Design, Analysis, and Implementation of DVSR: A Fair High-Performance Protocol for Packet Rings

Violeta Gambiroza, Ping Yuan, Laura Balzano, *Student Member, IEEE*, Yonghe Liu, Steve Sheafor, *Member, IEEE*, and Edward Knightly, *Member, IEEE*

***Abstract*—The Resilient Packet Ring (RPR) IEEE 802.17 standard is a new technology for high-speed backbone 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 three 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, we show that under unbalanced and constant-rate traffic inputs, the RPR fairness algorithm suffers from severe and permanent oscillations spanning nearly the entire range of the link capacity. Such oscillations hinder spatial reuse, decrease throughput, and increase delay jitter. 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 develop an idealized fairness reference model and bound the deviation in service between DVSR and the 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 versus 50 ms), oscillations are mitigated (e.g., ranges of 0.1% versus up to 100%), and nearly complete spatial reuse is achieved (e.g., 0.1% throughput loss versus 33%). Finally, we provide a proof-of-concept implementation of DVSR on a 1 Gb/s network processor testbed and report the results of testbed measurements.**

***Index Terms*—Dynamic bandwidth allocation, fairness, metro rings, scheduling.**

I. Introduction

THE overwhelmingly prevalent topology for metro networks is a ring. The primary reason is fault tolerance: all nodes remain connected with any single failure of a bidirectional link span. Moreover, rings have reduced deployment costs as compared to star or mesh topologies as ring nodes are only connected to their two nearest neighbors versus to a centralized point (star) or multiple points (mesh).

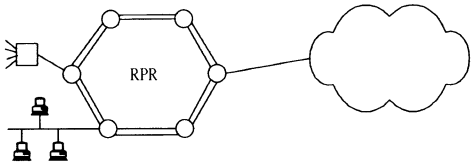


Fig. 1. Illustration of resilient packet ring.

Unfortunately, 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, but suffers from unfairness as well as bandwidth inefficiencies due to forwarding all traffic in the same direction around the ring as dictated by the spanning tree protocol [13]. For example, in the topology of Fig. 1, GigE nodes will obtain different throughputs to the core or hub node depending on their spatial location on the ring. Finally, legacy technologies such as FDDI and DQDB [7], [8] do not employ spatial reuse. For example, FDDI's use of a rotating token requires that only one node can transmit at a time.

The IEEE 802.17 Resilient Packet Ring (RPR) working group was formed in early 2000 to develop a standard for bidirectional packet metropolitan rings. Unlike legacy technologies, 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.[1]

To illustrate spatial reuse and fairness, consider the depicted scenario in Fig. 2, in which four infinite demand flows share link 4 in route to destination node 5. In this "Parallel Parking Lot" example, each of these flows should receive 1/4 of the link bandwidth to ensure fairness. Moreover, to fully exploit spatial

Manuscript received June 12, 2002; revised September 9, 2002; approved by IEEE/ACM Transactions on Networking Editor M. Ajmone Marsan. This work was supported by the National Science Foundation under Grants ANI-0085842 and ANI-0099148 and by a Sloan Fellowship.

V. Gambiroza, P. Yuan, Y. Liu, S. Sheafor, and E. Knightly are with the Electrical and Computer Department, Rice University, TX 77005 USA (e-mail: violeta@rice.edu; pyuan@rice.edu; yhliu@rice.edu; steve.sheafor@alumni.rice.edu; knightly@rice.edu).

L. Balzano was with the Electrical and Computer Department, Rice University, Houston, TX 77005 USA. She is now with Applied Signal Technology, Inc., Sunnyvale, CA 94086 USA (e-mail: sunbeam@alumni.rice.edu).

Digital Object Identifier 10.1109/TNET.2003.820432

[1] Additional RPR goals beyond the scope of this paper include 50-ms fault recovery, similar to that of SONET.

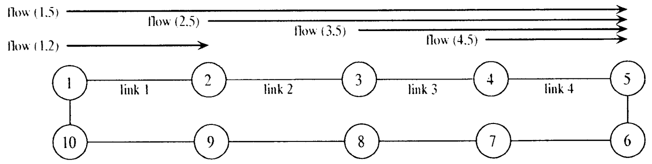


Fig. 2. Topology I: Parallel Parking Lot.

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 [20], 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 versus the desired 3/4. Thus, RPR algorithms must throttle traffic at ingress points based on downstream traffic conditions to achieve these rate allocations.

The RPR standard defines a fairness algorithm that specifies how upstream traffic should be throttled according to downstream measurements, namely, how a congested node will send fairness messages upstream so that upstream nodes can appropriately configure their rate limiters to throttle the rate of injected traffic to its fair rate. The standard also defines the scheduling policy to arbitrate service among transit and station (ingress) traffic as well as among different priority classes. The RPR fairness algorithm has several modes of operation including aggressive/conservative modes for rate computation and single-queue and dual-queue buffering for transit traffic.

Unfortunately, we have found that the RPR fairness algorithm has a number of important performance limitations. First, it is prone to severe and permanent oscillations in the range of the entire link bandwidth in simple "unbalanced traffic" scenarios in which all flows do not demand the same bandwidth. Second, it is not able to fully achieve spatial reuse and fairness. Third, for cases where convergence to fair rates does occur, it requires numerous fairness messages to converge (e.g., 500), thereby hindering fast responsiveness.

The goals of this paper are threefold. First, we provide an idealized reference model termed Ring Ingress Aggregated with Spatial reuse (RIAS) fairness. RIAS fairness achieves maximum spatial reuse subject to providing fair rates to each ingress-aggregated flow at each link. We argue that this fairness model addresses the specialized design goals of metro rings, whereas proportional fairness [10] and flow max-min fairness [3] do not. We use this model to identify key problematic scenarios for RPR algorithm design, including those studied in the standardization process (e.g., "Parking Lot") and others that have not received previous attention (e.g., "Parallel Parking Lot" and "Unbalanced Traffic"). We then use the reference model and these scenarios as a benchmark for evaluating and comparing fairness algorithms, and to identify fundamental limits of current RPR control mechanisms.

Second, we develop a new dynamic bandwidth allocation algorithm termed Distributed Virtual-time Scheduling in Rings (DVSR). Like current implementations, DVSR has a simple transit path without any complex operations such as fair queueing. 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 was performing weighted fair queueing at the granularity of ingress-aggregated traffic. By distributing such control information upstream, we show how nodes can perform simple operations on the collected information and throttle their ingress flows to their ring-wide RIAS fair rates.

Finally, we study the performance of DVSR and the standard RPR fairness algorithm using a combination of theoretical analysis, simulation, and implementation. 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* simulations to compare fairness 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 (unequal input rates), the RPR fairness algorithm suffers from utilization losses of up to 33% in an example with two links and two flows. We also show how DVSR's RIAS fairness mechanism can provide performance isolation among nodes' throughputs. For example, in a Parking Lot scenario (Fig. 4) with even moderately aggregated TCP flows from one node competing for bandwidth with nonresponsive 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 develop a 1-Gb/s network processor implementation of DVSR and present the results of our measurement study on an eight-node ring.

The remainder of this paper is organized as follows. In Section II, we present an overview of the RPR node architecture and fairness algorithms. In Section III, we present the RIAS reference model for fairness. In Section IV, we present a performance analysis of the RPR algorithms and present oscillation conditions and expressions for throughput degradation. In Section V, we present the DVSR algorithm, and in Section VI, we analyze DVSR's fairness properties. Next, we provide extensive simulation comparisons of DVSR, RPR, and GigE in Section VII, and in Section VIII, we present measurement studies from our network processor implementation of DVSR. Finally, we review related work in Section IX and conclude in Section X.

II. Background on IEEE 802.17 RPR

In this section, we describe the basic operation of the RPR fairness algorithm. Due to space constraints, our description necessarily omits many details and focuses on the key mechanisms for bandwidth arbitration. Readers are referred to the standards documents for full details and pseudocode.

Throughout, we consider committed rate (Class B) and best effort (Class C) traffic classes in which each node obtains a minimum bandwidth share (zero for Class C) and reclaims unused bandwidth in a weighted fair manner, here considering equal weights for each node. We omit discussion of Class A traffic

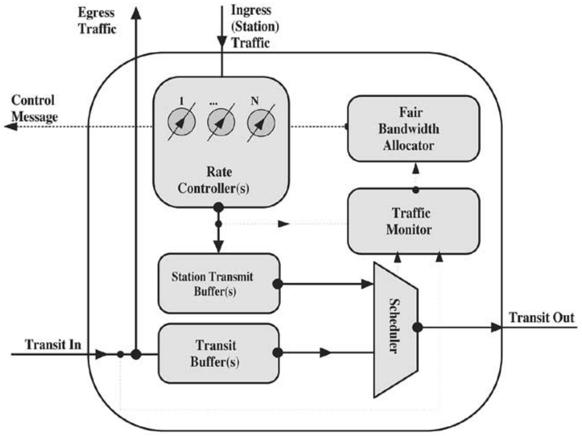


Fig. 3. Generic RPR node architecture.

which has guaranteed rate and jitter, as other nodes are prohibited from reclaiming unused Class A bandwidth.

A. RPR Node Architecture

The architecture of a generic RPR node is illustrated in Fig. 3. First, observe that all station traffic entering the ring is first throttled by rate controllers. In the example of 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 [17], 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 a 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. In *single-queue mode*, the transit path consists of a single FIFO queue referred to as the primary transit queue (PTQ). In this case, the scheduler employs strict priority of transit traffic over station traffic. In *dual-queue mode*, there are two transit path queues, one for guaranteed Class A traffic (PTQ), and the other for Class B and C traffic, called the secondary transit queue (STQ). In this mode, the scheduler always services Class A transit traffic first from PTQ. If this queue is empty, the scheduler employs round-robin service among the transit traffic in STQ and the station traffic until a buffer threshold is reached for STQ. If STQ reaches the buffer threshold, STQ transit traffic is always selected over station traffic to ensure a lossless transit path. In other words, STQ has strict priority over station traffic once the buffer threshold is crossed; otherwise, service is round robin among transit and station traffic.

In both cases, the objective is to ensure hardware simplicity (for example, avoiding expensive per-flow or per-ingress queues on the transit path) and to ensure 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.

B. RPR Fairness Algorithm

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 primary aspect in which the RPR fairness algorithm and DVSR differ and is the focus of the discussion below as well as throughout the paper.

There are two modes of operation for the RPR fairness algorithm. The first, termed Aggressive Mode (AM), evolved from the Spatial Reuse Protocol (SRP) [23] currently deployed in a number of operational metro networks. The second, termed Conservative Mode (CM), evolved from the Aladdin algorithm [5]. Both modes operate within the same framework described as follows. A congested downstream node conveys its congestion state to upstream nodes such that they will throttle their traffic and ensure that there is sufficient spare capacity for the downstream station traffic. To achieve this, a congested node transmits its local fair rate upstream, and all upstream nodes sending to the link must throttle to this same rate. After a convergence period, congestion is alleviated once all nodes' rates are set to the minimum fair rate. Likewise, when congestion clears, stations periodically increase their sending rates to ensure that they are receiving their maximal bandwidth share.

There are two key measurements for RPR's bandwidth control, *forward_rate* and *add_rate*. The former represents the service rate of all transit traffic and the latter represents the rate of all serviced station traffic. Both are measured as byte counts over a fixed interval length *aging_ interval*. Moreover, both measurements are low-pass-filtered using exponential averaging with parameter 1/*LPCOEF* given to the current measurement and 1–1/*LPCOEF* given to the previous average. 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.

At each *aging_ interval*, every node checks its congestion status based on conditions specific to the mode AM or CM. When node n is congested, it calculates its *local_fair_rate* $[n]$, which is the fair rate that an ingress-based flow can transmit to node n. Node n then transmits a fairness control message to its upstream neighbor that contains *local_fair_rate* $[n]$.

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 received *local_fair_rate* $[n]$ and its own *local_fair_rate* $[n-1]$. The objective is to inform upstream nodes of the minimum rate they can send along the path to the destination. If node $(n-1)$ is not congested but its *forward_rate* is greater than the received *local_fair_rate* $[n]$, it forward the fairness control message containing *local_fair_rate* $[n]$ upstream, as this situation indicates that the congestion is due to transit traffic from further upstream. Otherwise, a null-value fairness control message is transmitted to indicate a lack of congestion.

When an upstream node i receives a fairness control message advertising *local_ fair_rate* $[n]$, it reduces its rate limiter values, termed *allowed_rate* $[i][j]$, for all values of j, such that n lies on the path from i to j. The objective is to have upstream

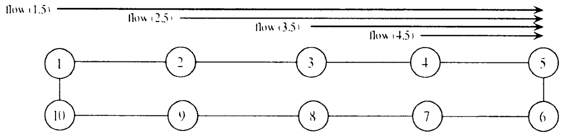


Fig. 4. Topology II: Parking lot.

nodes throttle their own station rate controller values to the minimum rate it can send along the path to the destination. Consequently, station traffic rates will not exceed the advertised *local_ fair_rate* value of any node in the downstream path of a flow. Otherwise, if a null-value fairness control message is received, it increments *allowed_rate* by a fixed value such that it can reclaim additional bandwidth if one of the downstream flows reduces its rate. Moreover, such rate increases are essential for convergence to fair rates even in cases of static demand.

The main differences between AM and CM are congestion detection and calculation of the local fair rate which we discuss below. Moreover, by default AM employs *dual-queue mode* and CM employs *single-queue mode*.

C. Aggressive Mode

Aggressive Mode is the default mode of operation of the RPR fairness algorithm and its logic is as follows. An AM node n is said to be congested whenever

$$STQ\_depth[n] > low\_threshold$$

or

$$forward\_rate[n] + add\_rate[n] > unreserved\_rate$$

where, as above, STQ is the transit queue for Class B and C traffic. The threshold value *low_threshold* is a fraction of the transit queue size with a default value of 1/8 of the STQ size.[2]

When a node is congested, it calculates its *local_fair_rate* as the normalized service rate of its own station traffic, *add_rate*, and then transmits a fairness control message containing *add_rate* to upstream nodes.

Considering the Parking Lot example in Fig. 4, if a downstream node advertises *add_rate* below the true fair rate (which does indeed occur before convergence), all upstream nodes will throttle to this lower rate; in this case, 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.

D. Conservative Mode

Each Conservative Mode node has an access timer measuring the time between two consecutive transmissions of station packets. As CM employs strict priority of transit traffic over station traffic via single queue mode, this timer is used to ensure that station traffic is not starved. Thus, a CM node n is said to be congested if the access timer for station traffic expires or if

$$forward\_rate[n] + add\_rate[n] > low\_threshold.$$

Unlike AM, *low_threshold* for CM is a rate-based parameter that is a fixed value less than the link capacity, 0.8 of the link capacity by default. In addition to measuring *forward_rate* and *add_rate*, a CM node also measures the number of *active* stations that have had at least one packet served in the past *aging_ interval*.

If a CM node is congested in the current *aging_ interval*, but was not congested in the previous one, the *local_ fair_rate* is computed as the total unreserved rate divided by the number of *active* stations. If the node is continuously congested, then *local_ fair_rate* depends on the sum of *forward_rate* and *add_rate*. If this sum is less than *low_threshold*, indicating that the link is under utilized, *local_ fair_rate* ramps up. If this sum is above *high_threshold*, a fixed parameter with a default value that is 0.95 of the link capacity, *local_ fair_rate* will ramp down.

Again considering the Parking Lot example in Fig. 4, when the link between nodes 4 and 5 is first congested, node 4 propagates rate 1/4, the true fair rate. At this point, the link will still be considered congested because its total rate is greater than *low_threshold*. Moreover, because the total rate is also greater than *high_threshold*, *local_ fair_rate* will ramp down periodically until the sum of *add_rate* and *forward_rate* at node 4 is less than *high_threshold* but greater than *low_threshold*. Thus, for CM, the maximum utilization of the link will be *high_threshold*, hence, the name "conservative."

III. Fairness Reference Model for Packet Rings

For flows contending for bandwidth at a single network node, a definition of fairness is immediate and unique. However, for multiple nodes, there are various bandwidth allocations that can be considered to be fair in different senses. For example, proportional fairness allocates a proportionally decreased bandwidth to flows consuming additional resources, i.e., flows traversing multiple hops, whereas max-min fairness does not [3], [10]. Moreover, any definition of fairness must carefully address the granularity of flows for which bandwidth allocations are defined. Bandwidth can be granted on a per-micro-flow basis or alternately to particular groups of aggregated micro-flows.

In this section, we define RIAS fairness, a reference model for achieving fair bandwidth allocation while maximizing spatial reuse in packet rings. The RIAS reference model presented in [11] is now incorporated into the IEEE 802.17 standard's targeted performance objective [9]. We justify the model based on the design goals of packet rings and compare it with proportional and max-min fairness. We then use the model as a design goal in DVSR's algorithm design and the benchmark for general RPR performance analysis.

A. Ring Ingress Aggregated With Spatial Reuse (RIAS) Fairness

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, but not necessarily destined to a single egress node. The targeted service model of packet rings justifies this: to provide fair and/or

[2] *unreserved_rate* is the link capacity minus the reserved rate for guaranteed traffic. As we consider only best-effort traffic, *unreserved_rate* is the link capacity in the rest of this paper.

guaranteed bandwidth to the networks and backbones that it interconnects. Thus, our reference model ensures that an ingress node's traffic receives an equal share of bandwidth on each link relative to other ingress nodes' traffic on that link. The second component of RIAS fairness ensures maximal spatial reuse subject to this first constraint. That is, bandwidth can be reclaimed by IA flows (that is, clients) when it is unused either due to lack of demand or in cases of sufficient demand in which flows are bottlenecked elsewhere.

Below, we present a formal definition that determines if a set of candidate allocated rates (expressed as a matrix R) is RIAS fair. For simplicity, we define RIAS fairness for the case that all ingress nodes have equal weight; the definition can easily be generalized to include weighted fairness. Furthermore, for ease of discussion and without loss of generality, we consider only traffic forwarded on one of the two rings, and assume *fluid* arrivals and services in the idealized reference model, with all rates in the discussion below referring to instantaneous fluid rates. We refer to a *flow* as all unidirectional traffic between a certain ingress and egress pair, and we denote such traffic between ring ingress node i and ring egress node j as flow (i,j) as illustrated in Fig. 2.[3] To simplify notation, we label a tandem segment of N nodes and $N-1$ links such that flow (i,j) traverses node n if $i \leq n \leq j$, and traverses link n if $i \leq n < j$.

Consider a set of infinite-demand flows between pairs of a subset of ring nodes, with remaining pairs of nodes having no traffic between them. Denote R_{ij} as the candidate RAIS fair rate for the flow between nodes i and j. The allocated rate on link n of the ring is then

$$F_n = \sum_{\text{all flows } (i,j) \text{ crossing link } n} R_{ij}. \tag{1}$$

Let C be the capacity of all links in the ring. Then we can write the following constraints on the matrix of allocated rates $R = \{R_{ij}\}$:

$$R_{ij} > 0, \text{ for all flows } (i,j) \tag{2}$$

$$F_n \leq C, \text{ for all links } n. \tag{3}$$

A matrix R satisfying these constraints is said to be feasible. Further, let $IA(i)$ denote the aggregate of all flows originating from ingress node i such that $IA(i) = \sum_j R_{ij}$.

Given a feasible rate matrix R, we say that link n is a bottleneck link with respect to R for flow (i,j) crossing link n, and denote it by $B_n(i,j)$, if two conditions are satisfied. First, $F_n = C$. For the second condition, we distinguish two cases depending on the number of ingress-aggregated flows on link n. If $IA(i)$ is not the only IA flow at link n, then $IA(i) \geq IA(i')$ for all IA flows $IA(i')$, and within ingress aggregate $IA(i)$ $R_{ij} \geq R_{ij'}$ for all flows (i,j') crossing link n. If $IA(i)$ is the only ingress-aggregated flow on link n then $R_{ij} \geq R_{ij'}$ for all flows (i,j') crossing link n.

Definition 1: A matrix of rates R is said to be RIAS fair if it is feasible and if for each flow (i,j), R_{ij} cannot be increased while maintaining feasibility without decreasing $R_{i'j'}$ for some flow (i',j') for which

$$R_{i'j'} \leq R_{ij}, \text{ when } i = i' \tag{4}$$

$$IA(i') \leq IA(i), \text{ at some common link otherwise.} \tag{5}$$

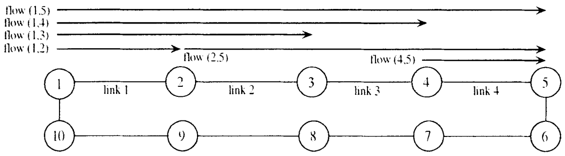


Fig. 5. Illustration of RIAS.

We distinguish two cases in Definition 1. First, in (4), since flows (i,j) and (i',j') have the same ingress node, the inequality ensures fairness among an IA flow's subflows to different egress nodes. In the second case, flows (i,j) and (i',j') have different ingress nodes and $IA(i)$ and $IA(i')$ are both traversing only one or none of $B_n(i,j)$ and $B_{n'}(i',j')$. Thus, the inequality in (6) ensures fairness among different IA flows.

Fig. 5 illustrates the above definition. Assuming that capacity is normalized and all demands are infinite, the RIAS fair shares are as follows: $R_{13} = R_{14} = R_{15} = 0.2$, and $R_{12} = R_{25} = R_{45} = 0.4$. If we consider flow (1,2), its rate cannot be increased while maintaining feasibility without decreasing the rates of flow (1,3), (1,4), or (1,5), where $R_{12} \geq R_{13}, R_{14}, R_{15}$, thus violating (4). If we consider flow (4,5), its rate cannot be increased while maintaining feasibility without decreasing the rate of flow (1,5) or (2,5), and thereby violating (5).

Proposition 1: A feasible rate matrix R is RIAS-fair if and only if each flow (i,j) has a bottleneck link with respect to R.

Proof: Suppose that R is RIAS-fair, and to prove the proposition by contradiction, assume that there exists a flow (i,j) with no bottleneck link. Then, for each link n crossed by flow (i,j) for which $F_n = C$, there exists some flow $(i',j') \neq (i,j)$ such that (4) or (5) is violated (which one depends on the relationship between flows (i',j') and (i,j)). Here, we present the proof for the case that (5) is violated or more precisely when $IA(i') > IA(i)$. The proof is similar for the other case. Now, we can write

$$\delta_n = \begin{cases} C - F_n, & \text{if } F_n < C \\ IA(i') - IA(i), & \text{if } F_n = C \end{cases} \tag{6}$$

where δ_n is positive. Therefore, by increasing the rate of flow (i,j) by $\epsilon \leq \min\{\delta_n : \text{link } n \text{ crossed by flow } (i,j)\}$ while decreasing by the same amount the rate of the flow from $IA(i')$ on links where $F_n = C$, we maintain feasibility without decreasing the rate of any flow $IA(i')$ with $IA(i') \leq IA(i)$. This contradicts Definition 1.

For the second part of the proof, assume that each flow has a bottleneck with respect to R. To increase the rate of flow (i,j) at its bottleneck link while maintaining feasibility, we must decrease the rate of at least one flow from $IA(i')$ (by definition we have $F_n = C$ at the bottleneck link). Furthermore, from the definition of bottleneck link, we also have $IA(i') \leq IA(i)$. Thus, rate matrix R satisfies the requirement for RIAS fairness. ■

[3] Such a flow is composed of aggregated micro-flows such as individual TCP sessions. While the reference model does not address fairness among micro-flows, we consider individual and aggregated TCP traffic in Section VII.

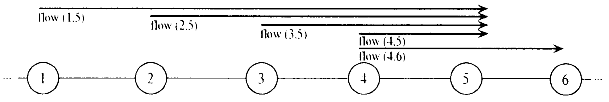


Fig. 6. Topology III: Two-exit Parking Lot.

We make three observations about this definition. First, observe that on each link, each ingress node's traffic will obtain no less than bandwidth C/N provided that its demanded bandwidth is at least C/N.[4] Second, note that these minimum bandwidth guarantees can be weighted to provide different bandwidths to different ingress nodes. Finally, we note that RIAS fairness differs from flow max-min fairness in that RIAS simultaneously considers traffic at two granularities: ingress aggregates and flows. Consequently, as discussed and illustrated below, RIAS bandwidth allocations are quite different that flow max-min fairness as well as proportional fairness.

B. Discussion and Comparison With Alternate Fairness Models

Here, we illustrate RIAS fairness in simple topologies and justify it in comparison with alternate definitions of fairness.

Consider the classical "Parking Lot" topology of Fig. 4. In this example, we have five nodes and four links, and all flows sending to the right-most node numbered 5. If node 5 is a gateway to a core or hub node, and nodes 1–4 connect access networks, then achieving equal or weighted bandwidth shares to the core is critical for packet rings. Suppose that the four flows have infinite demand so that the RIAS fair rates are 1/4 as defined above.

In contrast, a *proportional fair* allocation scales bandwidth allocations according to the total resources consumed [10]. In particular, since flow (1,5) traverses four links whereas flow (4,5) traverses only one, the former flow is allocated a proportionally lesser share of bandwidth. For proportional fairness, the fair rates are given by $R_{15} = .12$, $R_{25} = .16$, $R_{35} = .24$, and $R_{45} = .48$. While proportional fairness has an important role in the Internet and for TCP flow control (see [10], [15], and [18]), in this context it conflicts with our design objective of providing a minimum bandwidth between any two nodes (including gateways), independent of their spatial location.

Second, consider the Parallel Parking Lot topology of Fig. 2 which contains a single additional flow between nodes 1 and 2. In this case, RIAS fairness allows flow (1,2) to claim all excess bandwidth on link 1 such that $R_{12} = 3/4$ and all other rates remain 1/4. Observe that although RIAS fairness provides fair shares using ingress aggregated demand, actual rates are determined on a flow granularity. That is, flows (1,2) and (1,5) have different RIAS fair rates despite having the same ingress node. As described in Section II, allocations having only a single ingress rate for all destinations suffer from underutilization in scenarios such as in Fig. 2.

Finally, consider the "Two-Exit" topology of Fig. 6. Here, we consider an additional node 6 and an additional flow (4,6) so that ingress node 4 now has two flows on bottleneck link 4. In this case, the RIAS fair rates of flows (1,5), (2,5), and (3,5) are still $R_{15} = R_{25} = R_{35} = 1/4$, whereas ingress node 4 divides its IA-fair rate of 1/4 among its two flows such that $R_{45} = R_{46} = 1/8$. This allocation contrasts to a traditional "global" *flow-based max-min fair allocation* of [3, pp. 524–529] in which all five flows would receive rate 1/5, an allocation that is not desirable in packet rings. Extrapolating the example to add more nodes 7, 8, 9, ..., and adding flows (4,7), (4,8), (4,9), ..., it is clear that flow-based max-min fairness rewards an ingress node (node 4) for spreading out its traffic across many egress nodes, and penalizes nodes (1, 2, and 3) that have all traffic between a single ingress-egress pair. RIAS fairness in contrast, ensures that each *ingress* node's traffic receives an equal bandwidth share on each link for which it demands traffic.

IV. Performance Limits of RPR

In this section, we present a number of important performance limits of the RPR fairness algorithm in the context of the RIAS objective.

A. Permanent Oscillation With Unbalanced Constant-Rate Traffic Inputs

The RPR fairness algorithm suffers from severe and permanent oscillations for scenarios with unbalanced traffic. There are multiple adverse effects of such oscillations, including throughput degradation and increased delay jitter. The key issue is that the congestion signals *add_rate* for Aggressive Mode and (*C/number of active stations*) for Conservative Mode do not accurately reflect the congestion status or true fair rate and, hence, nodes oscillate in search of the correct fair rates.

1) Aggressive Mode: Recall that without congestion, rates are increased until congestion occurs. In AM, once congestion occurs, the input rates of all nodes contributing traffic to the congested link are set to the minimum input rate. However, this minimum input rate is not necessarily the RIAS fair rate. Consequently, nodes overthrottle their traffic to rates below the RIAS rate. Subsequently, congestion will clear and nodes will ramp up their rates. Under certain conditions of unbalanced traffic, this oscillation cycle will continue permanently and lead to throughput degradation. Let r_{ij} denote the demanded rate of flow (i, j). The AM oscillation condition is given by the following.

Proposition 2: For a given RIAS rate matrix R, demanded rates r, and congested link j, permanent oscillations will occur in RPR-AM if there is a flow (n, i) crossing link j such that following two conditions are satisfied:

$$\begin{aligned} r_{\text{osc}} = \min_{n<k\leq j, l>j} \min(r_{kl}, R_{kl}) &< R_{ni} \\ r_{\text{osc}} &< r_{ni}. \end{aligned}$$

Moreover, for small buffers and zero propagation delay, the range of oscillations will be from r_{osc} to $\min(r_{ni}, R_{ni})$.

For example, consider AM with 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 (cf. Fig. 7). Here, considering flow (1,3), we have $j = 2$, $r_{\text{osc}} = \epsilon$ and $R_{13} = C - \epsilon$, where $R_{13} > r_{\text{osc}}$ and $r_{13} > r_{\text{osc}}$. Hence, the demands are constant rate and unbalanced.

[4] Note that if the tandem segment has N nodes, the ring topology has $2N$ nodes: if flows use shortest-hop-count paths, each link will be shared by at most half of the total number of nodes on the ring.

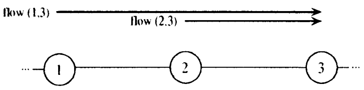


Fig. 7. Oscillation scenario.

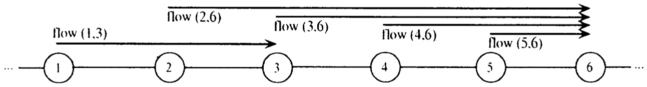


Fig. 8. Topology IV: Upstream Parallel Parking Lot.

Since the aggregate 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. Subsequently, the upstream flow will increase its rate back to $C - \epsilon$ upon receiving null congestion messages. Repeating the cycle, the upstream flow's rate will permanently oscillate between $C - \epsilon$ and the low rate of the downstream flow ϵ.

Observe from Proposition 2 that oscillations also occur with *balanced* input rates but unbalanced RIAS rates. An example of such a scenario is depicted in Fig. 8 in which each flow has identical demand C. In this case, flow (1,3) will permanently oscillate between rates 1/4 and 3/4 since $R_{13} = 3/4, r_{\mathrm{osc}} = 1/4$ and $r_{13} = \infty$, thus $r_{\mathrm{osc}} < R_{13}$ and $r_{13} > r_{\mathrm{osc}}$.

2) Conservative Mode: Unbalanced traffic is also problematic for CM. With CM, the advertised rate is determined by the number of *active* flows when a node first becomes congested for two consecutive *aging_intervals*. If a flow has even a single packet transmitted during the last *aging_interval*, it is considered *active*. Consequently, permanent oscillations occur according to the following condition.

Proposition 3: For a given RIAS rate matrix R, demanded rates r, and congested link j, let n_a denote the number of *active* flows on link j, and n_g denote the number of flows crossing link j that have both demand and RIAS fair rate greater than C/n_a. Ignoring low-pass filtering and propagation delay, permanent oscillations will occur in RPR-CM if there is a flow (n, i) crossing link j such that the following two conditions are satisfied:

$$\min(R_{ni}, r_{ni}) < \frac{C}{n_a}$$

$$n_g \frac{C}{n_a} + S_s > low\_threshold$$

where

$$S_s = \sum_{k \le j, l > j, \min(R_{kl}, r_{kl}) < C/n_a} \min(R_{kl}, r_{kl})$$

Moreover, the lower limit of the oscillation range is C/n_a. The upper limit is less than $low\_threshold$, and depends on the offered load of the n_g flows.

For example, consider a two-flow scenario similar to that above except with the *upstream* flow (1,3) having demand ϵ and the downstream flow having demand C. Since flow (1,3) with rate ϵ is considered *active*, the feedback rate of CM at link 2 is $C/2$, and flow (2,3) will throttle to this rate in the next *aging_interval*. At this point, the arrival rate at node 2 is $C/2+\epsilon$, less than the *low_threshold*, so that congestion clears, and flow (2,3) increases its rate periodically until the downstream link is congested again. Repeating the cycle, the rate of the downstream flow will permanently oscillate between $C/2$ and *low_threshold* $-\epsilon$.

B. Throughput Loss

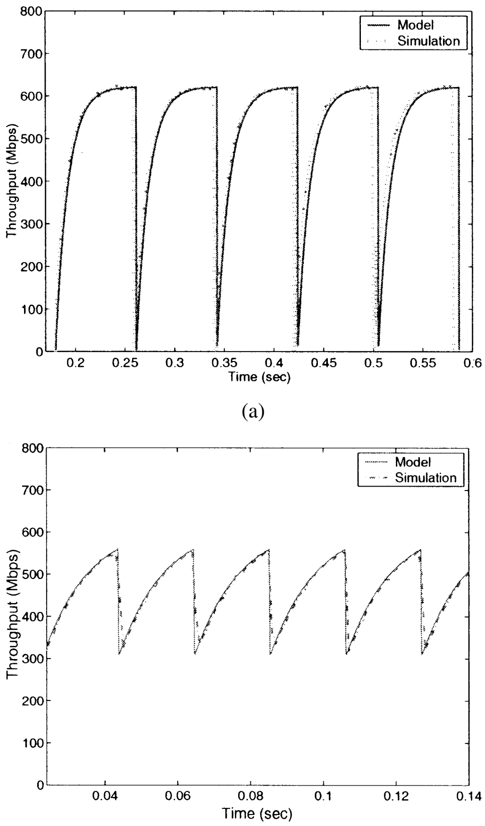
As a consequence of permanent oscillations, RPR-AM and RPR-CM suffer from throughput degradation and are not able to fully exploit spatial reuse.

1) Aggressive Mode: Here, we derive an expression for throughput loss due to oscillations. For simplicity and without loss of generality, we consider two-flow cases as depicted in Fig. 7. We ignore low-pass filtering and first characterize the rate increase part of a cycle, denoting the minimum and maximum rate by $r_{\min}$ and $r_{\max}$, respectively. Further, let τ_a denote the *aging_interval*, τ_p the propagation delay, Q_k the value of the second node's queue size at the end of the kth *aging_interval*, R the RIAS fair rates, and B_t the buffer threshold. Finally, denote r_k as the upstream rate after the kth *aging_interval* and let the cycle begin with $r_0 = r_{\min}$. The rate increase portion of the cycle is then characterized by the following:

$$\begin{aligned}
r_0 &= r_{\min} \\
r_k &= r_{k-1} + \frac{C - r_{k-1}}{rampcoef} \\
r_K &= \{r_k \mid r_k \le r_{\max} \text{ and } r_{k+1} > r_{\max}\} \\
r_L &= \{r_k \mid Q_{k-1} = 0 \text{ and } Q_k > 0\} \\
r_M &= \left\{ r_k \mid \tau_a \sum_{i=L+1}^{i=M-1} (r_i - R) < B_t \right. \\
&\qquad \left. \text{and } \tau_a \sum_{i=L+1}^{i=M} (r_i - R) \ge B_t \right\} \\
r_N &= \{r_k \mid (N - M)\tau_a \ge \tau_p \text{ and } (N - M - 1)\tau_a < \tau_p\}.
\end{aligned}$$

Note that $r_{N+1} = r_{\min}$ such that the cycle repeats according to the definition of RPR-AM. From the expressions above, observe that during one oscillation cycle, the Kth *aging_interval* is the last interval for which the rate is less than the RIAS fair rate, the Lth *aging_interval* is the interval in which the second node's queue starts filling up, the Mth *aging_interval* is the interval in which the second node's queue reaches its threshold, and finally, the Nth *aging_interval* is the interval in which the rate reaches its maximum value $r_{\max}$.

Fig. 9(a) depicts the oscillations obtained according to the above model as well as those obtained by simulation for a scenario in which upstream flow (1,3) has demand 622 Mb/s and



(a)

(b)

Fig. 9. RPR oscillations: Analytical and simulation results. (a) RPR-Aggressive Mode. (b) RPR-Conservative Mode.

downstream flow (2,3) has demand 5 Mb/s.[5] Observe that even ignoring low-pass filtering, the model matches RPR-AM's oscillation cycle very accurately.

From this characterization of an oscillation cycle, we can compute the throughput loss for the flow oscillating between rates r_0 and r_N as follows:

$$\rho_{\text{loss}} = \frac{1}{N} \sum_{k=0}^{k=N} (R - r_k) \tag{7}$$

where R is the RIAS fair rate.

Fig. 10 depicts throughput loss versus the downstream flow (2,3) rate for the two-flow scenario for the analytical model of (7) and simulations. Observe that the throughput loss can be as high as 26% depending on the rate of the downstream flow. Moreover, the analytical model is quite accurate and matches the simulation results within 2%. Finally, observe that the throughput loss is nonmonotonic. Namely, for downstream input rates that are very small, the upstream rate controller value drops dramatically but quickly recovers as there is little congestion downstream. For cases with higher rate downstream flows, the range of oscillation for the upstream rate controller is smaller, but the recovery to full rate is slower due to increased congestion. Finally, if the offered downstream rate is the fair rate (311 Mb/s here), the system is "balanced" and no throughput degradation occurs.

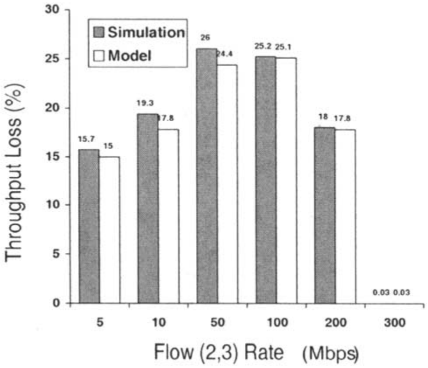


Fig. 10. RPR-AM throughput loss.

2) Conservative Mode: Throughput loss for CM has two origins. First, as described in Section II, the utilization in CM is purposely restricted to less than *high_threshold*, typically 95%. Second, similar to AM, permanent oscillations occur with CM under unbalanced traffic resulting in throughput degradation and partial spatial reuse. We derive an expression to characterize CM throughput degradation in a two-flow scenario as above. Let r_k denote the sending rate of flow (2,3) in the kth *aging_interval* as specified by the RPR-CM algorithm. Moreover, let the oscillation cycle begin with $r_0 = r_{\min} = C/n_a$, where n_a is the number of *active* flows. The following illustrates the rate oscillating behavior of flow (2,3) in a cycle

$$\begin{aligned} r_0 &= \frac{C}{n_a} \\ r_k &= r_{k-1} + \frac{C - r_{k-1}}{rampcoef}, \\ &\quad \text{if } \operatorname{lpf}(r_{k-1} + r_{13}) < low\_threshold \\ r_N &= \{r_k \mid \operatorname{lpf}(r_k + r_{13}) \geq low\_threshold \\ &\quad \text{and } \operatorname{lpf}(r_{k-1} + r_{13}) < low\_threshold\} \end{aligned}$$

where r_{13} is the sending and demanded rate of flow (1,3). The function lpf() is the low-pass filtered total transmit rate of flow (1,3) and flow (2,3) at link 2. When the lpf() rate is less than *low_threshold* at the kth *aging_interval*, link 2 is not congested and flow (2,3) increases its rate with a constant parameter *rampcoef.* At the Nth *aging_interval*, the lpf() rate reaches *low_threshold*, such that link 2 becomes congested again, and consequently, flow (2,3) immediately sets its rate to $r_{\min}$. Thus, the maximum sending rate of flow (2,3) in steady state is r_N.

Notice that link 2 will not be continuously congested after the Nth *aging_interval* because flow (2,3) originates at link 2 such that there is no delay for flow (2,3) to set its rate to $r_{\min}$. Thus, a new cycle starts right after the $(N+1)$th *aging_interval.*

Fig. 9(b) depicts the oscillations obtained from analysis and simulations for an example with the upstream flow (1,3) having input rate 5 Mb/s and the downstream flow (2,3) having input rate 600 Mb/s, and indicates an excellent match despite the model simplifications.

Finally, to analyze the throughput loss of RPR-CM, we consider Parking Lot scenarios with N unbalanced flows originating from N nodes sending to a common destination. For a reasonable comparison, the sum of the demanding rate of all

[5] As described in Section VII, the simulator provides a complete implementation of the RPR fairness algorithms.

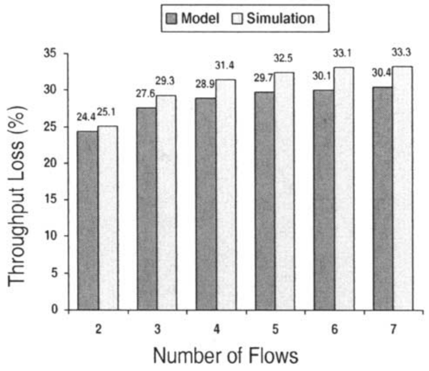


Fig. 11. RPR-CM throughput loss.

flows is 605 Mb/s, which is less then the link capacity. The first to $(N-1)$th flows demand 5 Mb/s, and the Nth flow that is closest to the common destination demands 605–5$(N-1)$ Mb/s. In simulations, the packet size of the Nth flow is 1 kB, and that of the others is 100 B to ensure that the $(N-1)$ flows are *active* in each *aging_interval*.

Fig. 11 depicts throughput loss obtained from simulations as well as the above model using (7). We find that the throughput loss with RPR-CM can be up to 30%, even though the sum of the offered load is less than the link capacity. Finally, observe that the analytical model is again quite accurate and matches the simulation results within 3%

C. Convergence

Finally, the RPR algorithms suffer 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). We explore this effect using simulations in Section VII.

V. 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). It is based on simple computations of measured IA byte counters such that we compute the local bandwidth shares *as if* the node was performing IA-granularity fair queueing, when, in fact, the node is performing FIFO queueing. 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 [20] with an IA flow granularity. We then progressively remove the impractical assumptions of the idealized setting, leading to the network-processor implementation described in Section VIII.

TABLE I
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 (( $l_i/CT < F$ )&&($l_k/CT \geq F$)) {
        Count- -
        Rcapacity -= $l_i/CT$
        $F$ = Rcapacity / Count
        $l_i = l_{i+1}$
    }
}
```

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. Finally, let the operation $\max\_\min_i(C, x_1, x_2, \ldots, x_n)$ return the max-min fair share for the user with index i of a *single* resource with capacity C, and demands $x_1, x_2, \ldots, x_n$. The operational definition of max-min fairness for a *single* resource is a special case of the multilink operational definition of [3, p. 527], and is presented in Table I in the context of DVSR.

A. 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 multihop flows. We next consider each issue independently.

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.).

Fig. 12 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 [labeled MUX in Fig. 12(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

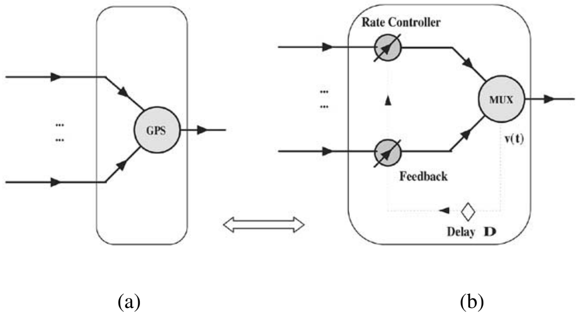


Fig. 12. Illustration of remote fair queueing. (a) GPS server. (b) Approximation.

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 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[6]

$$\rho_i(t) = \min\left(1, \frac{dv(t)}{dt}\right)$$

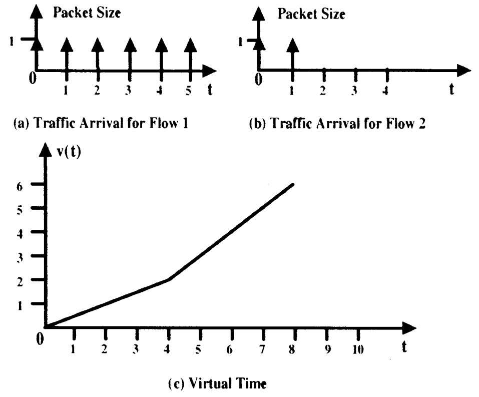
when $v_i(t) > 0$ and 1 otherwise.

For example, consider the four-flow Parking Lot example of Section III. 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 Fig. 12(b) becomes backlogged, $v(t)$ has slope 1/4. With this value instantly fed back, all rate controllers are immediately set to $\rho_i = 1/4$ and flows are serviced at their fair rate.

Suppose at some later time the fourth flow shuts off so that the fair rates are now 1/3. As the fourth 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 four 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.

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.



(a) Traffic Arrival for Flow 1

(b) Traffic Arrival for Flow 2

(c) Virtual Time

Fig. 13. Temporally aggregated virtual time feedback.

For the same single resource example of Fig. 12, 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 zero 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

$$\rho_i(t) = \min\left(\frac{1, (v(t) - v(t-T))}{T + (1-b)}\right). \tag{8}$$

The example depicted in Fig. 13 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 one 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 two flows are backlogged in the interval $[0, 4]$, one flow in the interval $[4, 8]$, and zero flows in [8], [10]. Thus, since $b = 0.8$ the rate limiter value according to (8) 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.

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 VI and via simulations in Section VII.

3) Multinode RIAS Fairness: There are three components to achieving RIAS fairness encountered in multiple node sce-

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

narios. 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 suballocate its per-link IA-fair rate to these flows. For example, in the Two-Exit Parking Lot scenario of Fig. 6, node 4 must divide its rate of 1/4 at link 4 between flows (4,5) and (4,6) such that each rate is 1/8. (Recall that this allocation, as opposed to all flows receiving rate 1/5, is RIAS fair.) The first and second steps can be combined by setting the rate limiter value to be

$$\rho_{i,j}(t) = \min\left(\frac{1, \min_{i \le n < j} \rho_i^n}{|P_i^n|}\right) \tag{9}$$

where ρ_i^n is the single link fair rate at link n as given by (8) and $|P_i^n|$ denotes the number of flows at link n with ingress node i.[7]

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 Fig. 8 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 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.

B. 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 versus transit packets, computation of the feedback signal (control message), transmission of the feedback signal, and rate limit computation.

1) Scheduling of Station versus Transit Packets: As described in Section II, the high speed of the transit path and requirements for hardware simplicity prohibit per-ingress transit queues and therefore prohibit use of fair queueing or any of its variants, even at the IA granularity. Consequently, we employ FIFO scheduling of all offered traffic (station or transit) in both the simulator and implementation.

Recall that the objective of DVSR is to throttle flows to their ring-wide RIAS-fair rate at the ingress point. Once this is achieved and steady state is reached, queues will remain empty and the choice of the scheduler is of little impact. Before convergence (typically less than several ring propagation times in our experiments) the choice of the scheduler impacts the jitter and short-term fairness properties of any fairness algorithm. While a number of variants on FIFO are possible, especially when also considering high-priority Class A traffic, we leave a detailed study of scheduler design to future work and focus on the fairness algorithm.

2) Feedback Signal Computation: As inputs to the algorithm, a node measures the number of arriving bytes from each ingress node, including the station, over a window of duration T.[8] 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 two 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 two packet transmission times.

However, a 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 virtual time, and is used in calculation of the control message's rate. We denote F as $v(t) - v(t-T)/T + (1-b)$ at a particular node. Moreover, consider that the byte counts from each ingress node are ordered such that $l_1 \le l_2 \le \cdots \le l_k$ for k flows transmitting any traffic during the interval. Then F is computed every T seconds as given by the pseudocode of Table I.[9]

Note that when $b < 1$ (the link is not always busy over the previous interval), the value of F is simply the largest IA flow transmission rate l_k/CT plus the unused capacity. When $b = 1$, the pseudocode computes the max–min fair allocation for the largest IA flow so that F is given by $F = \max\_\min_k(1, l_1/CT, l_2/CT, \ldots l_k/CT)$.

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 $l_1 \le l_2 \le \cdots \le l_k$ 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 I has at most k iterations. As DVSR's computational complexity does not increase with link capacity, and typical values of T are 0.1–5 ms, the algorithm is easily performed in real time in our implementation's 200-MHz network processor.

3) Feedback Signal Transmission: 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 $F^1, F^2, \ldots, F^N$ for the N-node ring. Upon receiving a

[7] This suballocation could also be scaled to the demand using the $\max\_\min$ operator. For simplicity, we consider equal suballocation here.

[8] Thus, the measurements used by DVSR are identical to those of RPR.

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

control message, node n replaces the nth byte with its most recently computed value of F^n as determined according to Table I.

An alternate messaging approach more similar to RPR is to have each node periodically transmit messages with a single value F^n versus having all values in a circulating message. Our adopted approach results in fewer control message packet transmissions.

4) Rate Limit Computation: 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 above in which each (ingress) node suballocates its per-link fair rates to the flows with different egress nodes.

C. 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, we note that the DVSR protocol is a distributed mechanism to compute the RIAS fair rates. In particular, to calculate the RIAS fair rates, we first estimate the local IA-fair rates using local byte counts. Once nodes receive their locally fair rates, they adapt their rate limiter values converging to the RIAS rates.

Finally, we observe that unlike the RPR fairness algorithm, 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 a control signal that conveys a bound on the time-averaged increase in IA virtual time as opposed to the station transit rate, no further damping is required.

VI. Analysis of DVSR Fairness

There are many factors of a realistic system that will result in deviations between DVSR service rates and ideal RIAS 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.

A. Scenario

We consider a simplified but illustrative scenario with remote fair queueing and temporally aggregated feedback as in Fig. 12. We further assume that the multiplexer is an ideal fluid GPS server,[10] 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, Fig. 14 depicts the analysis scenario and highlights the relative roles of the node buffer queueing station traffic at rate controllers versus the scheduler buffer queueing traffic at transit nodes.

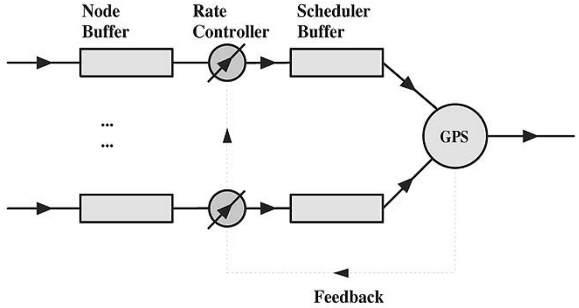


Fig. 14. Single-node model for DVSR.

We say that a flow is *node-backlogged* if the buffer at its ingress node's rate controller is nonempty and that a flow is *scheduler-backlogged* if the (transit/station) scheduler buffer is nonempty. 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 *overthrottled*. Likewise, if the available GPS service rate is smaller than the rate limiter value in DVSR, the flow is *underthrottled*. 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 underthrottled situation. However, while an overthrottled situation may result in a flow being underserved, it may also be overserved if the flow has traffic queued previously.

B. 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 underthrottled traffic as a function of the averaging interval T.

For notational simplicity, we consider fixed-size packets such that time is slotted, and denote $v(k)$ as the virtual time at time kT. Moreover, let $b(k)$ denote the total nonidle time in the interval $[kT, (k+1)T]$ and denote the number of flows (representing ingress nodes) by N. The bound for underthrottled traffic is derived as follows.

Lemma 1: A node-backlogged flow in DVSR can be underthrottled by at most $(1-(1/N))CT$.

Proof: For a node-backlogged flow i, an underthrottled 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 underthrottling occurs when the slope repeatedly decreases for N consecutive periods of duration T. Otherwise, if the fair rate increases, flow i will be overthrottled, and the occupancy of the scheduler buffer is decreasing during that period. Thus, assuming flow i enters the system at time 0, and denoting $U_i(N)$

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

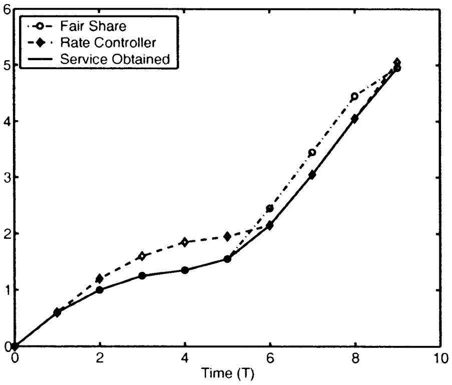


Fig. 15. Illustration of fairness bound.

as the total amount of underthrottled traffic for flow i by time N, we have

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

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 $(1/N)CT$ and CT during an underthrottled period. ■

Similarly, the following lemma establishes the bound for the overthrottled case.

Lemma 2: A node-backlogged flow in DVSR can be overthrottled by at most $(1-(1/N))CT$.

Proof: For a node-backlogged flow i, overthrottling occurs when the available fair rate increases. In other words, a flow will be overthrottled 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 overthrottled 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 overthrottling of flow i by slot N, we have that

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

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

Lemmas 1 and 2 are illustrated in Fig. 15. Let $f(t)$ (labeled "fair share") denote the cumulative (averaged) fair share for flow i in each time slot given the requirements in this time slot. Let $p(t)$ (labeled "rate controller") denote the throttled traffic for flow i. Lemmas 1 and 2 specify that $p(t)$ will be within the range of $(1-1/N)CT$ of $f(t)$.

Furthermore, let $s(t)$ (labeled "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 $f(t) - (1-(1/N))CT$. 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 overthrottled situation, and $s(t) = p(t)$, and from Lemma 2, $p(t)$ is no less than $f(t)-(1-1/N)CT$. 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 $(1-(1/N))CT$ of $f(t)$ as well.

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 $2(C-(1/N)C)T$ under DVSR.

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

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

C. 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, underthrottled 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 $2(C-(1/N)C)T$. For a 1-Gb/s ring with 64 nodes and $T = 0.5$ ms, 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.

VII. Simulation Experiments

In this section, we use simulations to study the performance of DVSR and provide comparisons with the RPR fairness algorithm. Moreover, as a baseline we compare with a GigE ring that has no distributed bandwidth control algorithm and simply services arriving packets in FIFO order.

We divide our study into two parts. First, we study DVSR in the context of the basic RPR goals of achieving spatial reuse

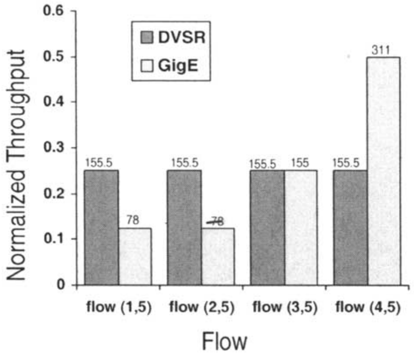


Fig. 16. Parking lot.

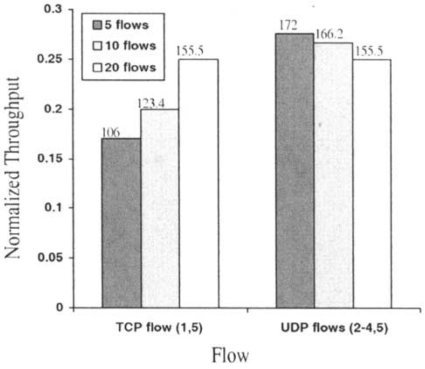


Fig. 17. DVSR's TCP and UDP flow bandwidth shares.

and fairness. We also explore interactions between TCP congestion control and DVSR's RIAS fairness objectives. Second, we compare the convergence times of DVSR and RPR. We do not further consider scenarios with unbalanced traffic that result in oscillation and throughput degradation for RPR as treated in Section IV.

All simulation results are obtained with our publicly available *ns-2* implementations of DVSR and RPR. Unless otherwise specified, RPR simulations refer to the default AM. We consider 622-Mb/s links (OC-12), 200-kB buffer size, 1-kB packet size, and 0.1-ms link propagation delay between each pair of nodes. For a ring of N nodes, we set T to be $0.1N$ ms such that one DVSR control packet continually circulates around the ring.

A. Fairness and Spatial Reuse

1) Fairness in the Parking Lot: We first consider the Parking Lot scenario with a ten-node ring as depicted in Fig. 4 and widely studied in the RPR standardization process [9]. Four constant-rate UDP flows (1,5), (2,5), (3,5), and (4,5) each transmit at an offered traffic rate of 622 Mb/s, and we measure each flow's throughput at node 5. We perform the experiment with DVSR, RPR-AM, RPR-CM, and GigE (for comparison, we set the GigE link rate to 622 Mb/s) and present the results in Fig. 16. The figure depicts the average normalized throughput for each flow over the 5-s 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 megabits per second.

We make the following observations about the figure. First, DVSR as well as RPR-AM and RPR-CM (not depicted) all achieve the correct RIAS fair rates (622/4) to within ±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%.[11]

2) Performance Isolation for TCP Traffic: Unfairness among congestion-responsive TCP flows and nonresponsive 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.

To answer this question, we consider the same Parking Lot topology of Fig. 4 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 Mb/s).

Ideally, the TCP traffic would obtain throughput 0.25, which is the RIAS fair rate between nodes 1 and 5. However, Fig. 17 indicates that whether this rate is achieved depends on the number of TCP micro-flows composing flow (1,5). For example, with only five 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.

3) RIAS Versus Proportional Fairness for TCP Traffic: 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 Fig. 18. As expected, with a GigE ring the flows with the fewest number of hops and lowest round-trip time (RTT) receive the largest bandwidth shares (cf. Section III). 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 ±8%. This margin narrows to ±1% by ten TCP micro-flows per ingress node (not shown). Thus, with sufficiently aggregated TCP

[11] 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.

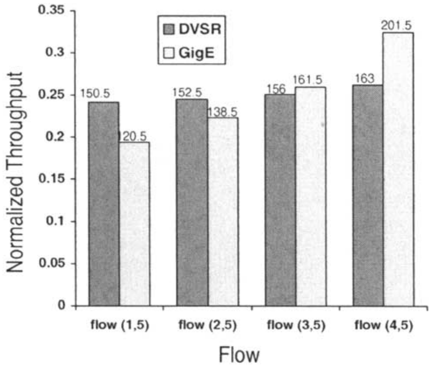


Fig. 18. DVSR throughputs for TCP micro-flows.

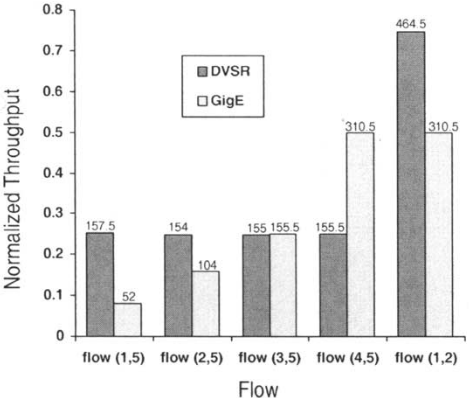


Fig. 19. Spatial reuse in the Parallel Parking Lot.

traffic, a DVSR ring appears as a single node to TCP flows such that there is no bias to different RTTs.

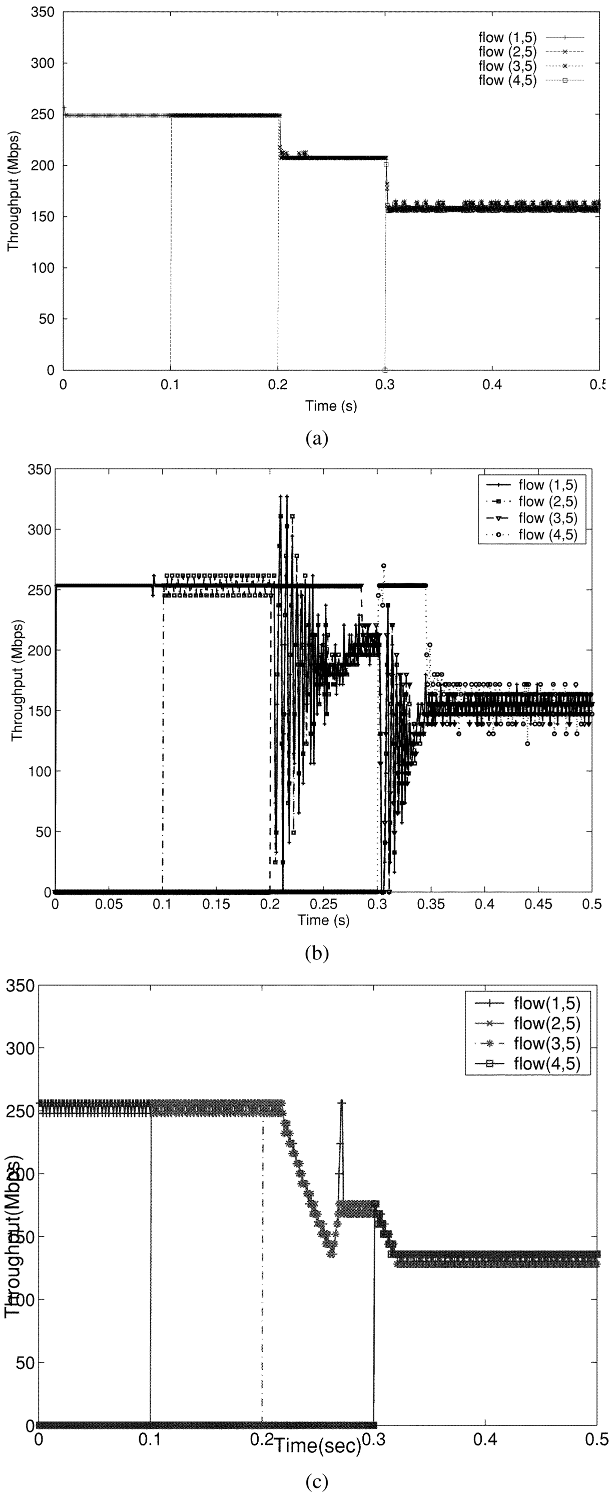
4) Spatial Reuse in the Parallel Parking Lot: We now consider the spatial reuse scenario of the Parallel Parking Lot (Fig. 2) again with each flow offering traffic at the full link capacity (and, hence, "balanced" traffic load). As described in Section III, the rates that achieve IA-fairness while maximizing 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.

Fig. 19 shows that the average throughput for each flow for DVSR is within $\pm 1\%$ of the RIAS fair rates. RPR-AM and RPR-CM can also achieve these ideal rates within the same range when using the per-destination queue option. 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.

B. Convergence Time Comparison

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 Mb/s). 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 s, respectively.

Fig. 20 depicts the throughput over windows of duration T for the three algorithms. Observe that DVSR converges in two ring times, i.e., 2 ms, whereas RPR-AM takes approximately 50 ms to converge, and RPR-CM takes about 18 ms. Moreover, the range of oscillation during convergence is significantly reduced for DVSR as compared to RPR. However, note that the



(a)

(b)

(c)

Fig. 20. Algorithm convergence times. (a) DVSR. (b) RPR-Aggressive Mode. (c) RPR-Conservative Mode.

algorithms have a significantly different number of control messages. RPR's control update interval is fixed to 0.1 ms so that RPR-AM and RPR-CM have received 180 and 500 respective control messages before converging. In contrast, DVSR has received two control messages.

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 ms, DVSR takes approximately 22 ms to converge, slightly larger than $2T$.

Finally, we note that RPR algorithms differ significantly in their ability to achieve spatial reuse with unbalanced traffic. As described in Section IV, RPR-AM and RPR-CM suffer from permanent oscillations and throughput degradation in cases of unbalanced traffic. In contrast DVSR achieves rates within 0.1% of the RIAS rates in simulations of all unbalanced scenarios presented in Section IV.

VIII. Network Processor Implementation

The logic of each node's dynamic bandwidth allocation algorithm depicted in Fig. 3 may be implemented in custom hardware or in a programmable device such as a network processor (NP). We adopt the latter approach for its feasibility in an academic research lab as well as its flexibility to reprogram and test algorithm variants. In this section, we describe our implementation of DVSR on a 2-Gb/s NP testbed. The DVSR algorithm is implemented in assembly language in the NP, utilizing the rate controllers and output queuing system of the NP in the same way that a hardware-only implementation would. The result allows an accurate emulation of DVSR behavior in a realistic environment.[12]

A. NP Scenario

The DVSR implementation is centered around a Vitesse IQ2000 NP (part no. VSC2100 [24]). The IQ2000 has four 200-MHz 32-bit RISC processing cores, each running four user contexts and including 4 kB of local memory. This allows up to 16 packets to be processed simultaneously by the NP. For communication interfaces, it has four 1-Gb/s input and output ports with eight communication channels each, one of which is connected to an eight-port 100-Mb/s Ethernet MAC (part no. VSC2800 [24]). Its memory capacity is 256 MB of external DRAM memory and 4 MB of external SRAM memory.

As described in Section V, the inputs to the DVSR bandwidth control algorithm are byte counts of arriving packets. In the NP, these byte counts are kept per destination for station traffic and per ingress for transit traffic, and are updated with each packet arrival and stored in SRAM. Using these measurements as inputs, the main steps to computing the IA-fair bandwidth as given in Table I are written in a MIPS-like assembly language and performed by the RISC processors.

In our implementation, a single control packet circulates continuously around the ring. The control packet contains N 1-byte virtual-time fair rate values $F_1, \ldots, F_N$ (N is 8 for our testbed and no larger than 256 for IEEE 802.17.) Upon receiving the control packet, node n stores the N bytes to local memory, updates its own value of F_n, and forward the packet to the next upstream node. Using the received $F_1, \ldots, F_N$, the control software computes the rate limiter values as given by (9). The rate limiter values are, therefore, discretized to 256 possible values between zero and the link capacity.

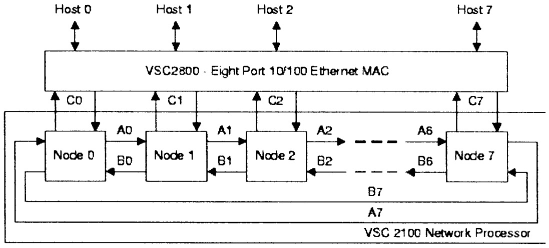


Fig. 21. Testbed configuration.

The output modules for each of the ports contain eight hardware queues per output channel, and each of these queues can be assigned a separate rate limit. Hence, for our eight-node ring, we use these hardware rate limiters to adaptively shape station traffic according to the fairness computation by writing the computed values of the station throttling rates to the output module.

Finally, on the data path, the DRAM of the NP contains packet buffers to hold data on the output queues, with a separate queue for transit versus station traffic, and transit traffic scheduled alternately with the rate-limited station traffic.

Thus, considering the generic RPR node architecture of Fig. 3, the dynamic bandwidth allocation algorithm and forwarding logic is programmed on the NP, and all other components are hardware. On the transit path, the DVSR rate calculation algorithm is implemented in approximately 171 instructions. Moreover, the logic for nodes to compute their ingress rate controller values given a received control signal contains approximately 40 instructions, plus 37 to write the values to hardware. These operations are executed every T seconds. In our implementation, the NP also contains forwarding logic which increases the NP workload.

B. Testbed

In our testbed configuration, we emulate an eight-node ring node on a single NP using 24 interfaces operating at 100 Mb/s each, as illustrated in Fig. 21. For each station connection, seven of the eight queues are assigned to the seven destination nodes on this ring as in Fig. 3. Transit traffic and control traffic occupy two additional queues.

As illustrated in Fig. 21, the eight Ethernet interfaces of the VSC2800 connected to port C provide the eight station connections. Ports A and B of the NP emulate the outer and inner rings respectively, and each channel represents one of the node-to-node connections. The arrival port and channel information is readily available for each packet so that the processor can determine which node to emulate for the current packet. For example, a packet arriving from port A on channel 0 has arrived from the inner ring connection of node 1 (it has come from node 0).

There are several factors in the emulation which may differ from the behavior of a true packet ring. Since the "connections" between nodes are wires within a single chip, the link propagation delay is negligible. In order to have increased latency as in a realistic scenario, the emulation includes a mechanism for delaying a packet by a tightly controlled amount of time before

[12] DVSR assembly language modules are available at http://www.ece.rice.edu/networks/DVSR.

it is transmitted. In the experiments below, we have set these values such that the total ring propagation delay (and, hence, T) is 0.6 ms.

Since all nodes reside in the same physical chip, all information (particularly the rate counters) is accessible to the emulation of all nodes. However, to ensure accurate emulation, all external memory accesses are indexed by the number of the current node, and all control information is read and written to the control packet only.

C. Results

We performed experiments in two basic scenarios: the Parking Lot and unbalanced traffic. For the Parking Lot experiments, we first use an eight-node ring and configure a Parking Lot scenario with two flows originating from nodes 1 and 2 and all with destination node 3. A Unix workstation is connected to each node with the senders running a UDP constant-rate traffic generation program and the receiver running $\mathrm{tcpdump}$. In the experiment, each source node generates traffic at rate 58 Mb/s such that the downstream link is significantly congested. Using on-chip monitoring tools, we found that the byte value of the control message was $0 \times 7F$ in the second node's fields. Consequently, the upstream rates were all correctly set to 100 Mb/s $\times 0 \times 7F/0 \times FF$ and the fair rates were achieved within a narrow margin. Similary, we performed experiments with a three-flow Parking Lot with the upstream flows generating traffic at rate 58 Mb/s and the downstream flow generating traffic at 97 Mb/s. The measured rate limiter values yielded the correct values of 0×55 for all three flows. The throughputs of the three flows were measured using $\mathrm{tcpdump}$ as 33.7, 33.7, and 32.6 Mb/s. Next, we considered the case of unbalanced traffic problematic to RPR. Here, the upstream flow inputs traffic at nearly 100 Mb/s and the downstream flow inputs traffic at rate 42 Mb/s. The measured rate limiter value of the upstream flow was 0×94, correctly set to 58 Mb/s.

In future work, we plan to configure the testbed with 1-Gb/s interfaces and perform a broader set of experiments to study the impact of different workloads (including TCP flows), configurations (including the Parallel Parking Lot), and many of the scenarios explored in Section VII.

IX. 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 achieves these goals in general topologies (see, e.g., [10], [15], and [18]). However, as demonstrated in Section VII, 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 versus TCP micro-flow granularity; to provide rate guarantees in addition to fairness, etc. Similarly, ABR rate control [12], [21] and other distributed fairness protocols [1], [16] 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 [14] and [19], 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 [6], [22], and 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 forward 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 [2]; the SAT rotation time is dominated by the worst case link prohibiting full spatial reuse; etc.

X. Conclusions

In this paper, we presented DVSR, a dynamic bandwidth allocation algorithm targeted to achieve high utilization, spatial reuse, and fairness in packet rings. We showed through analysis, simulations, and implementation that DVSR overcomes limitations of the standard RPR algorithm and fully exploits spatial reuse, rapidly converges (typically within two ring times), and closely approximates our idealized fairness reference model, RIAS. Finally, we note that RIAS and the DVSR algorithm can be applied to any packet ring technology. For example, DVSR can be used as a separate fairness mode for RPR or as a control mechanism on top of Gigabit Ethernet used to ensure fairness in Metro Ethernet rings.

References

[1] I. Akyldiz, J. Liebeherr, and D. Sarkar, “Bandwidth regulation or real-time traffic classes in internetworks,” in *Proc. 15th IEEE Int. Conf. Distributed Computer Systems*, vol. 28, Apr. 1996, pp. 855–872.

[2] G. Anastasi and L. Lenzini, “Performance evaluation of a metaring MAC protocol carrying asynchronous traffic,” *J. High Speed Netw.*, vol. 6, no. 1, pp. 35–52, 1997.

[3] D. Bertsekas and R. Gallager, *Data Networks*. Englewood Cliffs, NJ: Prentice-Hall, 1992.

[4] I. Cidon and Y. Ofek, “Metaring—a full-duplex ring with fairness and spatial reuse,” *IEEE Trans. Commun.*, vol. 41, pp. 110–120, Jan. 1993.

[5] A. Mekkittikul *et al.*, “*Aladdin Proposal for IEEE Standard 802.17, Draft 1.0*,” IEEE, Nov. 2001.

[6] L. Georgiadis, R. Guerin, and I. Cidon, “Throughput properties of fair policies in ring networks,” *IEEE/ACM Trans. Networking*, vol. 1, pp. 718–728, Dec. 1993.

[7] E. Hahne, A. Choudhury, and N. Maxemchuk, “DQDB networks with and without bandwidth balancing,” *IEEE Trans. Commun.*, vol. 40, pp. 1192–1204, July 1992.

[8] *IEEE Standard 802.6: Distributed Queue Dual Bus (DQDB) Access Method and Physical Layer Specifications*, 1994.

[9] *IEEE Standard 802.17: Resilient Packet Ring (Draft Version 1.0)*, http://ieee802.org/17, Aug. 2002.

[10] F. Kelly, A. Maulloo, and D. Tan, “Rate control in communication networks: shadow prices, proportional fairness and stability,” *J. Oper. Res. Soc.*, vol. 49, pp. 237–252, 1998.

[11] E. Knightly, L. Balzano, V. Gambiroza, Y. Liu, S. Shaefor, P. Yuan, and H. Zhang, “Achieving high-performance with Darwin’s fairness algorithm,” IEEE 802.17 Standard Presentation, Mar. 2002.

[12] H. T. Kung and R. Morris, “Credit based flow control for ATM networks,” *IEEE Network*, vol. 9, pp. 40–48, Mar. 1995.

[13] J. Kurose and K. Ross, *Computer Networking: A Top-Down Approach Featuring the Internet*, 2nd ed. Boston, MA: Addison Wesley, 2003.

[14] H. Luo, S. Lu, and V. Bharghavan, “A new model for packet scheduling in multihop wireless networks,” in *Proc. ACM/IEEE MOBICOM*, Boston, MA, Aug. 2000, pp. 76–86.

[15] L. Massoulie and J. Roberts, “Bandwidth sharing: Objectives and algorithms,” in *Proc. IEEE INFOCOM*, vol. 3, New York, NY, Mar. 1999, pp. 1395–1403.

[16] A. Mayer, Y. Ofek, and M. Yung, “Approximating max-min fair rates via distributed local scheduling with partial information,” in *Proc. IEEE INFOCOM*, San Francisco, CA, Mar. 1996, pp. 928–936.

[17] N. McKeown, V. Anantharam, and J. Walrand, “Achieving 100% throughput in an input-queued switch,” in *Proc. IEEE INFOCOM*, San Francisco, CA, Mar. 1996, pp. 296–302.

[18] J. Mo and J. Walrand, “Fair end-to-end window-based congestion control,” *IEEE/ACM Trans. Networking*, vol. 8, pp. 556–567, Oct. 2000.

[19] T. Nandagopal, T. Kim, X. Gao, and V. Bharghavan, “Achieving MAC layer fairness in wireless packet networks,” in *Proc. ACM/IEEE MOBICOM*, Boston, MA, Aug. 2000, pp. 87–98.

[20] A. Parekh and R. Gallager, “A generalized processor sharing approach to flow control in integrated services networks: the single-node case,” *IEEE/ACM Trans. Networking*, vol. 1, pp. 344–357, June 1993.

[21] C. Su, G. de Veciana, and J. Walrand, “Explicit rate flow control for ABR services in ATM networks,” *IEEE/ACM Trans. Networking*, vol. 8, pp. 350–361, June 2000.

[22] L. Tassiulas and J. Joung, “Performance measures and scheduling policies in ring networks,” *IEEE/ACM Trans. Networking*, vol. 4, pp. 576–584, Oct. 1995.

[23] D. Tsiang and G. Suwala, “The Cisco SRP MAC layer protocol,” IETF, RFC 2892, Aug. 2000.

[24] *IQ2000 Family of Network Processors*, http://www.vitesse.com/products/categories.cfm?family_id =5&category_id=16.

Violeta Gambiroza received the B.S. degree in electrical engineering from the University of Belgrade, Serbia, in 1996, and the M.S. degree in electrical and computer engineering from Rice University, Houston, TX, in 2001. She is currently working toward the Ph.D. degree in electrical and computer engineering at Rice University.

Her research interests are in the area of quality of service in wired networks, with emphasis on metropolitan area networks.

Ping Yuan received the B.S. degree (1996) from Tianjin University, China, in 1996 and the M.S. degree from the Institute of Computing Technology, Chinese Academy, China, in 1999, both in computer science. She is currently working toward the Ph.D. degree in electrical and computer engineering at Rice University, Houston, TX.

Her research is focused on quality of service in metropolitan area networks.

Laura Balzano (S’00) received the B.S. degree in electrical and computer engineering from Rice University, Houston, TX, in December 2001.

She is currently an Engineer with Applied Signal Technology, Inc., Sunnyvale, CA, where she develops signal intelligence software. Her research interests include networking algorithms, multidimensional signal processing and the interdisciplinary examination of scientific concepts.

Yonghe Liu received the B.S. and M.S. degrees in control theory and engineering from Tsinghua University, Beijing, China, in 1998 and 1999, respectively. He is currently working toward the Ph.D. degree in electrical and computer engineering at Rice University, Houston, TX.

Since May 2001, he has been working for the Digital Signal Processing Solutions R&D Center, Texas Instruments Incorporated, in the area of wireless LANs. His current research interests are wireless networking, media access control, and system integration.

Steve Sheafor (M’02) received the B.A. and M.E.E. degrees from Rice University, Houston, TX, the Ph.D. degree in electrical engineering from the University of Illinois at Urbana-Champaign, and the M.B.A. degree from the University of Santa Clara, Santa Clara, CA.

He currently serves as Adjunct Professor of Electrical and Computer Engineering at Rice University, and was Founder and Chief Technical Officer of Cornerstone Imaging and Sitera.

Edward Knightly (M’96) received the B.S. degree from Auburn University, Montgomery, AL, in 1991 and the M.S. and Ph.D. degrees from the University of California at Berkeley in 1992 and 1996, respectively.

He is currently an Associate Professor in the Electrical and Computer Engineering and Computer Science Departments, Rice University, Houston, TX. His research interests are in the areas of quality-of-service, scheduling, admission control, and media access protocols in wireless and wireline networks.

Dr. Knightly is an Associate Editor of the *Computer Networks Journal*, the IEEE/ACM TRANSACTIONS ON NETWORKING, IEEE TRANSACTIONS ON MULTIMEDIA, and previously, *IEEE Network Magazine*. He served as Co-chair of IWQoS 1998 and on the steering committee for IWQoS from 1999 to 2001. He served as Finance Chair for ACM MOBICOM 2002 and 2003, Tutorial Co-chair for IEEE ICNP 2001 and MOBIHOC 2003, and on the program committee for numerous networking conferences including ICNP, INFOCOM, IWQoS, MOBICOM, and SIGMETRICS. He received the National Science Foundation CAREER Award in 1997 and the Sloan Fellowship Award in 2001.