Providing Deterministic End-to-end Fairness Guarantees in Core-stateless Networks[1]

Jasleen Kaur
Department of Computer Science
University of North Carolina at Chapel Hill

Harrick Vin
Department of Computer Sciences
University of Texas at Austin

Abstract End-to-end fairness guarantee is an important service semantics that network providers would like to offer to their customers. A network provider can offer such service semantics by deploying a network where each router employs a fair packet scheduling algorithm. Unfortunately, these scheduling algorithms require every router to maintain per-flow state and perform per-packet flow classification; these requirements limit the scalability of the routers. In this paper, we propose the *Core-stateless Guaranteed Fair (CSGF)* network architecture—the first work-conserving architecture that, without maintaining per-flow state or performing per-packet flow classification in core routers, provides to flows fairness guarantees similar to those provided by a network of core-stateful fair routers.

1 Introduction

With the commercialization of the Internet, there is a significant incentive for network service providers to offer value-added services to customers. An opportunity for adding value comes with the emergence of wide-area real-time and mission-critical applications; these applications benefit from network services that provide guarantees, on a per-flow basis, on the end-to-end delay, jitter, loss, and throughput. Over the past decade, several link scheduling algorithms that enable networks to provide such guarantees—by arbitrating access to shared link bandwidth at routers by packets of different flows—have been proposed [3, 4, 9–11, 19, 23, 24]. In this paper, we address the problem of designing a *work-conserving, core-stateless* network architecture that can provide *fairness* guarantees to flows. In what follows, we first justify the need for fairness as well as core-stateless and work-conserving network architectures, and then summarize our contributions.

Why Fairness? *Fairness* of bandwidth allocation is an important guarantee provided by link scheduling algorithms. Using a fair scheduling algorithm, routers provide throughput guarantees to backlogged flows at short time-scales (independent of their past usage of link bandwidth), and allocate idle link capacity to competing flows in proportion to their weights (or reserved rates).

The property of providing throughput guarantees at short time-scales independent of the past bandwidth usage by the flow is important for two reasons. First, in many applications, sources may not be able to predict precisely their bandwidth requirements at short time-scales (consider, for instance, the problem of transmitting variable bit-rate

[1] This work appeared at the Eleventh International Workshop on Quality of Service (IWQoS'03), Monterey, CA, June 2003.

encoded live video stream). To support these applications effectively, a network should allow flows to utilize occasionally more than their reserved bandwidth if such over-usage does not come at the expense of violating the bandwidth guarantees provided to other flows. Second, it is in the best interest of a network to allow sources to transmit data in bursts; bursty transmissions allow a network to benefit from statistical multiplexing of the available network bandwidth among competing traffic. If a network were to penalize a flow for using idle bandwidth, then the source would have no incentive to transmit bursts into the network; this, in turn, would reduce the statistical multiplexing gains and thereby reduce the overall utilization of network resources.

The property of fair scheduling algorithms of allocating available bandwidth to flows in proportion to their reserved rates is desirable from an economic perspective. Consider, for instance, the case when a network provider charges its customers based on their reserved bandwidth. In such a network, if a user A pays twice as much as user B, then A expects the network to allocate bandwidth in the ratio 2:1 to users A and B; any other allocation would be considered unfair. Fair scheduling algorithms allow a network to ensure this proportionate allocation property independent of the amount of available bandwidth.

Why Core-stateless? The design of next-generation networks is faced with the challenge that link capacities and traffic demands are increasing rapidly [8, 12], whereas processor speeds are increasing at only about half the rate [1]. This implies that the per-packet processing and computation performed by next-generation routers must be simplified to enable them to operate at high link speeds. Link scheduling algorithms proposed in the last decade to enable networks to provide fairness guarantees to flows [3, 4, 9–11, 19], on the other hand, require routers to maintain and use per-flow state, and perform packet classification to identify the flow to which an incoming packet belongs. The complexity of these operations grows as the number of flows increase. Thus, routers in such fair networks may not be able to operate at high link speeds, especially routers in the core of the network that aggregate a large number of flows originating from different edges of the network.

In order to alleviate this issue, over the past few years, several core-stateless networks have been designed to provide end-to-end service guarantees without maintaining or using any per-flow state at the core routers of a network [6, 7, 13, 16, 18, 20, 21, 25]; this property improves the scalability of the core routers to large number of flows and high-speed links. Existing proposals for providing fairness in core-stateless networks, however, only provide *approximate* fairness in the end-to-end throughput achieved by flows over large time-scales [6, 7, 18, 20]. In particular, due to the statistical nature of these guarantees, such schemes can not provide fairness guarantees to short-lived flows or for specific durations of interest in the lifespan of long-lived flows.

Why Work-conserving? Throughput and proportionate allocation guarantees can be ensured in networks that are *non work-conserving*, in which flows are allocated no more than their reserved rates at any time. In fact, the only known core-stateless networks that guarantee deterministically, that flows would receive throughput in proportion to their reserved rates, are non work-conserving [21]. Non work-conserving networks shape the rate of incoming traffic to a maximum of the reserved rate for a flow; sources are not allowed to achieve larger transmission rates, even if network bandwidth is idle. This property results in high average delays [22] and limits the ability of the network to utilize resource efficiently. With the predicted growth in traffic demands [12], this is undesirable. To the best of our knowledge, the only kind of deterministic guarantees provided by work-conserving core-stateless networks proposed in the literature, are on delay and throughput [13, 16, 17, 25].

Research Contributions In this paper, we propose the *first* work-conserving core-stateless network that provides deterministic fairness guarantees. We argue that an end-to-end notion of proportionate allocation in fair networks is meaningful only when defined across flows that share the same end-to-end network path. We show that networks that provide throughput guarantees are a crucial building block for networks that provide proportionate allocation guarantees. We then use a set of two simple mechanisms—namely, aggregation and fair-ingress—to enable a core-stateless network that provides throughput guarantees (previously proposed) to also guarantee proportionate allocation. We show that the resultant network, referred to as a Core-stateless Guaranteed Fair (CSGF) network, provides deterministic fairness guarantees similar to those provided by core-stateful networks.

The rest of this paper is organized as follows. In Section 3, we formulate the problem of end-to-end fairness. In Section 4, we present the key insights and mechanisms used to design CSGF networks. We derive properties of CSGF networks in Section 4.2. Deployment considerations are discussed in Section 6. We summarize related work in Section 7 and our conclusions in Section 8.

2 Notation and Assumptions

Throughout this paper, we use the following symbols and notation.

p_f^k	: the k^{th} packet of flow f
$a_{f,j}^k$	: arrival time of p_f^k at node j on its path
$d_{f,j}^k$	: departure time of p_f^k from node j
l_f^k	: length of packet p_f^k
r_f	: rate reserved for flow f
π_j	: upper bound on propagation delay of the link connecting node j and $(j+1)$
C_j	: outgoing link capacity at node j
$W_{f,j}(t_1, t_2)$	: throughput of flow f at server j during $[t_1, t_2]$
$\gamma_{f,j},\ U_{j,\{f,m\}},\ I_{j,m,f}$	: constants associated with service guarantees

H denotes the number of routers along the path of flow f. The source of flow f is connected to router 1 and the destination is connected to router H. A source is said to transmit packets *at least at its reserved rate* r_f, if $a_{f,1}^k \leq a_{f,1}^{k-1} + \frac{l_f^{k-1}}{r_f}$. The k^{th} packet, p_f^k, transmitted from the source, is said to have a *sequence number* of k. The *throughput*, $W_{f,j}(t_1, t_2)$, received by flow f at server j during a time interval $[t_1, t_2]$, is defined as the number of bits of flow f that depart server j during the time interval $[t_1, t_2]$. Also, a flow f is said to be *continuously backlogged* at server j in a time interval $[t_1, t_2]$ if, at all instances within this interval, there is at least one packet belonging to flow f in the server queue. Throughout our analysis, we use the terms *server* and *router* interchangeably; further, we assume that the sum of rates reserved for flows at any server does not exceed the server capacity (i.e., the link bandwidth).

3 Problem Formulation

Background Networks can provide fairness guarantees by employing *fair scheduling algorithms* at all routers [3, 4, 9–11, 19]. The fairness guarantee provided by fair scheduling algorithms at a *single* node can be formalized[2] as follows:

Definition 1. *The scheduling algorithm at node j is said to provide a fairness guarantee if in any time interval $[t_1, t_2]$ during which two flows f and m are continuously backlogged, the number of bits of flows f and m transmitted by the server, $W_{f,j}(t_1, t_2)$ and $W_{m,j}(t_1, t_2)$ respectively, satisfy:*

$$\left| \frac{W_{f,j}(t_1, t_2)}{r_f} - \frac{W_{m,j}(t_1, t_2)}{r_m} \right| \le U_{j,\{f,m\}} \tag{1}$$

where r_f and r_m are the rates reserved for flows f and m respectively, and $U_{j,\{f,m\}}$ is an unfairness measure—a constant that depends on the scheduling algorithm and traffic characteristics at server j. Further, if the sum of reserved rates of all flows at node j does not exceed the outgoing link capacity, then in any time interval $[t_1, t_2]$, during which the source of a flow f transmits packets at least at its reserved rate r_f, the server guarantees a minimum throughput to flow f:

$$W_{f,j}(t_1, t_2) > r_f(t_2 - t_1) - r_f \gamma_{f,j} \tag{2}$$

where $\gamma_{f,j}$ is an error term — a constant that also depends on traffic and server characteristics.

Fair scheduling algorithms are capable of providing a proportionate allocation guarantee slightly stronger than given in (1): if flow m is continuously backlogged during $[t_1, t_2]$, then the throughput of any other flow f, whether backlogged or not, is given by [15]:

$$\frac{W_{f,j}(t_1, t_2)}{r_f} \le \frac{W_{m,j}(t_1, t_2)}{r_m} + I_{j,m,f} \tag{3}$$

where $I_{j,m,f}$ is a constant that also depends on the server and traffic characteristics. Different fair scheduling algorithms differ in the values of $U_{j,\{f,m\}}$, $I_{j,m,f}$, and $\gamma_{f,j}$ [3, 4, 10, 11, 19]. The smaller is the value of these constants, the better is the corresponding throughput or proportionate allocation guarantee. Table 1 lists known values for several algorithms.

The literature contains analyses that extend these single-server guarantees on throughput and proportionate allocation to *end-to-end* guarantees provided by a network of fair scheduling servers [2, 14, 15]. Specifically, a network of fair servers (1) guarantees a minimum end-to-end throughput to flows with an associated error term, $\gamma^{net}_{f,H}$, and

[2] Fairness of a link scheduling algorithm can be defined equivalently in terms of a bound on its deviation from an idealized fluid model of fairness [3]. In terms of describing network properties perceivable by end-users, however, Definition 1 is more useful.

Table 1. Unfairness measures for some fair algorithms

	$U_{j,\{f,m\}}$	$I_{j,m,f}$	$\gamma_{f,j}$
GPS	0	0	0
SCFQ	$\frac{l_f^{max}}{r_f} + \frac{l_m^{max}}{r_m}$	$\frac{l_f^{max}}{r_f} + \frac{l_m^{max}}{r_m}$	-
SFQ	$\frac{l_f^{max}}{r_f} + \frac{l_m^{max}}{r_m}$	$\frac{l_f^{max}}{r_f} + \frac{l_m^{max}}{r_m}$	$\sum_{n \in F-\{f\}} \frac{l_n^{max}}{C_j} + \frac{l_f^{max}}{r_f}$
WF^{2}Q	$l^{max}(\frac{1}{r_f} + \frac{1}{r_m} - \frac{1}{C})$	$\frac{l^{max}}{r_m} + l_f^{max}(\frac{1}{r_f} - \frac{1}{C})$	$\frac{l_f^{max}}{r_f} + \frac{(l^{max} - l_f^{max})}{C_j}$

(2) guarantees proportionate allocation of end-to-end throughput to flows that share the same path, with an unfairness measure, $U^{net}_{H,\{f,m\}}$, where these constants are given by [2, 14, 15]:

$$\gamma^{net}_{f,H} = (H+1)\frac{l_f^{max}}{r_f} + \sum_{j=1}^{H-1} \pi_j + \sum_{j=1}^{H} \gamma_{f,j} \tag{4}$$

$$U^{net}_{H,\{f,m\}} = U_{1,\{f,m\}} + \sum_{h=2}^{H} (I_{h,f,m} + I_{h,m,f}) \tag{5}$$

Unfortunately, to provide fairness guarantees, fair scheduling algorithms require routers to maintain per-flow state. It is especially challenging to design networks that provide fairness without maintaining per-flow state in core routers because unlike delay guarantees (that can be characterized entirely in terms of the intrinsic properties of a flow such as its reserved rate), a fairness guarantee is inherently a function of the state (throughput) of all other flows sharing a resource (Definition 1). Prior attempts at designing core-stateless fair networks have realized this constraint; hence, prior designs of core-stateless fair networks provide only *statistical* (or approximate) fairness over large time-scales and for long-lived flows [6, 7, 18, 20]. In this paper, we attempt to design a core-stateless network architecture that can provide deterministic end-to-end fairness guarantees to flows.

Our Approach To derive a work-conserving, core-stateless network architecture that can provide deterministic fairness guarantees, we observe that a core-stateful network of fair routers provide to flows two types of guarantees: (1) throughput guarantee, and (2) a per-link proportionate bandwidth allocation guarantee.

Recently, we have proposed CSGT—a Core-stateless Guaranteed Throughput network architecture—that can provide end-to-end throughput guarantees to flows [16]. CSGT can provide, at all time-scales, a throughput guarantee that is within a constant of what is provided by a core-stateful network of fair scheduling routers. Thus, CSGT provides part of the functionality offered by a core-stateful network of fair routers. Unfortunately, since core routers in CSGT networks do not maintain any per-flow state, they can not ensure per-link proportionate allocation of bandwidth to flows.

In what follows, we argue that the per-link proportionate bandwidth allocation offered by fair scheduling algorithms translates to meaningful end-to-end guarantees only when flows share the *entire* end-to-end paths. To support this argument, observe that the end-to-end bandwidth allocated to a flow depends on the flow's share of the bottleneck[3] link bandwidth. Flows that share the entire end-to-end network path also share the bottleneck link; hence, the allocation of bandwidth on the bottleneck link governs the relative end-to-end bandwidth allocation to these flows. On the other hand, flows that do not share the entire end-to-end network paths may not share the bottleneck link. Further, since the bottleneck link for each flow may change continuously with fluctuations in traffic conditions, even when the bottleneck link is shared between such flows, the sharing is likely to be transient (or short-lived).[4] Hence, in networks where each router employs a fair scheduling algorithm to allocate spare bandwidth proportionally among competing flows on a per-link basis, it is difficult to relate, in a consistent manner, the end-to-end bandwidth allocated to two flows that do not share the complete end-to-end path. Consequently, core-stateful networks can not provide any strong consistent *guarantees* with respect to the relative bandwidth allocated to flows that do not share complete path.

From the above arguments, we conclude that, from the perspective of a network provider, an architecture that only supports *end-to-end* proportionate bandwidth allocation (a weaker guarantee) is likely to be indistinguishable from a core-stateful fair network architecture that supports proportionate allocation on a per-link basis. Hence, in this paper, we explore the design of networks that can provide end-to-end throughput guarantees and proportionate allocation guarantees to flows that share the same *end-to-end* path. Our design proceeds in two steps. First, we show that a network that provides throughput guarantees is a crucial building block for designing one that provides fairness guarantees. Second, we explore mechanisms that, when integrated with a work-conserving core-stateless network that provides throughput guarantees, lead to the design of the *Core-stateless Guaranteed Fair* (CSGF) network architecture—the *first* work-conserving core-stateless architecture that provides deterministic fairness guarantees.

4 The Design of Core-stateless Guaranteed Fair Networks

4.1 Providing Fairness Guarantees: Key Insights

Our objective is to design a *work-conserving, core-stateless* network architecture that can provide *fairness* guarantees to flows. Specifically, we want to provide two types of guarantees: (1) an end-to-end throughput guarantee to each flow, and (2) a proportionate bandwidth allocation guarantee to flows that share the same end-to-end path. In what follows, we show that by *decoupling* the objectives of providing these two guarantees,

[3] In this paper, the bottleneck link for a flow refers to the link with the least share of *available bandwidth* for the flow, rather than the link with the least link capacity. Depending on the cross traffic load, the link with the least share of available bandwidth for a flow may be different from the link with the least link capacity.

[4] For some flows, such as those that originate from a common source behind a slow modem line, the access link may be the non-transient bottleneck link due to its limited capacity. However, such links lie outside the scope of the network provider architectures we consider in this paper. The edge and core routers that we consider belong to the provider's network, which does not include slow access links.

and by using the following two observations, we can design core-stateless networks that provide both kinds of guarantees.

Observation 1 As the following theorem indicates, providing a throughput guarantee is necessary for providing a proportionate allocation guarantee.

Theorem 1. *A work-conserving server that provides proportionate allocation guarantees to a continuously-backlogged flow m of the form:*[5]

$$\frac{W_{f,j}(t_1,t_2)}{r_f} \le \frac{W_{m,j}(t_1,t_2)}{r_m} + I_{j,m,f} \tag{6}$$

where f is any other flow, also provides to flow m a throughput guarantee of the form:

$$W_{m,j}(t_1,t_2) \ge r_m(t_2 - t_1) - r_m * \gamma_{m,j}$$

where $\gamma_{m,j} = \frac{l^{max}}{C_j} + \frac{\sum_{f \in F} r_f I_{j,m,f}}{C_j}$, and C_j is the total capacity of the server.

Proof: See Appendix A. □

The converse of the above theorem indicates that *a server that does not provide throughput guarantees can not provide proportionate allocation guarantees.* A core-stateless network that provides throughput guarantees is, therefore, a crucial building block for the design of one that provides proportionate allocation.

Observation 2 A network that is capable of providing throughput guarantees, can additionally provide end-to-end proportionate bandwidth allocation to flows that share the same path, by employing a set of three mechanisms:

1. Treat the aggregate traffic between a pair of edge nodes as a single flow and provide throughput guarantees to it,
2. Employ a fair scheduling algorithm at the *ingress edge node*, that allocates a proportionate share of the aggregate throughput (at the ingress) to individual flows within the aggregate, and
3. Ensure that the network preserves the order in which packets are transmitted within the aggregate at the ingress node.

The third mechanism implies that the sequence in which packets depart at the last node in any time interval $[t_1, t_2]$—and hence the *relative* number of packets of two flows that depart in this time interval—can be equated to the sequence of packet departures at the ingress node in some other time interval $[t_1', t_2']$. The end-to-end proportionate allocation guarantee provided by the network is, therefore, exactly the same as the one provided by the ingress server. The second mechanism ensures that the ingress server does provide such a guarantee. The first mechanism ensures that the aggregate traffic on the end-to-end path is guaranteed a minimum throughput; since the individual flows are allocated a proportionate share of this aggregate throughput, it follows that individual flows are provided minimum throughput guarantees as well. A network that employs the above three mechanisms, therefore, provides throughput as well as proportionate allocation guarantees to individual flows.

[5] Note that this notion of proportionate allocation is slightly different from that in Definition 1, which requires *both* flows to be backlogged. It can be shown that fair scheduling algorithms provide this stronger notion of proportionate allocation as well [15].

Note that any core-stateless network that employs the three mechanisms described above can provide fairness guarantees. Below, we present one specific instantiation of such a network, called the Core-stateless Guaranteed Fair (CSGF) network.

4.2 Realization: A CSGF Network

As discussed above, a core-stateless network that provides throughput guarantees is a crucial building block for the design of one that provides fairness guarantees. In [16], we have proposed the Core-stateless Guaranteed Throughput (CSGT) network architecture, a work-conserving core-stateless architecture that enables a network to provide throughput guarantees. We briefly describe this architecture below.

A CSGT Network

A number of work-conserving core-stateless networks that provide delay guarantees have been proposed in the literature [13, 17, 25]. These networks, however, do not provide throughput guarantees to flows at short time-scales. This is a consequence of a central property of these networks to let packet deadlines grow ahead of current time for flows that use idle bandwidth to transmit packets at more than their reserved rates. Such flows may be penalized during a subsequent time interval by being denied throughput at even their reserved rate. To avoid this, a CSGT network *re-uses* deadlines of packets that depart the network much earlier than their deadlines, for new packets within the same flow. Formally, a CSGT network is defined as follows [16].

The Definition of a CSGT Network A CSGT network consists of two types of routers: *edge routers* and *core routers* (see Figure 1). The ingress edge router, in addition to maintaining per-flow state, maintains a sorted-list $\mathcal{R}$ of re-usable deadline vectors. On receiving a packet p_f^k of flow f, the ingress router assigns to it a *deadline vector*[6] $[F_1(p_f^k), F_2(p_f^k), ..., F_H(p_f^k)]$ where H is the number of servers along the path of flow f, and $F_j(p_f^k)$ is the deadline of packet p_f^k at server j. The assignment of the deadline vector to packet p_f^k proceeds as follows: If $\mathcal{R} \neq \emptyset$, an incoming packet is assigned the smallest deadline vector from $\mathcal{R}$. Otherwise, a new deadline vector is created as follows:

$$F_1(p_f^k) = \max\left(a_{f,1}^k, \widehat{F}_f(a_{f,1}^k)\right) + \frac{l_f^k}{r_f} \tag{7}$$

$$F_j(p_f^k) = F_1(p_f^k) + \sum_{h=1}^{j-1}(\beta_{f,h} + \pi_h) + (j-1)\max_{1 \leq i \leq k} \frac{l_f^i}{r_f}\ , \ j > 1 \tag{8}$$

where $\beta_{f,h} = \frac{l_h^{max}}{C_h}$, π_h is the propagation latency on the link connecting node j and $j+1$, and $\widehat{F}_f(t)$ is the maximum deadline at server 1 assigned to any packet of flow f by time t. All servers in the CSGT network transmit packets in the increasing order of their deadlines at that server. The egress server notifies the ingress server, using acknowledgment packets, whenever packets depart much earlier than their deadlines.

[6] In practice, the ingress router computes only $F_1(p_f^k)$—the other values are computed at the respective routers by adding appropriate constants to $F_1(p_f^k)$.

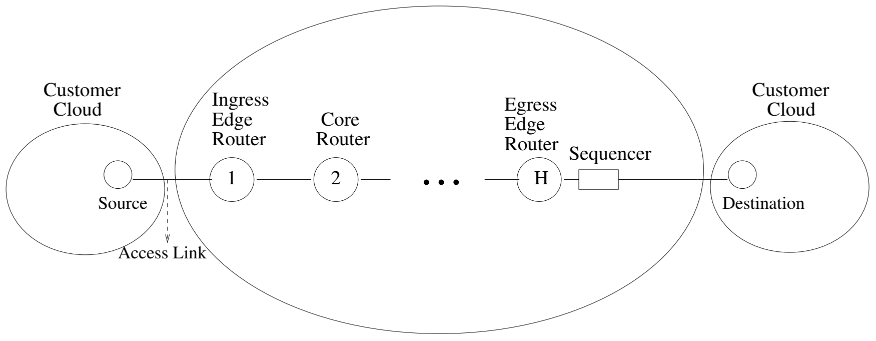


Fig. 1. The CSGT Network Architecture

When deadlines get re-used as described above, packets of a flow f may depart the egress router out-of-order. This is because packets transmitted later by the source may be assigned smaller deadlines than packets transmitted earlier, and may overtake the latter inside the network. To provide in-order delivery semantics to applications, a CSGT network employs an entity, referred to as the *sequencer*, to buffer packets of a flow that arrive out-of-sequence at the egress router, and restore packet order before delivering them to the destination. In order to bound the buffer space occupancy at the sequencer, the maximum number of deadlines simultaneously in use by packets of a given flow is maintained within an upper bound, W.

The following theorem from [16] derives the throughput guarantee provided by a CSGT network.

Theorem 2. *If the source of flow f transmits packets at least at its reserved rate, and D^{max} is an upper bound on the latency after which an acknowledgment packet transmitted by the egress node reaches the ingress node, then a CSGT network guarantees a minimum throughput in any time interval $[t_1, t_2]$, $W_{f,H}(t_1, t_2)$, given by:*

$$\begin{aligned} W_{f,H}(t_1,t_2) > r_f(t_2-t_1) - r_f * D^{max} - W * l_f \\ -r_f\left((H+1)\frac{l_f}{r_f} + \sum_{j=1}^{H-1}\pi_j + \sum_{j=1}^{H}\beta_{f,j}\right) \end{aligned}$$

where $\beta_{f,j}$ is a constant that depends on the server and traffic characteristics at node j.

Recall that fair networks provide two kinds of guarantees: a minimum throughput guarantee at the reserved rate and a proportionate allocation guarantee. Theorem 1 indicates that networks that guarantee proportionate allocation, also provide throughput guarantees. A network that provides throughput guarantees, however, need not guarantee proportionate throughput allocation to different flows. For instance, a network may provide throughput exactly at the reserved rate to one flow, but may allow another flow to use significantly more than its reserved rate. In fact, it can be shown, through examples, that a CSGT network does not guarantee proportionate throughput allocation. Appendix B presents one such example.

A CSGF Network

We use the set of three mechanisms described in Section 4.1 (Observation 2), in conjunction with a CSGT architecture that provides throughput guarantees, to design a core-stateless network architecture that provides fairness guarantees. Observe that a CSGT network already has the third mechanism, namely in-order delivery of packets, in place — the role of a *sequencer* is precisely to restore the correct packet order before packets depart the network. We instantiate the first two mechanisms in a CSGT network to derive a new architecture—referred to as a *Core-stateless Guaranteed Fair (CSGF)* network—which is defined below. Figure 2 depicts the scheduling framework deployed at the ingress router of a CSGF network.

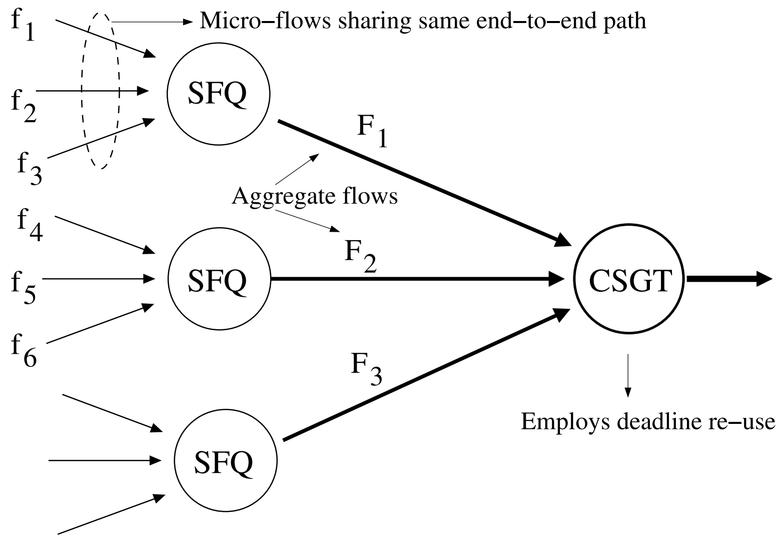


Fig. 2. Scheduling in a CSGF Ingress Router

Definition of a CSGF network: The ingress router for F, a set of flows sharing the same end-to-end path in a CSGF network, has two logical components:

- **Deadline Assignment:** A packet that belongs to an "aggregate" flow F is assigned a tag-vector exactly as in a CSGT network; new tag-vectors are computed using a reserved rate of $R = \sum_{f \in F} r_f$.
- **Packet Selection:** The next packet to be assigned a deadline within an aggregate flow F, is selected according to a fair schedule of transmission across individual flows in F. Since the bandwidth available to the aggregate F can fluctuate over time due to variations in cross-traffic, it is desirable to use a scheduler that achieves fair allocation even with fluctuating capacity. We use the Start-time Fair Queuing

(SFQ) [11] scheduler, which has this property, to determine the next flow to select a packet from.[7]

The core routers and the egress router in a CSGF network function in the same manner as in a CSGT network. At the egress, a sequencer re-orders packets within the "aggregate" before they are split into micro-flows.

Properties of a CSGF network

We now formally derive the proportionate allocation and throughput guarantees provided to individual flows by a CSGF network. We assume that all flows between the same pair of edge routers transmit packets of the same size l.[8]

Proportionate Allocation Guarantees Our objective is to compute: $\left(\frac{W_{f,H}(t_1,t_2)}{r_f} - \frac{W_{m,H}(t_1,t_2)}{r_m}\right)$, the difference in the normalized number of bits of two backlogged flows f and m, that depart the sequencer during a time interval $[t_1, t_2]$.

Let F be the aggregate flow containing packets of micro-flows f and m. Let p_1 and p_2, respectively, be the first and last bits belonging to the aggregate F, that depart the sequencer during $[t_1, t_2]$. Then, since packets belonging to F depart the sequencer in the same order as their transmission from the SFQ server at the ingress node, all (and only) bits that are transmitted between p_1 and p_2 at the ingress SFQ server, depart the sequencer during $[t_1, t_2]$. If p_1 and p_2 are transmitted from the ingress SFQ server at t'_1 and t'_2, respectively, then the throughput of flow f and flow m at the sequencer during $[t_1, t_2]$ is exactly the same as their throughput at the ingress server during $[t'_1, t'_2]$. That is, $\frac{W_{f,H}(t_1,t_2)}{r_f} = \frac{W_{f,1}(t'_1,t'_2)}{r_f}$, and $\frac{W_{m,H}(t_1,t_2)}{r_m} = \frac{W_{m,1}(t'_1,t'_2)}{r_m}$. The unfairness measure guaranteed to flows for the end-to-end throughput during $[t_1, t_2]$ in a CSGF network is, therefore, *equal* to that of the SFQ server at the ingress node during $[t'_1, t'_2]$. For a backlogged flow m, it follows that:

$$\frac{W_{f,H}(t_1,t_2)}{r_f} \leq \frac{W_{m,H}(t_1,t_2)}{r_m} + U^{SFQ}_{f,m} \tag{9}$$

Throughput Guarantees Theorem 2, which states the throughput guarantees of a CSGT network, indicates that the *aggregate* traffic between the pair of edge routers in a CSGF network is guaranteed a minimum throughput characterized by $R = \sum_{f \in F} r_f$, the cumulative reserved rate. Since each flow gets a fair share of this throughput, it follows (using the same reasoning as used to prove Theorem 1), that each micro-flow is provided a throughput guarantee as well. We formally derive this guarantee for a backlogged flow m below.

[7] Note that *any* fair scheduling algorithm that guarantees proportionate allocation despite fluctuating capacity can be used at the ingress. SFQ has one of the best known unfairness measures, $U_{j,\{f,m\}}$, among such algorithms (see Table 1).

[8] If l^{max} is the maximum allowed packet size, it is reasonable to expect a source that is backlogged with data to transmit, to use packets of size l^{max}.

The throughput guarantee of a CSGT network derived in Theorem 2, when applied to a CSGF network, implies that

$$W_{F,H}(t_1,t_2) > R(t_2-t_1) - Wl - R\left((H+1)\frac{l}{R} + \sum_{j=1}^{H-1}\pi_j + \sum_{j=1}^{H}\beta_{f,j} + D^{max}\right) \quad (10)$$

Let $a = \frac{W_{m,H}(t_1,t_2)}{r_m}$. From (10) and (9), it follows that:

$$\begin{aligned} R(t_2-t_1) - W*l - R\left((H+1)\frac{l}{R} + \sum_{j=1}^{H-1}\pi_j + \sum_{j=1}^{H}\beta_{f,j} + D^{max}\right) \\ &\leq \sum_{f\in F} W_{f,H}(t_1,t_2) \\ &\leq \sum_{f\in F} a*r_f + r_f*U^{SFQ}_{f,m} \\ &\leq a*R + \sum_{f\in F} r_f(\frac{l}{r_m}+\frac{l}{r_f}) \\ &\leq a*R + \frac{l}{r_m}R + \sum_{f\in F} l \end{aligned}$$

This implies that a CSGF network provides a per-flow throughput guarantee given by:

$$\begin{aligned} W_{m,H}(t_1,t_2) \geq r_m(t_2-t_1) - \frac{r_m}{R}(W+1)l - \frac{r_m}{R}\sum_{f\in F} l \\ -r_m\left((H+1)\frac{l}{R} + \sum_{j=1}^{H-1}\pi_j + \sum_{j=1}^{H}\beta_{f,j} + D^{max}\right) \end{aligned} \quad (11)$$

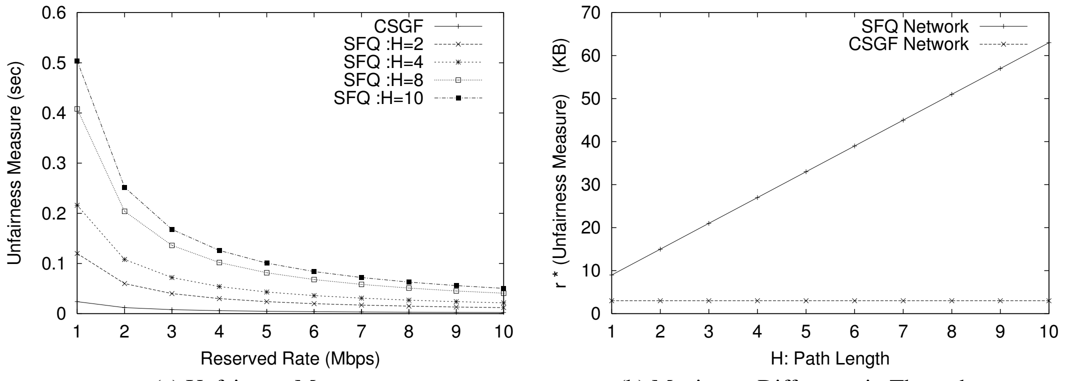
5 Evaluation of a CSGF network

The CSGF is the *first* work-conserving core-stateless network architecture that provides deterministic end-to-end throughput and proportionate allocation guarantees. We next address the question: *how do the end-to-end fairness guarantees of a CSGF network compare to those provided by core-stateful fair networks?* We answer these questions below by comparing the error terms ($\gamma^{net}_{f,H}$) and unfairness measures ($U^{net}_{H,\{f,m\}}$), associated with the end-to-end throughput and proportionate allocation guarantees respectively (Section 3), of CSGF and core-stateful networks. For our computations, we consider example network topologies in which link capacities are $100Mbps$ and the propagation latency on each link is $1ms$.

5.1 Proportionate Allocation Guarantees

Observe that the proportionate allocation guarantee in a CSGF network (Inequality (9)) is even better than that provided by a core-stateful network of SFQ servers (see (5)). The reason for this is that while packets of different flows depart the sequencer in a CSGF network exactly in the same order as transmitted by the fair ingress server, packets from different flows that share the same end-to-end path in a core-stateful network may not depart the network in the same order. The end-to-end fairness guarantee of a core-stateful network, therefore, can not be equated to that of its ingress server. We compute the unfairness measures provided by CSGF and a core-stateful networks of SFQ servers[9] for the example topology described above ($C = 100 Mbps$ and single-link propagation latency = $1ms$).

Observe that the difference in the unfairness measures provided by CSGF and a core-stateful network of SFQ servers is directly proportional to H, the path length, and inversely proportional to r_f, the reserved rates of the concerned flows. Figure 3(a) plots the unfairness measures provided by both network architectures, as a function of the reserved rates of individual flows (assuming $r_f = r_m$) and the path length (varied from 1 to 10). We observe that, for large-scale network topologies, the unfairness measure in a core-stateful architecture can be an order of magnitude higher than in a CSGF architecture.



(a) Unfairness Measure (b) Maximum Difference in Throughput

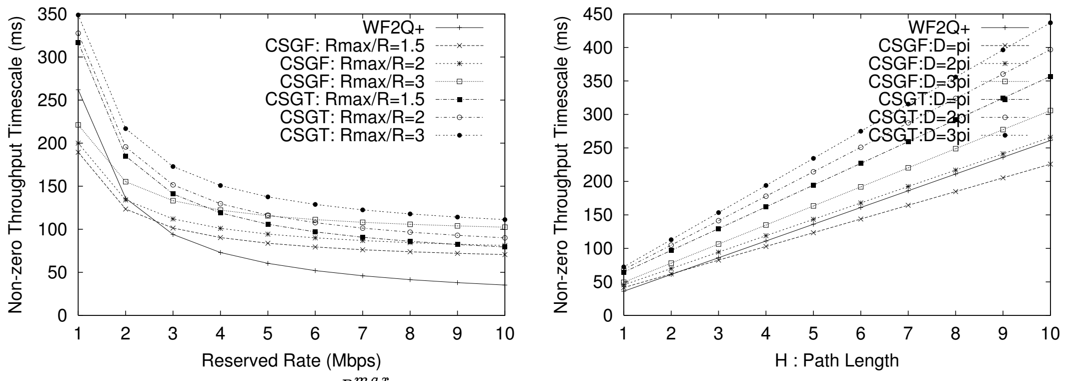
Fig. 3. Proportionate Allocation of Throughput in a CSGF Network

Figure 3(a) indicates that the throughput received in a CSGF network by two flows f and m, during any given time interval, may differ by an amount worth playing out for a few milli-seconds. To put this observation in perspective, we plot the reserved rate multiplied by the unfairness measure, in Figure 3(b). We observe that, during a given time interval, a CSGF network may deliver only a few kilo-bytes of more data for one flow, in comparison to other flows. In large-scale core-stateful networks, on the other hand, the difference in throughput could be of the order of tens of kilo-bytes.

[9] SFQ provides one of the smallest unfairness measures among known stateful fair scheduling algorithms.

5.2 Throughput Guarantees

We next compute and compare $\gamma_{f,H}^{net}$—the minimum timescale at which non-zero throughput is guaranteed to an application—in a CSGF , CSGT, and core-stateful network of WF^{2}Q+ [4] servers.[10] The smaller is the value of $\gamma_{f,H}^{net}$ for a network, the better is its throughput guarantee [16]. The difference in $\gamma_{f,H}^{net}$ for the three network architectures is governed mainly by the quantities (see (4), (11), and Theorem 2): r_f, the reserved rate of a flow, R, the aggregate reserved rate between a pair of edge routers, H, the number of hops in the path, D^{max}, the maximum latency experienced by feedback messages, and R^{max}, the maximum rate a flow in a CSGT or CSGF network is allowed to achieve.[11] We use the topology described initially ($C = 100Mbps$ and single-link propagation latency = $1ms$), to compute $\gamma_{f,H}^{net}$ for different settings of these quantities.



(a) Effect of r_f and $\frac{R^{max}}{R}$ (b) Effect of H and D^{max}

Fig. 4. Throughput Guarantee in a CSGF Network

Figure 4(a) plots for the three network architectures, the value of $\gamma_{f,H}^{net}$ as a function of r_f, the reserved rate for a flow, for different values of R^{max}/R, when the flow traverses a 10-hop path. We assume D^{max} is equal to the sum of link propagation latencies on the reverse network path. We observe the following:

1. For flows with small reserved rates, the per-flow throughput guarantee provided by a CSGF network may be better than that of a CSGT network. The reason for this can be understood by observing that $\gamma_{f,H}^{net}$ for a flow f is inversely related to r_f, its reserved rate (see (11) and Theorem 2). The throughput guarantee provided to the "aggregate" flow (larger reserved rate) in a CSGF network is, therefore, better than that provided to a micro-flow (smaller reserved rate) in a CSGT network. When

[10] A WF^{2}Q+ network guarantees the smallest $\gamma_{f,H}^{net}$ among known stateful fair scheduling algorithms. For networks with uniform packet sizes, the single-node throughput guarantee of a WF^{2}Q+ server is characterized by $\gamma_{f,j} = \frac{l}{r_f}$ [4].

[11] R^{max} governs the value of W, the maximum number of deadlines used simultaneously in CSGT and CSGF networks [16].

combined with the fact that the unfairness measure of a CSGF network is small, this implies that the throughput guarantee provided to a micro-flow in a CSGF network may be better than in a CSGT network. The difference between the two networks is less for flows with large bit-rates.
2. The throughput guarantee of a CSGF network may be better than that of a core-stateful network for flows with small reserved rates. This is because although $\gamma_{f,H}^{net}$ is inversely related to r_f for both CSGF and core-stateful networks, the inverse relation is much stronger for flows that traverse multi-hop paths in core-stateful networks (see (4) and (11)). For flows with large reserved rates, however, the additional terms in (11), as compared to (4), dominate the value of $\gamma_{f,H}^{net}$.

Figure 4(b) plots for the three network architectures, the value of $\gamma_{f,H}^{net}$ for a $1Mbps$ flow as a function of H, for D^{max} ranging from a multiple of 1 to 3 times the sum of link propagation latencies. Flows in the CSGT and CSGF networks are allowed to achieve a maximum rate of up to 3 times their reserved rates ($R^{max}/R = 3$). We observe the following:

1. $\gamma_{f,H}^{net}$ increases linearly with the number of hops traversed by a flow in all three network architectures. In CSGT and CSGF networks, $\gamma_{f,H}^{net}$ also increases with D^{max}; these networks, therefore, benefit from the provisioning of low-delay feedback channels between edge routers.
2. The throughput guarantee provided by a CSGF network is comparable to that provided by core-stateful networks (even when D^{max} is two times the sum of propagation latencies). Perhaps more importantly, we find that a CSGF network can guarantee non-zero throughput to flows, including flows with small reserved rates, at short time-scales of hundreds of milliseconds.

Our observations illustrate that a CSGF network is capable of providing throughput guarantees to individual flows at small time-scales. Large-scale CSGF networks provisioned with low-delay feedback channels may provide even better throughput guarantees to flows with small reserved rates, than core-stateful networks.

5.3 Discussion

Our observations in this section reveal that the end-to-end proportionate allocation guarantee of a CSGF network is better than that provided by core-stateful fair networks. Note that because a CSGF network does not maintain per-flow state in core routers, it, unlike core-stateful networks, does not guarantee the stronger notion of single-link (bottleneck) proportionate allocation. It may seem tempting to conclude that this stronger guarantee is useful to relate the throughput of flows that share their paths only *partially*. However, as argued in Section 3, two such flows may not share their bottleneck links, which govern the allocation of end-to-end throughput. Further, even if the flows share their bottleneck links, the sharing may be short-lived. Hence, it is difficult to relate, in a consistent manner, the end-to-end bandwidth attained by the two flows whose paths overlap only partially. Hence, we believe that our weaker notion of end-to-end proportionate allocation across flows that share the same *end-to-end* path, is adequate for the purposes of defining meaningful end-to-end service semantics.

We also find that it is possible to design CSGF networks that provide throughput guarantees similar to core-stateful networks by provisioning low-delay feedback channels between edge routers. However, the ability to provide comparable guarantees, without maintaining per-flow state in core routers, does not come for free. We discuss some of these issues in the next section.

6 Deployment Considerations

Use of a Sequencer A CSGF network uses a sequencer to provide in-order delivery of packets to applications. It has been shown in [16] that sequencer buffer requirements are modest even in large-scale networks.

It may seem that due to re-sequencing delays, end-to-end delay guarantees provided to flows in a CSGT or a CSGF network are weaker than those provided in core-stateful fair networks. This is, however, not the case, as is evident from the following two facts. First, despite deadline re-use, the transmission deadlines assigned to a given packet at routers in a CSGT or a CSGF network are never larger than the deadlines assigned to it without deadline re-use [13], or in a corresponding core-stateful network (we omit a detailed proof of this assertion). Second, a CSGT network provides deadline guarantees (see Lemma 2 of [16]). Together, these two facts imply that the delay guarantees provided in CSGT or CSGF networks are no worse than those provided by a core-stateful network of fair servers.

Feedback Channels The CSGF proportionate allocation guarantee does not depend on the delay or losses experienced by feedback messages on the reverse network path—it merely depends on the proportionate allocation guarantee of the ingress server. The throughput guarantee, on the other hand, depends on the maximum delay experienced in the feedback channels—the larger are the delays and losses on the reverse path, the weaker is the throughput guarantee. Note that delays or losses, though only those that occur on the forward path, can weaken the throughput guarantee of even core-stateful networks. Nevertheless, adequately provisioned feedback channels between the edge routers are an essential component of a CSGF or a CSGT architecture.

Overhead Due to Feedback Messages The transmission of feedback messages from the egress routers to the ingress routers in a CSGT or CSGF network raises some concern about the overhead introduced by this traffic. This overhead can be reduced for flows with bi-directional data transmission; the acknowledgments from the egress router can be *piggy-backed* on to data packets on the reverse path. In general, this overhead is a price paid for eliminating per-flow state and computation in the core of the network. It is, however, worthwhile to note that feedback messages are transmitted on the reverse path only when packets depart *much* earlier than their deadlines. This, in turn, happens only when sufficient idle bandwidth is available on the forward path. The feedback messages enable the efficient and fair use of such idle bandwidth on the forward path.

Complexity of Edge Routers Edge routers in a CSGF network are more complex than in core-stateful networks—the additional complexity is in terms of both extra per-aggregate state (set of reusable deadlines), as well as the use of two schedulers instead of one. Note, however, that the scheduler complexity is associated mainly with maintaining priority queues. In a CSGF network, priority queue operations incur a complexity of $O(\log N_A + \log N_F)$, where N_A is the number of aggregates and N_F is the average number of flows within an aggregate. In a corresponding core-stateful network, the priority queue complexity is $O(\log(N_A * N_F))$, which is the same as above. Therefore, processing complexity of ingress routers in CSGF networks is similar to those in core-stateful networks. Since edge routers are likely to process lower volumes of traffic, the extra state maintenance may not affect the performance of the network.

The egress edge router does not maintain any extra per-flow or per-aggregate state information; the sequencer does buffer packets that arrive out of sequence and maintains them in a sorted order of their sequence number. However, as mentioned before, the sequencer buffer space requirement is modest even in large-scale networks. This also implies that the size of the priority queue is bounded and small; the priority queue maintenance, therefore, does not introduce significant costs.

Admission Control In this paper, we have addressed the issue of providing fairness guarantees without maintaining per-flow state in the data path of core routers. Note, however, that we still need to ensure that the sum of reserved rates of flows at any router does not exceed its outgoing link capacity. One way to ensure this is to maintain and use per-flow state only in the control plane of core routers; since the control plane is accessed at a much lower frequency than the data plane, this may not affect the scalability and performance of the core routers. Recently, admission control frameworks have been proposed, that, instead of maintaining state at all routers, either use one or more *bandwidth brokers* or the edge routers to perform admission control [5, 26].

7 Related Work

The Core-Jitter Virtual Clock (CJVC) [21] network provides the same end-to-end delay guarantees as a corresponding core-stateful Jitter Virtual Clock network. A CJVC network, however, is non work-conserving, which limits network utilization and results in higher average delays, as discussed before. A number of work-conserving core-stateless networks that provide end-to-end delay guarantees similar to core-stateful networks have been proposed recently [13, 17, 25]. However, these networks do not provide throughput guarantees at short time-scales (they may penalize flows that use idle capacity to transmit at more than their reserved rates). The first core-stateless network that provides throughput guarantees at short time-scales has been proposed recently [16]. This architecture, however, does not guarantee proportionate allocation of throughput across flows sharing the same end-to-end path. Core-stateless schemes proposed to provide fairness, provide only approximate fairness in the long-term throughput achieved by flows. In particular, these schemes do not provide guarantees for short-lived flows or for specific durations of interest in the lifespan of long-lived flows [6, 7, 20].

8 Summary

In this paper, we present the Core-stateless Guaranteed Fair (CSGF) network architecture—the *first* work-conserving core-stateless architecture that provides deterministic end-to-end fairness guarantees at short time-scales. We decouple the throughput and proportionate allocation guarantees provided by a fair network and use a number of insights to develop a core-stateless network that provides both guarantees. First, we argue that an *end-to-end* notion of proportionate allocation is meaningful only when defined across flows that share the same end-to-end path. Second, we show that for a network to guarantee proportionate allocation, it must also provide throughput guarantees. Third, we show that a set of three mechanisms—fair access at the ingress, aggregation of micro-flows in the core, and re-sequencing at the egress—when used in conjunction with a network that provides throughput guarantees, leads to a network that guarantees proportionate allocation as well. We use these insights, and the previously proposed CSGT network architecture, to design a CSGF network that provides deterministic fairness guarantees at short time-scales. The end-to-end proportionate allocation guarantee of a CSGF network is better than that or a core-stateful fair network. The end-to-end throughput guarantees of CSGF networks provisioned with low-delay feedback channels are comparable to core-stateful networks, and may even be better for flows with small bit-rate requirements.

References

1. V. Agarwal, M.S. Hrishikesh, S.W. Keckler, and D.C. Burger. Clock Rate Versus IPC: The End of the Road for Conventional Microarchitectures. In *27th International Symposium on Computer Architecture (ISCA)*, June 2000.
2. J.C.R. Bennett, K. Benson, A. Charny, W.F.Courtney, and J.Y. LeBoudec. Delay Jitter Bounds and Packet Scale Rate Guarantee for Expedited Forwarding. In *IEEE/ACM Transactions on Networking*, volume 10, pages 529–540, August 2002.
3. J.C.R. Bennett and H. Zhang. WF^{2}Q: Worst-case Fair Weighted Fair Queuing. In *Proceedings of INFOCOM'96*, pages 120–127, March 1996.
4. J.C.R. Bennett and H. Zhang. Hierarchical Packet Fair Queueing Algorithms. In *IEEE/ACM Transactions on Networking*, volume 5, pages 675–689, October 1997.
5. S. Bhatnagar and B.R. Badrinath. Distributed Admission Control to Support Guaranteed Services in Core-stateless Networks. In *Proceedings of IEEE INFOCOM*, April 2003.
6. Z. Cao, Z. Wang, and E. Zegura. Rainbow Fair Queueing: Fair Bandwidth Sharing Without Per-Flow State. In *Proceedings of IEEE INFOCOM*, March 2000.
7. A. Clerget and W. Dabbous. TUF: Tag-based Unified Fairness. In *Proceedings of IEEE INFOCOM*, April 2001.
8. K. Coffman and A. Odlyzko. The Size and Growth Rate of the Internet. March 2001. http://www.firstmoday.dk/issues/issue3_10/ coffman/.
9. A. Demers, S. Keshav, and S. Shenker. Analysis and Simulation of a Fair Queueing Algorithm. In *Proceedings of ACM SIGCOMM*, pages 1–12, September 1989.
10. S.J. Golestani. A Self-Clocked Fair Queueing Scheme for High Speed Applications. In *Proceedings of INFOCOM'94*, 1994.
11. P. Goyal, H. Vin, and H. Cheng. Start-time Fair Queuing: A Scheduling Algorithm for Integrated Services Packet Switching Networks. In *Proceedings of ACM SIGCOMM'96*, pages 157–168, August 1996.
12. P. Kaiser. A (R)evolutionary Technology Roadmap Beyond Today's OE Industry. *NSF Workshop on The Future Revolution in Optical Communications & Networking*, December 2000.
13. J. Kaur and H. Vin. Core-stateless Guaranteed Rate Scheduling Algorithms. In *Proceedings of IEEE INFOCOM*, volume 3, pages 1484–1492, April 2001.
14. J. Kaur and H. Vin. Core-stateless Guaranteed Throughput Networks. *Technical Report TR-01-47, Department of Computer Sciences, University of Texas at Austin*, November 2001.
15. J. Kaur and H. Vin. End-to-end Fairness Analysis of Fair Queuing Networks. In *Proceedings of the 23rd IEEE International Real-time Systems Symposium (RTSS)*, December 2002.
16. J. Kaur and H. Vin. Core-stateless Guaranteed Throughput Networks. In *Proceedings of IEEE INFOCOM*, volume 3, April 2003.

17. C. Li and E. Knightly. Coordinated Network Scheduling: A Framework for End-to-end Services. In *IEEE ICNP*, November 2000.
18. R. Pan, B. Prabhakar, and K. Psounis. CHOKE, A Stateless Active Queue Management Scheme for Approximating Fair Bandwidth Allocation. In *Proceedings of IEEE INFOCOM*, March 2000.
19. A.K. Parekh. *A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks*. PhD Thesis, Department of Electrical Engineering and Computer Science, MIT, 1992.
20. I. Stoica, S. Shenker, and H. Zhang. Core-Stateless Fair Queueing: Achieving Approximately Fair Bandwidth Allocations in High Speed Networks. In *Proceedings of ACM SIGCOMM*, September 1998.
21. I. Stoica and H. Zhang. Providing Guaranteed Services Without Per Flow Management. In *Proceedings of ACM SIGCOMM*, September 1999.
22. H. Zhang. Service Disciplines For Guaranteed Performance Service in Packet-Switching Networks. *Proceedings of the IEEE*, 83(10), October 1995.
23. H. Zhang and S. Keshav. Comparison of Rate-Based Service Disciplines. In *Proceedings of ACM SIGCOMM*, pages 113–121, August 1991.
24. L. Zhang. VirtualClock: A New Traffic Control Algorithm for Packet Switching Networks. In *Proceedings of ACM SIGCOMM'90*, pages 19–29, August 1990.
25. Z.L. Zhang, Z. Duan, and Y.T. Hou. Virtual Time Reference System: A Unifying Scheduling Framework for Scalable Support of Guarantees Services. *IEEE Journal on Selected Areas in Communication, Special Issue on Internet QoS*, December 2000.
26. Z.L. Zhang, Z. Duan, Y.T. Hou, and L. Gao. Decoupling QoS Control from Core Routers: A Novel Bandwidth Broker Arcitecture for Scalable Support of Guaranteed Services. In *Proceedings of ACM SIGCOMM, Sweden*, August 2000.

A Proof of Theorem 1

A work-conserving server that has at least one continuously backlogged flow in $[t_1, t_2]$ would satisfy: $\sum_{f \in F} W_{f,j}(t_1, t_2) \geq C_j(t_2 - t_1) - l^{max}$, where the l^{max} term appears due to packetization effects.

Consider a particular interval $[t_1, t_2]$. Let $a = \frac{W_{m,j}(t_2 - t_1)}{r_m}$. From (6), for any other flow f (whether continuously backlogged or not), we have: $W_{f,j}(t_2 - t_1) \leq r_f * a + r_f * I_{j,m,f}$. Since we assume that the sum of reserved rates at any server does not exceed its capacity (Section 2), we have:

$$\begin{aligned} C_j(t_2 - t_1) - l^{max} &\leq \sum_{f \in F} W_{f,j}(t_1, t_2) \\ &\leq a * \sum_{f \in F} r_f + \sum_{f \in F} r_f I_{j,m,f} \\ &\leq a * C_j + \sum_{f \in F} r_f I_{j,m,f} \end{aligned}$$

This implies: $a \geq (t_2 - t_1) - \frac{l^{max}}{C_j} - \frac{\sum_{f \in F} r_f I_{j,m,f}}{C_j}$. Therefore, $W_{m,j}(t_1, t_2) \geq r_m(t_2 - t_1) - r_m \gamma_{m,j}$, where $\gamma_{m,j} = \frac{l^{max}}{C_j} + \frac{\sum_{f \in F} r_f I_{j,m,f}}{C_j}$.

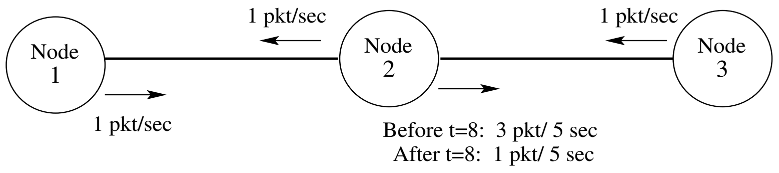


Fig. 5. Topology for the Example

B CSGT Networks Do not Guarantee Proportionate Allocation: An Example

At time $t = 0$, let flow f be created, and at time $t = 0.1$, let flow m be created. Let both flows traverse the 3-hop network depicted in Figure 5. Let the reserved rate of each of the flows at all nodes be 1 packet every 10 seconds. Let the CSGT parameter, W, for flows f and m be 2—this implies that by any time t, no packet (from either flow f or flow m) with a deadline greater than $t + 2 * 10$ has been transmitted from the first (ingress) node.

Let the transmission capacity on all outgoing links, except the link from node 2 to node 3, be 1 packet every second, and let there be no other flows sharing these links. Let the capacity available to flows f and m on the link from node 2 to node 3 (which is shared with other flows) be 3 packets every 5 seconds before $t = 8$, and 1 packet every 5 seconds after $t = 8$. Let the link propagation latencies be 0.

Since $W = 2$ for both flows, at $t = 1$, the first node transmits the following sequence of packets starting at $t = 0$: at $t = 1$, the first flow f packet with a deadline of 10; at $t = 2$, the first flow m packet with deadline of 10.1; at $t = 3$, the second flow f packet with deadline of 20; and at $t = 4$, the second flow m packet with deadline of 20.1.

The acknowledgment for the second flow f packet (deadline=20) arrives at the first node at $t = 9$, at which time, the first node *re-uses* it for the third flow f packet, which gets transmitted by $t = 10$. Further, since $W = 2$, the fourth flow f packet gets transmitted immediately after that, with a deadline of 30.

The acknowledgment for the second flow m packet does not arrive in time for deadline 20.1 to get re-used. The third blue packet is therefore assigned a deadline of 30.1, and gets transmitted after the fourth flow f packet, by $t = 12$. By $t = 12$, therefore, flow f has transmitted 4 packets, whereas flow m has transmitted only 3.

The acknowledgment for the fourth flow f packet (deadline=30) arrives at the first node at $t = 19$, at which time, the first node re-uses it for the fifth flow f packet, which gets transmitted by $t = 20$. Further, since $W = 2$, the sixth flow f packet gets transmitted immediately after that, with a deadline of 40.

The acknowledgment for the third flow m packet does not arrive in time for deadline 30.1 to get re-used. The fourth flow m packet is, therefore, assigned a deadline of 40.1 and is transmitted after the sixth flow f packet. By $t = 22$, therefore, flow f has transmitted 6 packets, whereas flow m has transmitted only 4.

It is easy to examine further time intervals to see that the difference between number of flow f packets transmitted and the number of flow m packets transmitted, grows with time. Specifically, the difference at time t is given by: $|(t-2)\ mod\ 10|$. It follows that the difference $\left|\frac{W_{f,j}(t_1,t_2)}{r_f} - \frac{W_{m,j}(t_1,t_2)}{r_m}\right|$ is not bounded, and grows approximately linearly with the length of the time interval $[t_1, t_2]$.