_rate*[n-1] to a null value to indicate a lack of congestion. Node $n-1$ will also set a null rate if it *is* congested but *my_rate*[n-1] $>$ *forward_rate*[n-1], as this situation indicates that the congestion is due to node $(n-1)$'s station traffic vs. transit traffic from further upstream.

When nodes receive null messages, they increment their values of *allowed_rate* by a fixed value. This allows the upstream node to reclaim additional bandwidth if one of the downstream flows reduces its rate. Moreover, such rate increases are essential to convergence to fair rates even in cases of static demand. Considering the above parking lot example, if a downstream node advertises *my_rate* below the true fair rate (which does indeed occur before convergence), all upstream nodes will throttle to this lower rate; in this case, the downstream nodes will later become "uncongested" so that flows will increase their *allowed_rate*. This process will then oscillate more and more closely around the targeted fair rates for this example. To dampen the system's oscillations during convergence, measurement and control variables such as *my_rate* and *allowed_rate* are necessarily low pass filtered.

The logic of the algorithm is that if all nodes sharing the bottleneck send at the rate of the minimum received *my_rate*, then flow rates are equalized and fairness is achieved. Without congestion, flows should periodically increase their *allowed_rate* to ensure they are receiving their maximal bandwidth share.

Finally, in Gandalf and Darwin, there is an option for each node to have only a single rate controller instead of per-destination rate controllers as depicted in Figure 3. Note that use of a single rate controller precludes achieving maximal spatial reuse. Returning to the example of Figure 2, flows (1,5) and (1,2) require *allowed_rate*[1][5] = 1/4 and *allowed_rate*[1][2] = 3/4 to achieve maximum spatial reuse. However, if only a single value *allowed_rate*[1] is used, this value must be 1/4. Hence, flows (1,5) and (1,2) would each obtain throughput 1/8 such that the *total* bandwidth for traffic originating from node 1 is 1/4.

2.3 Discussion

There are a number of important limitations to the current IEEE 802.17 draft algorithms which we discuss below and explore using simulations in Section 5. First, the algorithms suffer from severe and permanent oscillations for scenarios with unbalanced traffic. For example, consider the Gandalf algorithm with two nodes and two flows such that flow (1,3) originating upstream has demand for the full link capacity C, and flow (2,3) originating downstream has a low rate which we denote by ϵ. Hence the demands are constant rate and unbalanced. Since the traffic arrival rate downstream is $C+\epsilon$, the downstream link will become congested. Thus, a congestion message will arrive upstream containing the transmission rate of the downstream flow, in this case ϵ. Consequently, the upstream node must throttle its flow from rate C to rate ϵ. At this point, the rate on the downstream link is 2ϵ so that congestion clears, and upon

[5] We focus attention on the committed rate class in which each node obtains a minimum bandwidth share and reclaims unused bandwidth in a weighted fair manner. RPR also defines a guaranteed class in which bandwidth cannot be reclaimed (and hence no such control algorithm is required), and a best effort class that is a special case of committed with a minimum rate of 0.

[6] All rates actually represent byte counts over a fixed interval length.

receiving null congestion messages, the upstream flow increases its rate back to C. Repeating the cycle, the upstream flow's rate will permanently oscillate between the full link capacity C and the low rate of the downstream flow ϵ. There are multiple effects of such oscillations, including throughput loss for these open loop flows, adverse effects and throughput loss to TCP traffic, increased delay jitter, etc. The key issue is that the congestion signal *my_rate* does not accurately reflect the congestion status or fair rate at downstream nodes and hence the system oscillates in search of the correct fair rate.

Second, Gandalf is not able to fully exploit spatial reuse. For Gandalf and the above example, the upstream flow's rate should be $C - \epsilon$ to achieve full spatial reuse. Instead, the rate is oscillating between C and ϵ resulting in lower throughput and partial spatial reuse.

Third, the Gandalf suffers from slow convergence times. In particular, to mitigate oscillations even for constant rate traffic inputs as in the example above, all measurements are low pass filtered. However, such filtering, when combined with the coarse feedback information, has the effect of delaying convergence (for scenarios where convergence does occur).

3 Distributed Virtual time Scheduling in Rings (DVSR)

In this section, we devise a distributed algorithm to dynamically realize the bandwidth allocations in the RIAS reference model. Our key technique is to have nodes construct a proxy of virtual time at the Ingress Aggregated flow granularity. This proxy is a lower bound on virtual time temporally aggregated over time and spatially aggregated over traffic flows sharing the same ingress point (IA flows). By distributing this information to other nodes on the ring, all nodes can remotely compute their fair rates at downstream nodes, and rate control their per-destination station traffic to the RIAS fair rates.

We first describe the algorithm in an idealized setting, initially considering virtual time as computed in a GPS fluid system [18] with an IA flow granularity. We then progressively remove the impractical assumptions of the idealized setting, leading to the network-processor implementation described in [2].

As previously, we denote $r_{ij}(t)$ as the offered input rate (demanded rate) at time t from ring ingress node i to ring egress node j. Moreover let $\rho_{ij}(t)$ denote the rate of the per-destination ingress shaper for this same flow.

3.1 Distributed Fair Bandwidth Allocation

The distributed nature of the ring bandwidth allocation problem yields three fundamental issues that must be addressed in algorithm design. First, resources must be *remotely controlled* in that an upstream node must throttle its traffic according to congestion at a downstream node. Second, the algorithm must contend with *temporally aggregated and delayed control information* in that nodes are only periodically informed about remote conditions, and the received information must be a temporally aggregated summary of conditions since the previous control message. Finally, there are *multiple resources* to control with complex interactions among multi-hop flows. We next consider each issue independently.

3.1.1 Remote Fair Queueing

The first concept of DVSR is control of upstream rate-controllers via use of ingress-aggregated virtual time as a congestion message received from downstream nodes. For a single node, this can be conceptually viewed as remotely transmitting packets at the rate that they would be serviced in a GPS system, where GPS determines packet service order according to a granularity of packets' ingress nodes only (as opposed to ingress and egress nodes, micro-flows, etc.).

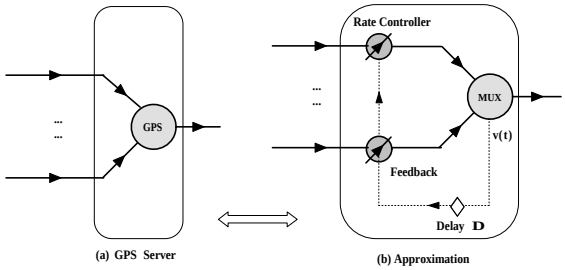


Figure 4: Illustration of Remote Fair Queueing

Figure 4 illustrates remote bandwidth control for a single resource. In this case, RIAS fairness is identical to flow max-min fairness so that GPS can serve as the ideal reference scheduler. Conceptually, consider that the depicted multiplexer (labelled "MUX" in Figure 4(b)) computes virtual time as if it is performing idealized GPS, i.e., the rate of change of virtual time is inversely proportional to the (weighted) number of backlogged flows. The system on the right approximates the service of the (left) GPS system via adaptive rate control using virtual time information. In particular, consider for the moment that the rate controllers receive continuous feedback of the multiplexer's virtual time calculation and that the delay in receipt of this information is $\Delta = 0$. The objective is then to set the rate controller values to the flows' service rates in the reference system. In the idealized setting, this can be achieved by the simple observation that the evolution of virtual time reveals the fair rates. In this case, considering a link capacity $C = 1$ and denoting virtual time as $v(t)$, the rate for flow i and hence the correct rate controller value is simply given by[7]

$$\rho_i(t) = \min(1, dv(t)/dt)$$

when $v_i(t) > 0$ and 1 otherwise.

For example, consider the four flow parking lot example of Figure 9. Suppose that the system is initially idle so that $\rho_i(0) = 1$, and that immediately after time 0, flows begin transmitting at infinite rate (i.e., they become infinitely backlogged flows). As soon as the multiplexer depicted in Figure 4(b) becomes backlogged, $v(t)$ has slope 1/4. With this value instantly fed back, all rate controllers are immediately set to ρ_i = 1/4 and flows are serviced at their fair

[7]Note that GPS has fluid service such that all flows are served at identical (or weighted) rates whenever they are backlogged.

rate.

Suppose at some later time the 4th flow shuts off so that the fair rates are now 1/3. As the 4th flow would no longer have packets (fluid) in the multiplexer, $v(t)$ will now have slope 1/3 and the rate limiters are set to 1/3. Thus, by monitoring virtual time, flows can increase their rates to reclaim unused bandwidth and decrease it as other flows increase their demand. Note that with 4 flows, the rate controllers will never be set to rates below 1/4, the minimum fair rate.

Finally, notice that in this ideal fluid system with zero feedback delay, the multiplexer is never more than infinitesimally backlogged, as the moment fluid arrives to the multiplexer, flows are throttled to a rate equal to their GPS service rates. Hence, all buffering and delay is incurred before service by the rate controllers.

3.1.2 Delayed and Temporally Aggregated Control Information

The second key component of distributed bandwidth allocation in rings is that congestion and fairness information shared among nodes is necessarily delayed and temporally aggregated. That is, in the above discussion we assumed that virtual time is continually fed back to the rate controllers without delay. However, in practice feedback information must be periodically summarized and transmitted in a message to other nodes on the ring. Thus, delayed receipt of summary information is also fundamental to a distributed algorithm.

For the same single resource example of Figure 4, and for the moment for $\Delta = 0$, consider that every T seconds the multiplexer transmits a message summarizing the evolution of virtual time over the previous T seconds. If the multiplexer is continuously backlogged in the interval $[t - T, t]$, then information can be aggregated via a simple time average. If the multiplexer is idle for part of the interval, then additional capacity is available and rate controller values may be further increased accordingly. Moreover, $v(t)$ should not be reset to 0 when the multiplexer goes idle, as we wish to track its increase over the entire window T. Thus, denoting b as the fraction of time during the previous interval T that the multiplexer is busy serving packets, the rate controller value should be

$$\rho_i(t) = \min(1, (v(t) - v(t-T))/T + (1-b)). \quad (1)$$

The example depicted in Figure 5 illustrates this time averaged feedback signal and the need to incorporate b that arises in this case (but not in the above case without time averaged information). Suppose that the link capacity is 1 packet per second and that $T = 10$ packet transmission times. If the traffic demand is such that six packets arrive from flow 1 and two packets from flow 2, then 2 flows are backlogged in the interval [0,4], 1 flow in the interval [4,8], and 0 flows in [8,10]. Thus, since b = 0.8 the rate limiter value according to Equation (1) is 0.8. Note that if both flows increase their demand from their respective rates of 0.6 and 0.2 to this maximum rate controller value of 0.8, congestion will occur and the next cycle will have $b = 1$ and fair rates of 0.5.

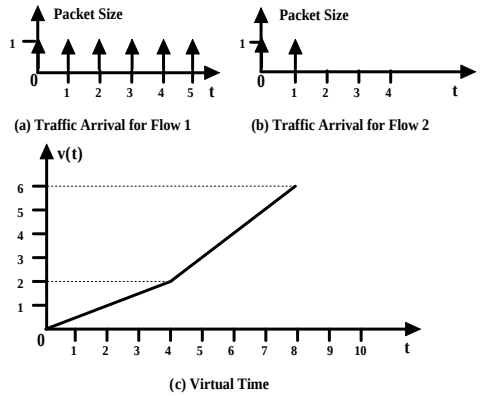


Figure 5: Temporally Aggregated Virtual Time Feedback

Finally, consider that the delay to receive information is given by $\Delta > 0$. In this case, rate controllers will be set at time t to their average fair rate for the interval $[t - T - \Delta, t - \Delta]$. Consequently, due to both delayed and time averaged information, rate controllers necessarily deviate from their ideal values, even in the single resource example. We consider such effects of Δ and T analytically in Section 4 and via simulations in Section 5.

3.1.3 Multi-node RIAS Fairness

There are three components to achieving RIAS fairness encountered in multiple node scenarios. First, an ingress node must compute its minimum fair rate for the links along its flows' paths. Thus, in the parking lot example, node 1 initially receives fair rates 1, 1/2, 1/3, and 1/4 from the respective nodes on its path and hence sets its ingress rate to 1/4.

Second, if an ingress node has multiple flows with different egress nodes sharing a link, it must sub-allocate its per-link IA fair rate to these flows. The first and second steps can be combined by setting the rate limiter value to be

$$\rho_{i,j}(t) = \min(1, \min_{i \le n < j} \rho_i^n / |P_i^n|) \quad (2)$$

where ρ_i^n is the single link fair rate at link n as given by Equation (1) and $|P_i^n|$ denotes the number of flows at link n with ingress node i.[8]

Finally, we observe that in certain cases, the process takes multiple iterations to converge, even in this still idealized setting, and hence multiple intervals T to realize the RIAS fair rates. The key reason is that nodes cannot express their true "demand" to all other nodes initially, as they may be bottlenecked elsewhere. For example, consider the scenario illustrated in Figure 6 in which all flows have infinite demand. After an initial window of duration T, flow (2,6) will be throttled to its RIAS fair rate of 1/4 on link 5. However, flow (1,3) will initially have its rate throttled to 1/2 rather than 3/4, as there is no way yet for node 1 to know that flow (2,6) is bottlenecked elsewhere. Hence it will take a second interval T in which the unused

[8]This sub-allocation could also be scaled to the demand using the max_min operator. For simplicity, we consider equal sub-allocation here.

capacity at link 2 can be signalled to node 1, after which flow (1,3) will transmit at its RIAS fair rate of 3/4.

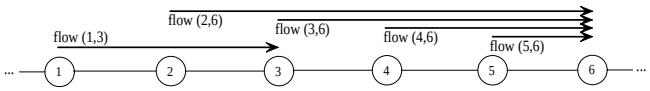


Figure 6: Upstream Parallel Parking Lot

In practical scenarios, there are many additional factors affecting convergence time. We experimentally study DVSR's convergence time through simulations in Section 5.

3.2 DVSR Protocol

In the discussion above, we presented DVSR's conceptual operation in an idealized setting. Here, we describe the DVSR protocol as implemented in the simulator and testbed. We divide the discussion into four parts: scheduling of station vs. transit packets, computation of the feedback signal (control message), transmission of the feedback signal, and rate limit computation.

For scheduling, we allow two policies to arbitrate service among station and transit traffic. The first policy is first-in first-out scheduling of all offered traffic (station or transit) and the second is strict priority of transit traffic over station traffic (as performed by the Aladdin algorithm). As with both Gandalf and Aladdin scheduling policies, the motivation for the choices of FIFO and SP is to have simple scheduling on the transit path amenable to high speed and cost effective implementation. In the hardware and simulations, we perform FIFO scheduling, and we discuss tradeoffs between the choice of FIFO vs. SP in Section 4.3.

As above, for hardware simplicity, the node is not required to perform weighted fair queueing nor any of its variants. Hence, virtual time (or an approximation of virtual time) is not available from the scheduler (e.g., if the node were scheduling according to IA-granularity deficit round robin (DRR) [19] or start-time fair queueing [9] the scheduler itself would yield a mechanism to approximate virtual time). Here, we describe a simple mechanism to lower bound the evolution of virtual time over an interval T using per-ingress byte counters.

As inputs to the algorithm, a node measures the number of arriving bytes from each ingress node, including the station, over the window of duration T.[9] We denote the measurement at this node from ingress node i as l_i (omitting the node superscript for simplicity).

First, we observe that the exact value of $v(t) - v(t-T)$ cannot be derived only from byte counters as $v(t)$ exposes shared congestion whereas byte counts do not. For example, consider that two packets from two ingress nodes arrive in a window of duration T. If the packets arrive back-to-back, then $v(t)$ increases by 1 over an interval of 2 packet transmission times. On the other hand, if the packets arrive separately so that their service does not overlap, then $v(t)$ increases from 0 to 1 twice. Thus, the total increase in the former case is 1 and in the latter case is 2, with both cases having a total backlogging interval of 2 packet transmission times.

However, a simple lower bound to $v(t) - v(t-T)$ can be computed by observing that the minimum increase in $v(t)$ occurs if all packets arrive at the beginning of the interval. This minimum increase will then provide a lower bound to the true fair rate. We denote F as this bound on $\frac{v(t)-v(t-T)}{T} + (1-b)$ at a particular node. Moreover, consider that the byte counts from each ingress node are ordered such that $l_1 \leq l_2 \leq \cdots \leq l_k$ for k flows transmitting any traffic during the interval. Then F is computed every T seconds as given by the pseudo code of Table 1.[10]

Table 1: IA-fair Rate Computation at Intervals T

```
if ( b < 1)  {F = l_k/CT + (1 - b)}
else {
   i=1
   F = 1/k
   Count = k
   Rcapacity = 1
   while  (( l_i/CT < F )&&(l_k/CT ≥ F)) {
      Count- -
      Rcapacity -= l_i/CT
      F = Rcapacity / Count
      l_i = l_{i+1}
   }
}
```

Note that when $b < 1$ (the link is not always busy over the previous interval), the fair rate F is simply the largest ingress-aggregated flow transmission rate l_k/CT plus the unused capacity. When $b = 1$, the pseudocode computes the max-min fair allocation for the largest ingress-aggregated flow so that F is given by $F = max\_min_k(1, l_1/CT, l_2/CT, \cdots l_k/CT)$.

Implementation of the algorithm has several aspects not yet described. First, b is easily computed by dividing the number of bytes transmitted by CT, the maximum number of bytes that could be serviced in T. Second, ordering the byte counters such that $l_1 \leq l_2 \leq \cdots \leq l_k$ requires a sort with complexity O($k \log k$). For a 64 node ring with shortest path routing, the maximum value of k is 32 such that $k \log k$ is 160. Finally, the main while loop in Table 1 has at most k iterations. As DVSR's computational complexity does not increase with the link capacity, and typical values of T are 0.1 to 5 msec, the algorithm is easily performed in real time in our implementation's 200 MHz network processor.

We next address transmission of the feedback signal. In our implementation, we construct a single N-byte control message containing each node's most recently computed value of F such that the message contains $F^1, F^2, \cdots, F^N$ for the N-node ring. Upon receiving a control message, node n replaces the n^{th} byte with its most recently com-

[9] Thus the measurements available to the algorithm are identical to those of the Aladdin draft.

[10] For simplicity of explanation, we consider the link capacity C to be in units bytes/sec and consider all nodes to have equal weight.

puted value of F^n as determined according to Table 1. An alternate messaging approach more similar to Gandalf is to have each node periodically transmit messages with a single value F^n vs. having all values in a circulating message. Our adopted approach results in fewer control message packet transmissions.

The final step is for nodes to determine their rate controller values given their local measurements and current values of F^i. This is achieved as described in Section 4.1.3 in which each (ingress) node sub-allocates its per-link fair rates to the flows with different egress nodes.

3.3 Discussion

We make several observations about the DVSR algorithm. First, note that if there are N nodes forwarding traffic through a particular transit node, rate controllers will never be set to rates below $1/N$, the minimum fair rate. Thus, even if all bandwidth is temporarily reclaimed by other nodes, each node can immediately transmit at this minimum rate; after receiving the next control message, upstream nodes will throttle their rates to achieve fairness at timescales greater than T; until T, packets are serviced in FIFO order.

Next, observe that by weighting ingress nodes, any set of minimum rates can be achieved, provided that the sum of such minimum rates is less than the link capacity.

Third, while we have chosen FIFO scheduling to arbitrate service order among station and transit traffic, we note that alternate scheduling algorithms can also be used. For example, if transit traffic is given strict priority over station traffic, DVSR will have a lossless transit path. However, the drawback of SP scheduling is that a station could be delayed in initially accessing the ring if upstream nodes have previously reclaimed unused capacity, i.e., the station traffic may not be transmitted until upstream nodes receive the updated control message expressing the station's new demand. A third choice for the scheduler is a variant of weighted fair queueing such as deficit round robin. Compared to FIFO and SP, DRR would have improved short-term fairness at timescales less than T, and outputs of the scheduler could also aid in the computing the IA virtual time approximation. The tradeoff is the additional transit path complexity as compared to FIFO and SP. As described in Section 2, current IEEE 802.17 proposals use strict priority and FIFO with buffer thresholds primarily for transit-path hardware simplicity, and we also adopt FIFO and SP for DVSR implementation.

Finally, we observe that DVSR does not low pass filter control signal values at transit nodes nor rate limiter values at stations. The key reason is that the system has a natural averaging interval built in via periodic transmission of control signals. By selecting an expressive control signal (namely, a bound on the average increase in IA virtual time as opposed to the station transit rate) no further damping is required. We experimentally study the algorithms' relative convergence times in Section 5.

4 Analysis of DVSR Fairness

There are many factors of a realistic system that will result in deviations between DVSR service rates and ideal RIA fair rates. Here, we isolate the issue of temporal information aggregation and develop a simple theoretical model to study how T impacts system fairness. The technique can easily be extended to study the impact of propagation delay, an issue we omit for brevity.

4.1 Scenario

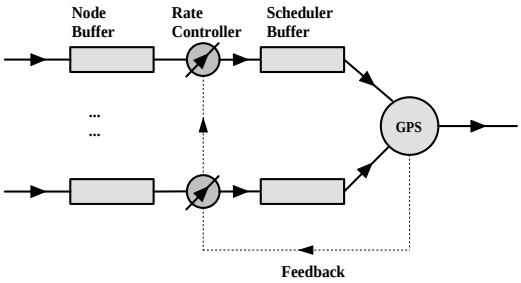


Figure 7: Single Node Model for DVSR

We consider a simplified but illustrative scenario with remote fair queueing and temporally aggregated feedback as in Figure 4. We further assume that the multiplexer is an ideal fluid GPS server,[11] and that the propagation delay is $\Delta = 0$. We consider two flows i and j that have infinite demand and are continuously backlogged. For all other flows we consider the worst case traffic pattern that maximizes the service discrepancy between flows i and j. Thus, Figure 7 depicts the analysis scenario and highlights the relative roles of the node buffer queueing station traffic at rate controllers vs. the scheduler buffer queueing traffic at transit nodes.

We say that a flow is *node-backlogged* if the buffer at its ingress node's rate controller is non-empty and that a flow is *scheduler-backlogged* if the (transit/station) scheduler buffer is non-empty. Moreover, whenever the available service rate at the GPS multiplexer is larger than the rate limiter value in DVSR, the flow is referred to as *over-throttled*. Likewise, if the available GPS service rate is smaller than the rate limiter value in DVSR, the flow is *under-throttled*. Note that as we consider flows with infinite demand, flows are always node-backlogged such that traffic enters the scheduler buffer at the rate controllers' rates. Observe that the scheduler buffer occupancy increases in under-throttled situation. However, while an over-throttled situation may result in a flow being under-served, it may also be over-served if the flow has traffic queued previously.

4.2 Fairness Bound

To characterize the deviation of DVSR from the reference model for the above scenario, we first derive an upper bound on the total amounts of over- and under-throttled traffic as a function of the averaging interval T.

For notational simplicity, we consider fixed size packets

[11] The true DVSR scheduler, packet FIFO, would be intractable for the analysis below.

such that time is slotted, and denote $v(k)$ as the virtual time at time kT. Moreover, let $b(k)$ denote the total non-idle time in the interval $[kT, (k+1)T]$ and denote the number of flows (representing ingress nodes) by N. The bound for under-throttled traffic is derived as follows.

Lemma 1 *A node-backlogged flow in DVSR can be under throttled by at most* $(1 - \frac{1}{N})CT$.

Proof: For a node-backlogged flow i, an under-throttled situation occurs when the fair rate decreases, since the flow will temporarily be throttled using the previous higher rate. In such a case, the average slope of $v(t)$ decreases between times kT and $(k+1)T$. For a system with N flows, the worst case of under-throttling occurs when the slope repeatedly decreases for N consecutive periods of duration T. Otherwise, if the fair rate increases, flow i will be over throttled, and the occupancy of the scheduler buffer is decreasing during that period. Thus, assuming flow i enters the system at time 0, and denoting $U_i(N)$ as the total amount of under-throttled traffic for flow i by time N, we have

$$\begin{aligned} U_i(N) &= \sum_{k=0}^{N-1}((v(k)-v(k-1))-(v(k+1)-v(k))) \\ &= (v(0)-v(-1))-(v(N)-v(N-1)) \\ &\leq (C-\tfrac{1}{N}C)T \end{aligned}$$

since $v(k+1) - v(k)$ is the total service obtained during slot kT for flow i as well as the total throttled traffic for slot $(k+1)T$. The last step holds because for a flow with infinite demand, $v(k) - v(k-1)$ is between $\frac{1}{N}CT$ and CT during an under-throttled period. ■

Similarly, the following lemma establishes the bound for the over-throttled case.

Lemma 2 *A node-backlogged flow in DVSR can be over throttled by at most* $(1 - \frac{1}{N})CT$.

Proof: For a node backlogged flow i, over throttling occurs when the available fair rate increases. In other words, a flow will be over throttled when the average slope of $v(t)$ increases from kT to $(k+1)T$. The worst case is when this occurs for N consecutive periods of duration T. For over-throttled situations, the server can potentially be idle. According to DVSR, the total throttled amount for time slot $(k+1)$ will be $v(k+1) - v(k) + (1 - b(k))CT$. Thus, assuming flow i enters the system at time 0, and denoting $O_i(N)$ as the over-throttling of flow i by slot N, we have that

$$\begin{aligned} O_i(N) &\leq \sum_{k=0}^{N-1}(\min(1, v(k+1)-v(k)+(1-b(k))CT)) \\ &- \min(1, (v(k)-v(k-1)+(1-b(k-1))CT)) \\ &= \min(1, v(N)-v(N-1)+(1-b(N-1))CT) \\ &- \min(1, v(0)-v(-1)+(1-b(-1))CT) \\ &\leq (C-\tfrac{1}{N}C)T \end{aligned}$$

where the last step holds since $(v(k) - v(k-1) + (1 - b(k-1))CT$ is no less than $\frac{1}{N}CT$. ■

Lemmas 1 and 2 are illustrated in Figure 8. Let $f(t)$ (labelled "fair share") denote the cumulative (averaged) fair share for flow i in each time slot given the requirements in this time slot. Let $p(t)$ (labelled "rate controller") denote the throttled traffic for flow i. Lemmas 1 and 2 specify that $p(t)$ will be within the range of $(1 - \frac{1}{N})CT$ of $f(t)$.

Furthermore, let $s(t)$ (labelled "service obtained") denote the cumulative service for flow i. Then DVSR guarantees that if flow i has infinite demand, $s(t)$ will not be less than $f(t) - (1 - \frac{1}{N})CT$. This can be justified as follows. As long as $s(t)$ is less than $p(t)$ (i.e., flow i is scheduler backlogged), flow i is guaranteed to obtain a fair share of service. Hence, the slope of $s(t)$ will be no less than that of $f(t)$. Otherwise, flow i would be in an over-throttled situation, and $s(t) = p(t)$, and from Lemma 2, $p(t)$ is no less than $f(t) - (1 - \frac{1}{N})CT$. Also notice that $s(t)$ can be no larger than $p(t)$, so that the service $s(t)$ for flow i is within the range of $(1 - \frac{1}{N})CT$ of $f(t)$ as well.

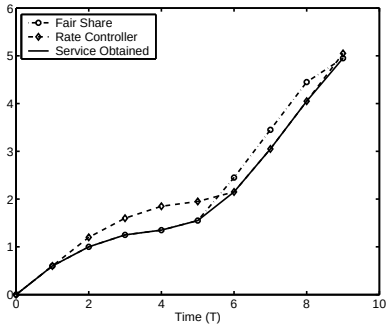


Figure 8: Illustration of Fairness Bound

From the above analysis, we can easily derive a fairness bound for two flows with infinite demand as follows.

Lemma 3 *The service difference during any interval for two flows* i *and* j *with infinite demand is bounded by* $2(C - \frac{1}{N}C)T$ *under DVSR.*

Proof: Observe that scheduler-backlogged flows will get no less than their fair shares due to the GPS scheduler. Therefore, for an under-throttled situation, each flow will receive no less than its fair share. Hence unfairness only can occur during over-throttling. In such a scenario, a flow can only obtain additional service of its under-throttled amount. On the other hand, a flow can at most be underserved by its over-throttled amount. From Lemmas 1 and 2, this amount can at most be $2(C - \frac{1}{N}C)T$. ■

Finally, note that for the special case of $T = 0$, the bound goes to zero so that DVSR achieves perfect fairness without any over/under throttling.

4.3 Discussion

The above methodology can be extended to multiple DVSR nodes in which each flow has one node buffer (at the ingress point) but multiple scheduler buffers. In this case, under-throttled traffic may be distributed among multiple scheduler buffers. On the other hand, for multiple nodes, to maximize spatial reuse, DVSR will rate control a flow at the ingress node using the minimum throttling rate from all the links. By substituting the single node throttling rate

with the minimum rate among all links, Lemmas 1 and 2 can be shown to hold for the multiple node case as well.

Despite the simplified scenario for the above analysis, it does provide a simple if idealized fairness bound of $2(C - \frac{1}{N}C)T$. For a 1 Gb/sec ring with 64 nodes and T = 0.5 msec, this corresponds to a moderate maximum unfairness of 125 kB, i.e., 125 kB bounds the service difference between two infinitely backlogged flows under the above assumptions.

5 Simulation Experiments

In this section, we use simulations to study the performance of DVSR and provide comparisons with Gandalf. Moreover, as a baseline we compare with a Gigabit Ethernet (GigE) Ring that has no distributed bandwidth control algorithm and simply services arriving packets in first-in first-out order.[12]

We divide our study into two parts. First, we study DVSR in the context of the basic RPR goals of achieving spatial reuse and fairness. We also explore interactions between TCP congestion control and DVSR's RIAS fairness objectives. Second, we compare RPR algorithms focusing on convergence time, oscillations, and throughput. We highlight the case of unbalanced traffic that causes throughput degradations to Gandalf and point out its root causes. While our study is necessarily not exhaustive, we choose a number of scenarios that illustrate the challenges and key issues in RPR algorithm design.

All DVSR simulation results are obtained with our *ns-2* implementation. All Gandalf results are obtained using the OPNET simulation modules from the IEEE 802.17 development group [10]. Unless otherwise specified, we consider 622 Mbps links (OC-12), 200 KByte buffer size, 1 kByte packet size, and 0.1 msec link propagation delay between each pair of nodes. For a ring of N nodes, we set T to be $0.1N$ msec such that one DVSR control packet continually circulates around the ring.

5.1 Fairness and Spatial Reuse

5.1.1 Fairness in the Parking Lot

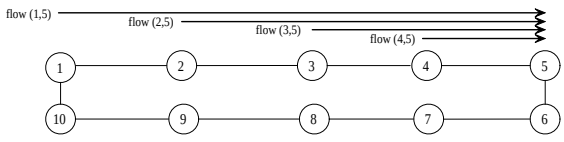


Figure 9: Parking Lot

We first consider the parking lot scenario with a ten-node ring as depicted in Figure 9 and widely studied in [10]. Four constant-rate UDP flows (1,5), (2,5), (3,5), and (4,5) each transmit at an offered traffic rate of 622 Mbps, and we measure each flow's throughput at node 5. We perform the experiment with DVSR, Gandalf, and GigE (for comparison, we set the GigE link rate to 622 Mbps) and present the results in Figure 10. The figure depicts the average normalized throughput for each flow over the 5 second simulation, i.e., the total received traffic at node 5 divided by the simulation time. The labels above the bars represent the un-normalized throughput in Mbps.

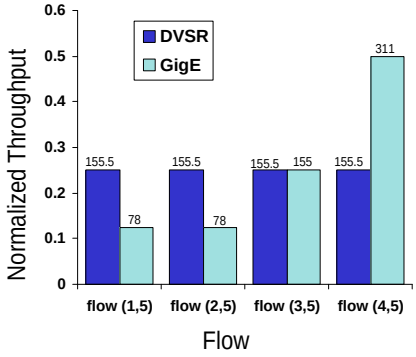


Figure 10: Parking Lot

We make the following observations about the figure. First, DVSR and Gandalf achieve the correct RIAS fair rates (622/4) to within $\pm 1\%$. In contrast, without the coordinated bandwidth control of the RPR algorithms, GigE fails to ensure fairness, with flow (4,5) obtaining 50% throughput share whereas flow (1,5) obtains 12.5%.[13]

5.1.2 Performance Isolation for TCP Traffic

Unfairness among congestion-responsive TCP flows and non-responsive UDP flows is well established. However, suppose one ingress node transmits only TCP traffic whereas all other ingress nodes send high rate UDP traffic. The question is whether DVSR can still provide RIAS fair bandwidth allocation to the node with TCP flows, i.e., can DVSR provide inter-node performance isolation? The key issue is whether DVSR's reclaiming of unused capacity to achieve spatial reuse will hinder the throughput of the TCP traffic.[14]

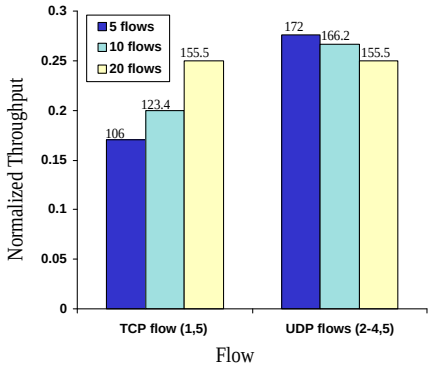


Figure 11: DVSR's TCP and UDP Flow Bandwidth Shares

To answer this question, we consider the same parking

[12] GigE nodes can employ a simple form of coordination (backpressure) by sending collision signals when buffers are full.

[13] For DVSR, we have repeated these and other experiments with Pareto on-off flows with various parameters and found identical average throughputs. The issue of variable rate traffic is more precisely explored with the TCP and convergence-time experiments below.

[14] Current Gandalf OPNET modules are not compatible with TCP modules, hence we limit our discussion to DVSR.

lot topology of Figure 9 and replace flow (1,5) with multiple TCP micro-flows, where each micro-flows is a long-lived TCP Reno flow (e.g., each representing a large file transfer). The remaining three flows are each constant rate UDP flows with rate 0.3 (186.6 Mbps).

Ideally the TCP traffic would obtain throughput 0.25, which is the RIAS fair rate between nodes 1 and 5. However, Figure 11 indicates that whether this rate is achieved depends on the number of TCP micro-flows composing flow (1,5). For example, with only 5 TCP micro-flows, the total TCP throughput for flow (1,5) is 0.17, considerably above the pure excess capacity of 0.1, but below the target of 0.25. The key reason is that upon detecting loss, the TCP flows reduce their rate providing further excess capacity for the aggressive UDP flows to reclaim. The TCP flows can eventually reclaim that capacity via linear increase of their rate in the congestion avoidance phase, but their throughput suffers on average. However, this effect is mitigated with additional aggregated TCP micro-flows such that for 20 or more micro-flows, the TCP traffic is able to obtain the same share of ring bandwidth as the UDP flows. The reason is that with highly aggregated traffic, loss events do not present the UDP traffic with a significant opportunity to reclaim excess bandwidth, and DVSR can fully achieve RIAS fairness. In contrast, for GigE and 20 TCP flows, the TCP traffic obtains a throughput share of 13%, significantly below its fair share of 25%. Thus, GigE rings cannot provide the node-level performance isolation provided by DVSR rings.

5.1.3 RIAS vs. Proportional Fairness for TCP Traffic

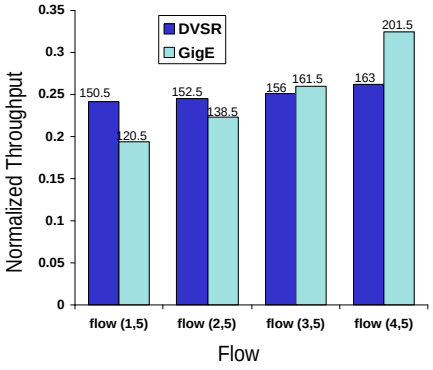


Figure 12: DVSR Throughputs for TCP Micro-Flows

Next, we consider the case that each of the four flows in the parking lot is a single TCP micro-flow, and present the corresponding throughputs for DVSR and GigE in Figure 12. As expected, with a GigE ring the flows with the fewest number of hops and lowest round trip time receive the largest bandwidth shares. However, DVSR seeks to eliminate such spatial bias and provide all ingress nodes with an equal share. For DVSR and a single flow per ingress this is achieved to within approximately $\pm 8\%$. This margin narrows to $\pm 1\%$ by 10 TCP micro-flows per ingress node (not shown). Thus, with sufficiently aggregated TCP traffic, a DVSR ring appears as a single node to TCP flows such that there is no bias to different RTTs.

5.1.4 Spatial Reuse in the Parallel Parking Lot

We now consider the spatial reuse scenario of the Parallel Parking Lot (Figure 2) again with each flow offering traffic at the full link capacity (and hence, "balanced" traffic load). The rates that achieve IA fairness while maximizing spatial reuse are 0.25 for all flows except flow (1,2) which should receive all excess capacity on link 1 and receive rate 0.75.

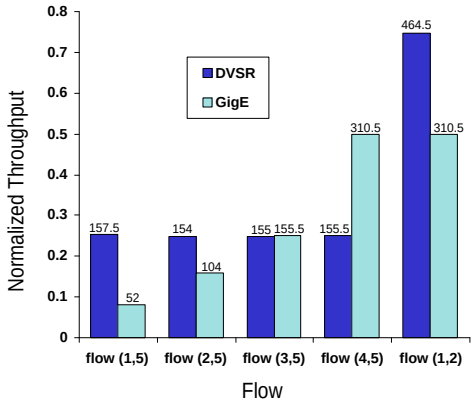


Figure 13: Spatial Reuse in Parallel Parking Lot

Figure 13 shows that the average throughput for each flow for DVSR is within $\pm 1\%$ of the RIAS fair rates. Gandalf can also achieve these ideal rates within the same range if the per-destination queue option is invoked. In contrast, as with the Parking Lot example, GigE favors downstream flows for the bottleneck link 4, and diverges significantly from the RIAS fair rates.

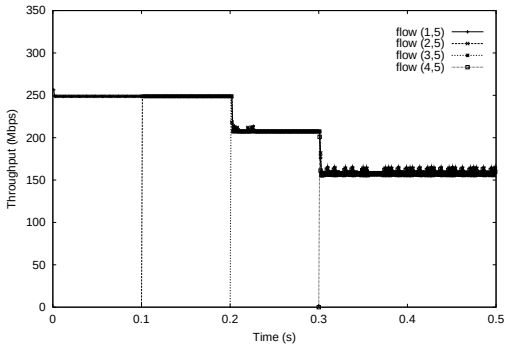
5.2 Comparison of RPR Algorithms

5.2.1 Convergence

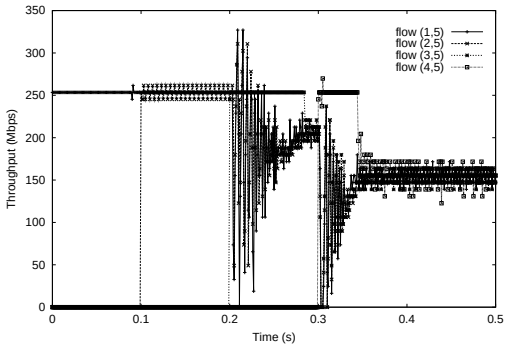
In this experiment, we study the convergence times of the algorithms using the parking lot topology and UDP flows with normalized rate 0.4 (248.8 Mbps). The flows' starting times are staggered such that flows (1,5), (2,5), (3,5), and (4,5) begin transmission at times 0, 0.1, 0.2, and 0.3 seconds respectively.

Figures 14(a) and 14(b) depict the throughput over windows of duration T. Observe that DVSR converges in two ring times, i.e., 2 msec, whereas Gandalf takes approximately 50 msec to converge. Moreover, the range of oscillation during convergence is significantly reduced for DVSR as compared to Gandalf. However, note that the algorithms have a significantly different number of control messages. Gandalf's control update interval is fixed to 0.1 msec so that it has received 500 control messages in 50 msec before converging. In contrast, DVSR has received 2 control messages in 2 msec.

For each of the algorithms, we also explore the sensitivity of the convergence time to the link propagation delay and feedback update time. We find that in both cases, the relationships are largely linear across the range of delays of interest for metropolitan networks. For example, with link propagation delays increased by a factor of 10 so that the ring time is 10 msec, DVSR takes approximately 22 msec to converge, slightly larger than $2T$.



(a) DVSR



(b) Gandalf

Figure 14: Algorithm Convergence Times

5.2.2 Unbalanced Traffic

Here, we return to the case of unbalanced traffic and present experiments to quantify the limitations of Gandalf. We consider an upstream parallel parking lot scenario with a twelve-node ring (half of the ring is presented in Figure 6). Figure 15 shows the throughputs of two flows at a 1 msec timescale (this corresponds to 10 congestion message times with Gandalf's default message interval of 0.1 msec). Observe that the throughput of flow (1,3) is permanently oscillating within the range of 155-500 Mbps. This occurs because flow (2,6) is restricted to rate 1/4 due to downstream congestion: when link 2 is congested, *my_rate* = 1/4 is propagated to node 1, which must then throttle flow (1,3) to rate 1/4 instead of continuing transmission at its RIAS fair rate of 3/4. As the congestion on link 2 will clear when flow (1,3) transmits at rate 1/4, flow (1,3) can gradually increase its rate until congestion occurs again.

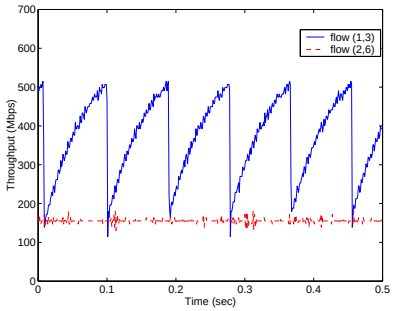


Figure 15: Oscillations in Gandalf

Such oscillations lead to throughput degradation (i.e., only partial spatial reuse), the extent of which is a function of the downstream traffic rate, filter time constants, ring propagation delay, etc. The throughput loss in this scenario is 14%, as depicted in Figure 16. Thus, this simple example illustrates a fundamental limit of throttling all flows to a measured minimum rate during congestion: with unbalanced traffic, the correct fairness mechanism is not for all flows to match the minimum downstream rate.

6 Related Work

The problem of devising distributed solutions to achieve high utilization, spatial reuse, and fairness is a fundamental one that must be addressed in many networking control algorithms. Broadly speaking, TCP congestion control

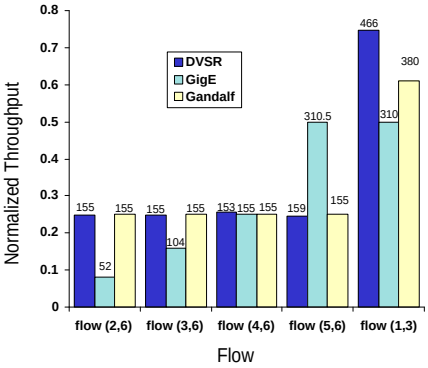


Figure 16: Spatial Reuse in the Upstream Parking Lot

achieves these goals in general topologies (see [11, 15, 16] for example). However, as demonstrated in Section 5, a pure end-point solution to bandwidth allocation in packet rings results in spatial bias favoring nodes closer to a congested gateway. Moreover, end-point solutions do not provide protection against misbehaving flows. In addition, the goals of RPR are quite different than TCP: to provide fairness at the ring ingress-node granularity vs. TCP micro-flow granularity; to provide rate guarantees in addition to fairness, etc. Similarly, ABR rate control [13, 20] can achieve max-min fairness, and as with TCP, provides a natural mechanism for spatial reuse. However, packet rings provide a highly specialized scenario (fixed topology, small propagation delays, homogeneous link speeds, a small number of IA flows, etc.) so that algorithms can be highly optimized for this environment, and avoid the longer convergence times and complexities associated with end-to-end additive-increase multiplicative-decrease protocols.

The problem also arises in specialized scenarios such as wireless ad hoc networks. Due to the finite transmission range of wireless nodes, spatial reuse can be achieved naturally when different sets of communicating nodes are out of transmission range of one another. However, achieving spatial reuse and high utilization is at odds with balancing the throughputs of different flows and hence in achieving fairness. Distributed fairness and medium access algorithms to achieve max-min fairness and proportional fairness can be found in references [14] and [17] respectively. While sharing similar core issues as RPR, such solutions

are unfortunately quite specialized to ad hoc networks and are not applicable in packet rings, as the schemes exploit the broadcast nature of the wireless medium.

Achieving spatial reuse in rings is also a widely studied classical problem in the context of generalizing token ring protocols (see [8, 21] and the references therein). A notable example is the MetaRing protocol [4], which we briefly describe as follows. MetaRing attained spatial reuse by replacing the traditional token of token rings with a 'SAT' (satisfied) message designed so that each node has an opportunity to transmit the same number of packets in a SAT rotation time. In particular, the algorithm has two key threshold parameters K and L, $K \geq L$. A station is allowed to transmit up to K packets on any empty slot between receipt of any two SAT messages (i.e., after transmitting K packets, a node cannot transmit further until receiving another SAT message.) Upon receipt of the SAT message, if the station has already transmitted L packets, it is termed "satisfied" and forwards the SAT message upstream. Otherwise, if the node has transmitted fewer than L packets and is backlogged, it holds the SAT message until L packets are transmitted. While providing significant throughput gains over token rings, the coarse granularity of control provided by holding a SAT signal limits such a technique's applicability to RPR. For example, the protocol's fairness properties were found to be highly dependent on the parameters K and L as well as the input traffic patterns [1]; the SAT rotation time is dominated by the worst case link prohibiting full spatial reuse; etc.

Finally, many of our algorithm design objectives are shared by the Aladdin and Gandalf IEEE 802.17 draft proposals [5, 6]. These algorithms are discussed in Sections 2 and 5, and we make no further comments here.

7 Conclusions

In this paper, we presented of Distributed Virtual-time Scheduling in Rings, a dynamic bandwidth allocation algorithm targeted to achieve high utilization, spatial reuse, and fairness in Resilient Packet Rings. We showed through analysis and simulations that DVSR overcomes limitations of current RPR draft algorithms and fully exploits spatial reuse, rapidly converges typically within two ring times, and closely approximates the Ring Ingress Aggregated with Spatial reuse (RIAS) fairness reference model.

References

[1] G. Anastasi and L. Lenzini. Performance evaluation of a MetaRing MAC protocol carrying asynchronous traffic. *Journal of High Speed Networks*, 6(1), 1997.

[2] L. Balzano, V. Gambiroza, Y. Liu, S. Shaefor, P. Yuan, and E. Knightly. Design, Analysis, and Implementation of DVSR: An Enhanced Protocol for Packet Rings, Rice University Technical Report #TREE0112, January 2002.

[3] D. Bertsekas and R. Gallager. *Data Networks*. Prentice Hall, 1992.

[4] I. Cidon and Y. Ofek. Metaring - a full-duplex ring with fairness and spatial reuse. *IEEE Transactions on Communications*, 41(1):110–120, January 1993.

[5] A. Mekkittikul et al. Alladin Proposal for IEEE Standard 802.17, Draft 1.0, November 2001.

[6] J. Kao et al. Gandalf Proposal for IEEE Standard 802.17, Draft 0.4, November 2001.

[7] J. Kao et al. Darwin Proposal for IEEE Standard 802.17, Draft 1.0, January 2002.

[8] L. Georgiadis, R. Guerin, and I. Cidon. Throughput properties of fair policies in ring networks. *IEEE/ACM Transactions on Networking*, 1(6):718–728, December 1993.

[9] S. Golestani. A self-clocked fair queueing scheme for broadband applications. In *Proceedings of IEEE INFOCOM'94*, Toronto, Ontario, June 1994.

[10] IEEE. IEEE Standard 802.17: Resilient Packet Ring. http://ieee802.org/17 (standard specification in progress).

[11] F. Kelly, A. Maulloo, and D. Tan. Rate control in communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research Society*, 49:237–252, 1998.

[12] E. Knightly, L. Balzano, V. Gambiroza, Y. Liu, S. Shaefor, P. Yuan, and H. Zhang. Achieving High-Performance with Darwin's Fairness Algorithm, March 2002.

[13] H. T. Kung and R. Morris. Credit based flow control for ATM networks. *IEEE Network*, 9(2):40–48, March 1995.

[14] H. Luo, S. Lu, and V. Bharghavan. A new model for packet scheduling in multihop wireless networks. In *Proceedings of ACM/IEEE MOBICOM 2000*, Boston, MA, August 2000.

[15] L. Massoulie and J. Roberts. Bandwidth sharing: objectives and algorithms. In *Proceedings of IEEE INFOCOM '99*, New York, NY, March 1999.

[16] J. Mo and J. Walrand. Fair end-to-end window-based congestion control. *IEEE/ACM Transactions on Networking*, 8(5):556–567, October 2000.

[17] T. Nandagopal, T. Kim, X. Gao, and V. Bharghavan. Achieving MAC layer fairness in wireless packet networks. In *Proceedings of ACM/IEEE MOBICOM 2000*, Boston, MA, August 2000.

[18] A. Parekh and R. Gallager. A generalized processor sharing approach to flow control in integrated services networks: the single-node case. *IEEE/ACM Transactions on Networking*, 1(3):344–357, June 1993.

[19] M. Shreedhar and G. Varghese. Efficient fair queueing using deficit round-robin. *IEEE/ACM Transactions on Networking*, 4(3):375–385, June 1996.

[20] C. Su, G. de Veciana, and J. Walrand. Explicit rate flow control for ABR services in ATM networks. *IEEE/ACM Transactions on Networking*, 8(3):350–361, June 2000.

[21] L. Tassiulas and J. Joung. Performance measures and scheduling policies in ring networks. *IEEE/ACM Transactions on Networking*, 4(5):576–584, October 1995.

[22] D. Tsiang and G. Suwala. The Cisco SRP MAC Layer Protocol, August 2000. Internet RFC 2892.