Core-stateless Guaranteed Throughput Networks*

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

Harrick M. Vin
Department of Computer Sciences
University of Texas at Austin

Abstract — End-to-end throughput 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 Throughput (CSGT)* 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 throughput guarantees that are within an additive constant of what is attained 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 export richer service semantics —with respect to end-to-end delay and throughput guarantees— to customers. Over the past decade, several packet scheduling algorithms that enable a network to offer such richer semantics have been proposed [2, 6, 9, 19]. A network can, for instance, provide end-to-end delay guarantees by employing Virtual Clock [20] and Delay Earliest-Due-Date (Delay EDD) [14, 18] scheduling algorithms; and delay-cum-throughput guarantees by employing fair packet scheduling algorithms —such as Weighted Fair Queuing(WFQ) [6, 12], Start-time Fair Queuing (SFQ) [9], and Self-clocked Fair Queuing (SCFQ) [7]— in routers. 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, especially routers in the core of the network that may carry a very large number of flows.

The topic of designing scalable network architectures that can export to flows[1] rich service semantics, but without maintaining per-flow state or performing per-packet flow classification in core routers, has received considerable attention in the recent past [10, 17, 21]. For instance, the Core-stateless Jitter Virtual Clock (CJVC) scheduling algorithm [17] and the class of Core-stateless Guaranteed Rate (CSGR) algorithms [10] enable a network to provide end-to-end delay guarantees similar to their core-stateful counterparts. CJVC also provides end-to-end throughput guarantees, but at the expense of making the network non-work-conserving. There have also been attempts at designing core-stateless versions of *fair* scheduling algorithms [4, 5, 13, 16]. Most of these attempts only provide *statistical* fairness at large time-scales; they do not provide any fairness or throughput guarantees for short-lived flows or for short intervals of interest in long-lived flows.

In this paper, we propose the Core-stateless Guaranteed Throughput (CSGT) network architecture—the *first* work-conserving network architecture that provides throughput guarantees to flows over finite time-scales, but without maintaining per-flow state in core routers. We develop the architecture in two steps. First, we show that for a network to provide end-to-end throughput guarantees, it must also provide end-to-end delay guarantees. Second, we demonstrate that two simple mechanisms —tag re-use and source rate control— when integrated with a work-conserving, core-stateless network that provides end-to-end delay guarantees, lead to the design of a CSGT network that provides end-to-end throughput bounds within an *additive constant* of that attained by a core-stateful network of fair servers.

The rest of this paper is organized as follows. In Section 3, we formulate the problem of providing end-to-end throughput guarantees. In Section 4, we present our CSGT network architecture, and derive bounds on the end-to-end throughput. Section 5 discusses deployment considerations. We summarize our contributions in Section 6.

2 Notation and Assumptions

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

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

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. 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). For improving readability, we include the proofs of all the Lemmas and Theorems in the appendix.

*This research was supported in part by grants from NSF (award ANI-0082294), Intel, IBM, and Cisco.

[1] We refer to a sequence of packets transmitted by a source as a *flow*.

3 Problem Formulation

A flow f with reserved rate r_f expects the network to provide, during any time interval, throughput at least at rate r_f. This service semantic is captured in the following definition of an end-to-end throughput guarantee:

Definition 1 *For a flow f, whose source transmits packets at least at its reserved rate r_f, a network is said to provide an end-to-end throughput guarantee if in any time interval $[t_1, t_2]$, the network guarantees a minimum throughput, $W_{f,H}(t_1, t_2)$, to flow f given by:*

$$W_{f,H}(t_1, t_2) > r_f(t_2 - t_1) - r_f\gamma_{f,H}^{net} \quad (1)$$

where H is the path length traversed by flow f and $\gamma_{f,H}^{net}$ is a constant that depends on the traffic and server characteristics at nodes along the path.

From the definition, a network guarantees a non-zero throughput to flow f if $t_2 - t_1 > \gamma_{f,H}^{net}$; thus, the value of $\gamma_{f,H}^{net}$ bounds the longest time interval for which a flow may receive no throughput from the network. Clearly, the smaller the value of $\gamma_{f,H}^{net}$, the better the quality of network service for applications that require sustained network throughput.

Observe that most networks that reserve a rate for each flow guarantee an *average* throughput at the reserved rate; however, these networks differ in the time-scales (namely, the value of $\gamma_{f,H}^{net}$) at which this guarantee is provided. For instance, for a network where each router employs an unfair packet scheduling algorithm (e.g., Virtual Clock or Delay EDD), the throughput received by a flow during a time interval is a function of the throughput received by the flow in the past. In fact, for such networks, $\gamma_{f,H}^{net}$ is not bounded, indicating that an unfair network cannot guarantee non-zero throughput at finite time scales. To provide throughput guarantees at short time-scales, networks employ fair packet scheduling algorithms at routers [3, 6, 7, 9]. Fair scheduling algorithms ensure that in any time interval in which two flows are backlogged, they receive service in proportion to their reserved rates. It can be shown that, when coupled with admission control, a network of fair servers provides an end-to-end throughput guarantee with $\gamma_{f,H}^{net} = (H+1)\frac{l_f^{max}}{r_f} + \sum_{j=1}^{H-1}\pi_j + \sum_{j=1}^{H}\gamma_{f,j}$, where $\gamma_{f,j}$ characterizes the throughput guarantee provided by fair server j in isolation [1, 11].

To provide throughput guarantees, existing fair scheduling algorithms define a concept of *virtual time* at each router; the virtual time at a router is inherently a function of the current state of each flow passing through the router. Implementations of fair scheduling algorithms, therefore, require every router to maintain per-flow state and perform per-packet flow classification. In this paper, our objective is to design a work-conserving core-stateless network architecture[2] that provides deterministic end-to-end throughput guarantees, similar to those provided by a network where each router employs a fair scheduling algorithm.

[2]Throughout this paper, the core-stateless property refers to a network that does not require its core routers to maintain per-flow state or perform per-packet flow classification.

4 CSGT Networks

We design core-stateless guaranteed throughput networks in two steps. First, we show that for a network to provide throughput bounds similar to those in fair networks, it *must* also provide end-to-end delay guarantees. Second, we demonstrate that two mechanisms —tag re-use and source rate control— when integrated with a core-stateless network that provides end-to-end delay guarantees, lead to the design of the Core-stateless Guaranteed Throughput (CSGT) networks that provide end-to-end throughput bounds within an *additive constant* of that attained by a core-stateful network of fair servers. Throughout this section, we assume that a source transmits equal-sized packets.

4.1 Need for Delay Guarantees

Let the *expected arrival time* of a packet p_f^k of flow f at the first server, $EAT(p_f^k)$, be defined as:

$$\begin{aligned} EAT(p_f^1) &= a_{f,1}^1 \\ EAT(p_f^k) &= \max\left(a_{f,1}^k, EAT(p_f^{k-1})\right) + \frac{l_f^{k-1}}{r_f},\ k > 1 \end{aligned}$$

Then, a network is said to provide an *end-to-end delay guarantee* to flow f, if, for all packets p_f^k, the network provides an upper bound on $(d_{f,H}^k - EAT(p_f^k))$, where $d_{f,H}^k$ is the departure time for packet p_f^k from node H. We use this definition of end-to-end delay guarantee to prove Theorem 1.

Theorem 1 *If a network provides lower-bounds on throughput of the form: $W_{f,H}(t_1, t_2) \geq r_f(t_2 - t_1) - \Phi$ to any flow f whose source transmits at least at its reserved rate, then it also provides to flow f an end-to-end delay guarantee of the form: $d_{f,H}^k - EAT_1(p_f^k) \leq \frac{\Phi + l_f^{max}}{r_f}$.*

The converse of Theorem 1 indicates that *a network that does not provide delay guarantee to packets cannot provide throughput bounds*. Hence, a work-conserving, core-stateless network that provides delay guarantee is a crucial building block for designing CSGT networks.

In [10], the authors propose Core-stateless Guaranteed Rate (CSGR) networks that provide the same delay guarantee as a network that employs stateful scheduling algorithms from the Guaranteed Rate (GR) class at the core routers. As we illustrate below, a work-conserving CSGR network, however, does not provide throughput guarantees at short time-scales. Our design of a CSGT network uses the CSGR network as a building block and enhances it with a set of end-to-end mechanisms that allow the network to retain its delay properties while providing throughput guarantees at short

time-scales. We describe the derivation of a CSGT network in the context of a Core-stateless Virtual Clock (CSVC) network [10, 21]—a specific instance of the class of CSGR networks.

4.2 Defining CSGT Networks

CSVC Networks A CSVC network consists of two types of routers: *edge routers* and *core routers*. Edge routers maintain per-flow state and perform per-packet flow classification. The *ingress* router —the edge router where a flow enters the network— assigns to each packet p_f^k a *service tag vector* $[VCore_{f,1}^k, VCore_{f,2}^k, ..., VCore_{f,H}^k]$ where H is the number of servers along the path, and $VCore_{f,j}^k$, the service tag value for server j, is derived as follows[3]:

$$VCore_{f,1}^k = \max\left(a_{f,1}^k, VCore_{f,1}^{k-1}\right) + \frac{l_f^k}{r_f}$$

$$VCore_{f,j}^k = VC_{f,1}^k + \sum_{h=1}^{j-1}(\beta_{f,h} + \pi_h + \max_{1\le i\le k}\frac{l_f^i}{r_f}),\ j \ge 1$$

where $VCore_{f,1}^0 = 0$ and $\beta_{f,h} = \frac{l_h^{max}}{C_h}$. At both the edge and core routers, packets are transmitted in the increasing order of their service tags. Note that the core routers only perform a sorting operation on the tag values; there is no need to maintain per-flow state or perform per-packet flow classification.

It has been shown that a CSVC network provides a *deadline guarantee* [10, 21]: packet p_f^k is guaranteed to depart server j by $(VCore_{f,j}^k + \beta_{f,j})$. However, the following example shows that a CSVC network does not provide throughput guarantees at finite time-scales.

Example 1 *Consider the first server, with a transmission capacity of* 10 *packets/sec, in a CSVC network. Let the sum of reserved rates of all flows be equal to the capacity. Let the rate reserved by a flow* f *be* 1 *packet/sec. At time* $t = 0$*, let* f *be the only backlogged flow. In this setting, by* $t = 1$*, 10 packets of flow* f *are serviced by the server; further,* $VCore_{f,1}^{11} = 11$*. Now, let all other flows become backlogged at time* $t = 1$*. Since the server services packets in the increasing order of virtual clock values, packet* p_f^{11} *may not be serviced until* $t = 10$*; hence, flow* f *receives no throughput during the interval* $[2, 10]$*. Given any time interval of arbitrary length, it is easy to extend this example to show that flow* f *receives no throughput during the interval of interest. Therefore, for any interval length, the CSVC server does not provide any non-trivial (non-zero) lower bound on throughput.*

In the above example, until time $t = 1$, because of the availability of idle bandwidth and the work-conserving nature of the CSVC server, flow f receives service at a rate greater than its reserved rate. Due to the way deadlines are computed, though, during the same period, flow f accumulates a *debit* at the rate r_f (indicated by the increase in its VCore value much beyond current time t), and is subsequently penalized for the duration of the accumulated debit once all the other flows become backlogged. It is important for networks to provide throughput guarantees at short time-scales, independent of the past usage of idle bandwidth by a flow, for two reasons:

1. In many settings, it is difficult for sources to predict precisely their bandwidth requirements at short time-scales. For instance, the bit-rate requirement of a variable bit-rate video stream may vary considerably and over short time-scales. Suppose a video stream, with a reserved bit-rate of $1Mbps$, transmits at $2Mbps$ for 4 seconds using idle bandwidth. In a network that does not provide throughput guarantees, the video stream may not receive any throughput at all in the next 4 seconds. The performance of the video application in such a network may therefore be unacceptable.
2. It is in the best interest of a network to allow sources to transmit data in transient bursts (i.e., at a rate greater than the reserved rate); bursty transmissions allow a network to benefit from statistical multiplexing of the available network bandwidth among competing traffic. In networks that penalize sources for using idle bandwidth, however, sources have no incentive to transmit bursts into the network. They may prefer to use constant bit-rate flows, instead of allowing the network to enforce arbitrary penalties. This, in turn, would reduce the statistical multiplexing gains and thereby reduce the overall utilization of network resources.

It is important to observe that while a CSVC network does not provide lower bounds on throughput at finite time-scales, it does guarantee an *average* throughput at the rate of r_f to a backlogged flow f over infinite time-scales. This implies that, the throughput of flow f in any interval $[t_1, t_2]$ would be below its reserved rate r_f only if the flow f receives service at a rate higher than r_f prior to t_1. In such an event, there must exist $t', t" < t_1$ such that during interval $[t', t"]$, packets of flow f arrive at the destination much ahead of their deadline guarantee (derived based on the reserved rate r_f). More formally, for packets p_f^k that reach destination at time t during the interval $[t', t"]$, $VCore_{f,H}^k \gg t$.

[3] In practice, the ingress router computes only $VCore_{f,1}^k$ for packet p_f^k. Before transmitting the packet, it encodes in the packet header a quantity called the *slack*—the difference between $VCore_{f,1}^k$ and the actual departure time—and $\max_{1\le i\le k}\frac{l_f^i}{r_f}$. When the second router receives the packet, it adds the slack, β_2, and $\max_{1\le i\le k}\frac{l_f^i}{r_f}$, to the arrival time of the packet to compute $VCore_{f,2}^k$ [10]. Subsequent routers compute their local $VCore$ values in a similar manner.

The Principle of Deadline Re-use The property of allowing a flow to accumulate arbitrarily large amount of debit —by increasing the deadline values (or service tag values) assigned to packets of the flow much beyond the current time— is central to the inability of CSVC networks to provide throughput guarantees at small time-scales. Hence, for a network to provide throughput bounds at small time-scales, it must reduce debit accumulation; this can be achieved by *allowing the ingress routers to re-use for future packets the deadline (or service tag) values of packets that reach the destination much prior to their deadlines*. This is the central concept in transforming a CSVC network into a CSGT network that provides throughput bounds.

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

$$F_1(p_f^k) = \max\left(a_{f,1}^k, \widehat{F}(a_{f,1}^k)\right) + \frac{l_f^k}{r_f} \quad (2)$$

$$F_j(p_f^k) = F_1(p_f^k) + \sum_{h=1}^{j-1}(\beta_{f,h} + \pi_h + \max_{1\leq i\leq k} \frac{l_f^i}{r_f}),\ j > 1 \quad (3)$$

where $\beta_{f,h} = \frac{l_h^{max}}{C_h}$, and $\widehat{F}(t)$ is the maximum of the service tags for server 1 assigned to any packet by time t. All servers in the CSGT network transmit packets in the increasing order of their service tags for that server.

Observe that if $\mathcal{R} = \emptyset$, then the assignment of tag vector in CSGT is identical to the CSVC network. When $\mathcal{R} \neq \emptyset$, then, by reusing a tag assigned to an earlier packet, CSGT prevents accumulation of unbounded debit for flow f. To instantiate such a CSGT network, we need to address the following issues.

1. When can an ingress server reuse a previously assigned tag vector for a new packet? What are the constraints that govern the re-usability of tag vectors? How does the ingress router create and maintain the sorted-list $\mathcal{R}$? We address these questions in Section 4.2.1.
2. With the reuse of previously assigned tag vectors in the CSGT network, packets of flow f with higher sequence number may, in fact, carry a smaller tag value (i.e., $F_h(p_f^j) < F_h(p_f^i)$ even if $i < j$). Since the tag values determine the priority for servicing packets in each router, it is quite possible that packet p_f^j may reach the egress edge router prior to packet p_f^i, even though packet p_f^i was transmitted prior to packet p_f^j at server 1. We discuss the associated packet re-ordering requirement in Section 4.2.2.

4.2.1 Maintaining the Sorted-list of Re-usable Tag Vectors

Reusing tag vectors allow CSGT networks to prevent unbounded debit accumulation for flows. Determination of whether a tag vector is eligible for reuse, however, is tricky because of two reasons.

- A CSGT network must ensure that the reuse of tag vectors for packets of flow f does not violate the deadline guarantees provided to *other* flows.

  To meet this requirement, the tag assigned to a packet p_f^k must differ by at least $\frac{l_f}{r_f}$ from the tags assigned to all packets p_f^i that were transmitted prior to packet p_f^k but have not reached the destination. This is because, if the separation is less than $\frac{l_f}{r_f}$, then flow f will be *guaranteed* service at a rate greater than its reserved rate r_f; this, in turn, could violate the deadline guarantees provided to other flows.
- A CSGT network must ensure that it can provide a deadline guarantee on the re-used tag vector.

  To meet this requirement, at the time of assigning a reusable tag to a packet, the ingress router must ensure that the tag value for the first server exceeds the current time by at least $\frac{l_f}{r_f}$.

Using these eligibility criteria, we formally define reusability of a tag vector as follows.

Definition 2 *A previously assigned tag vector* $[F_1, F_2, ..., F_H]$ *is said to be* **re-usable** *for a packet* p_f^m *at time* t *if it satisfies the following properties:*

$$\forall p_f^i \in \mathcal{U} : |F_j - F_j(p_f^i)| \geq \frac{l_f}{r_f} \quad (4)$$

$$t \leq F_1 - \frac{l_f}{r_f} \quad (5)$$

where $\mathcal{U}$ *is the set of packets transmitted by server 1 prior to packet* p_f^m *but have not reached the destination by time* t.

A CSGT network can enforce these conditions as follows.

1. An ingress router should consider a tag vector for re-use only after a packet carrying that tag vector departs the egress router H. This ensures that condition (4) is met. This can be achieved by requiring the egress router to send, on transmitting a packet p_f^m of flow f, an acknowledgment for that packet to the ingress router for flow f.

The ingress router, on receiving such an acknowledgment, can add the tag vector assigned to packet p_f^m to the sorted-list $\mathcal{R}$ of re-usable tag vectors for flow f.

2. On receiving a packet p_f^k from flow f at time t, the ingress router can scan through the sorted-list $\mathcal{R}$, discard all the tag vectors that violate condition (5), and assign to packet p_f^k the first re-usable tag that meets condition (5).

Observe that the tag vector assigned to a packet p_f^m is likely to re-usable (i.e., satisfy condition (5)) only if packet p_f^m departs server H "sufficiently" prior to its deadline. In particular, if D^{min} is the minimum latency incurred by the acknowledgment packet to reach the ingress router, then using (5), the tag vector of packet p_f^m can be re-used only if: $d_{f,H}^m + D^{min} \leq F_1(p_f^m) - \frac{l_f}{r_f}$. From (3), this is the same as:

$$d_{f,H}^m \leq F_H(p_f^m) - (\sum_{j=1}^{H-1}(\beta_{f,j} + \pi_j + \max_{1\leq i\leq m}\frac{l_f^i}{r_f}) + D^{min} + \frac{l_f}{r_f}) \quad (6)$$

Thus, the egress server sends an acknowledgment to the first server, *only* if packet p_f^k departs the network much before—as given by condition (6)—its deadline. We prove, in Lemma 2 (see Section 4.3), that if a CSGT network reuses tag vectors in accordance with the scheme described above, then it provides the same deadline-guarantee as a CSVC network.

4.2.2 Addressing Packet Re-ordering Requirements

With the tag re-use scheme described above, in a CSGT network, packets of flow f may reach the egress router out-of-order. For applications that desire in-order delivery semantics, a CSGT network needs to employ a *sequencer* that can buffer packets received out-of-order and then deliver to the applications packets in-order. A sequencer can reside either on the egress router, on a special network appliance located between the egress edge router and the destination node, or on the destination node itself[4]. Figure 1 depicts the setting where a sequencer is logically inserted between the egress router and the destination node. For the simplicity of analysis, we assume zero propagation delay between the egress router and the sequencer.

Now, let us consider the issues in designing the sequencer. The following example shows that, in a naive implementation of a CSGT network, the number of packets that may need to be buffered at the sequencer is not bounded.

[4]Deploying a sequencer on the destination node itself may require changes to end-hosts. Hence, the architectural options of instantiating the sequencer on the egress router or on an appliance located at the edge of the customer network may be more desirable.

Example 2 *Consider the case when the tag vector of packet p_f^k becomes re-usable at the source, there are n unacknowledged packets, $p_f^{k+1}, \ldots, p_f^{k+n}$, with* **larger** *tag vectors in the network. Let the tag vector of p_f^k be re-assigned to packet p_f^{k+n+1}. Now let the tag vectors of the first $(n-1)$ unacknowledged packets $p_f^{k+1}, \ldots, p_f^{k+n-1}$ also become available for re-use; let these tag vectors be assigned to subsequent $(n-1)$ packets, namely, $p_f^{k+n+2}, \ldots, p_f^{k+n+n}$. Consider the case where packet p_f^{k+n} departs the egress node at its deadline, $F_H(p_f^{k+n})$. Since packets $p_f^{k+n+1}, \ldots, p^{k+n+n}$ have smaller deadlines, they are guaranteed to depart the egress router earlier than p_f^{k+n}. Therefore, these n packets need to be buffered, simultaneously for some time, at the sequencer till packet p_f^{k+n} arrives. Larger the value of n, the larger the buffer space requirement at the sequencer.*

In practice, a sequencer would have a fixed amount of buffer space. In order to avoid packet loss due to overflow of the sequencer buffers, therefore, the aggressiveness of sources using a CSGT network may need to be controlled. We do this by employing a *flow control algorithm* that limits the maximum number of deadlines that are simultaneously in use for packets of a flow. Specifically, the flow control algorithm ensures that at any point in time t, no packet is assigned a deadline larger than $t + W\frac{l_f}{r_f}$, where W is a configuration parameter. When a packet arrives at time t, if no deadline smaller than $t + W\frac{l_f}{r_f}$ is available for assigning to it, the packet is held till one is available.

Observe that a large value of W increases the buffer space requirement at the sequencer (Example 2). A small value of W, on the other hand, limits the extent to which the source can utilize idle bandwidth in the network. In fact, if $W = 1$, the first server does not transmit a packet before its expected arrival time; in this case the server reduces to the non-work-conserving Jitter Virtual Clock server. In practice, the largest value of W—such that buffer overflow at the sequencer can be avoided—should be selected. If B denotes the available sequencer buffer space, in units of the packet size l_f, then Lemma 1 provides a condition that when satisfied by W, avoids packet loss due to buffer overflow.

Lemma 1 *Packets of flow f will not be dropped at the sequencer due to unavailability of re-ordering buffers if W satisfies:*

$$B \geq \begin{cases} (N+1)(W-1) - (N)(N+1)\frac{k^{min}}{2}, & \text{if } T^{min} \geq \frac{l_f}{r_f} \\ \frac{W(W-1)}{2k^{min}}, & \text{if } T^{min} < \frac{l_f}{r_f} \end{cases} \quad (7)$$

where $N = \lfloor \frac{W-2}{k^{min}} \rfloor$, $k^{min} = \frac{T^{min}}{l_f/r_f}$, and T^{min} is a lower bound on the round-trip time[5].

[5]T^{min} is a lower bound on the time difference between the transmission

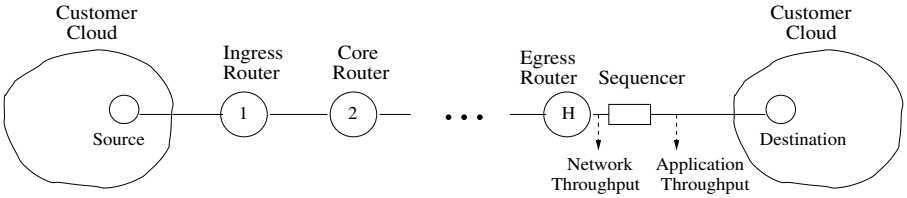


Figure 1: The CSGT Network Architecture

Given the largest value of W that satisfies (7), there is a bound on the maximum amount of available bandwidth that a flow f can utilize. Conversely, one can provision buffer space at the sequencer that allows a flow to utilize up to a maximum bandwidth (say r'). In Appendix C, we show that to allow a source to utilize bandwidth r', the chosen value of W should satisfy the following condition[6]:

$$W \geq \frac{r'}{l_f}\left(\sum_{j=1}^{H-1}\pi_j + \sum_{j=1}^{H}\beta_{f,j} + D^{max}\right) + H + 1 \quad (8)$$

Such a value of W can then be used to provision the sequencers with the appropriate amount of buffers. In particular, given r', the maximum bandwidth that a flow should be allowed to utilize, one can derive a bound on W using (8); this value of W when substituted in (7) determines the minimum buffer requirement at the sequencer.

4.3 Properties of CSGT Networks

Delay Guarantee The following lemma proves that deadline guarantees of CSVC are preserved in a CSGT network.

Lemma 2 *A packet p_f^k is guaranteed to depart server j in a CSGT network by $(F_j(p_f^k) + \beta_{f,j})$.*

It is important to observe that the deadlines assigned to a packet in a CSGT network are never larger than the dealines assigned to the same packet in a corresponding CSGR network. From Lemma 2, therefore, it follows that a CSGT network is guaranteed to deliver a packet no later than a CSGR network. This is true despite the additional delay introduced by the sequencer—the sequencer is guaranteed to deliver a packet p_f^k by its CSGR deadline. This is because all packets with smaller sequence numbers are guaranteed to arrive at the sequencer before their CSGR deadlines, which are smaller than the CSGR deadline of packet p_f^k.

of a packet at the first server and the arrival of its acknowledgment at the first server. For instance, the sum of propagation and minimum transmission latencies on all the links on both the forward and reverse path qualifies as a lower bound.

[6]The condition in (8) can be intuitively seen to be a form of the commonly used *delay-bandwidth product* rule-of-thumb.

Throughput Guarantee To quantify the effect of packet re-ordering on the throughput received by the applications, we define two different throughput measures. We define *network throughput* as the number of bits that depart the egress router during a given time interval, and *application throughput* as the number of bits that depart the sequencer (after re-ordering) during the interval. Note that the application throughput in any given interval may be different from the network throughput. Theorem 2 provides lower bounds on the network and application throughput in 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 sent by the egress node reaches the ingress node, then the network guarantees a minimum throughput in any time interval $[t_1, t_2]$, $W_{f,H}(t_1, t_2)$, given by:*

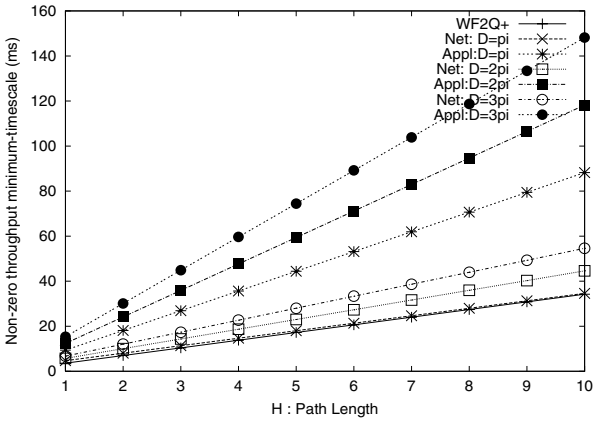
$$W_{f,H}(t_1,t_2) > r_f(t_2-t_1) - r_f * D^{max} - r_f\left((H+2)\frac{l_f}{r_f} + \sum_{j=1}^{H-1}\pi_j + \sum_{j=1}^{H}\beta_{f,j}\right)$$

Further, the sequencer guarantees a minimum throughput, $W_f^{app}(t_1, t_2)$, given by:

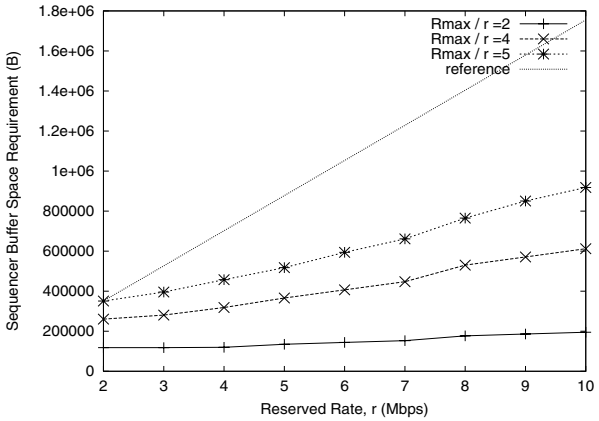
$$W_f^{app}(t_1,t_2) > r_f(t_2-t_1) - r_f * D^{max} - W * l_f - r_f\left((H+1)\frac{l_f}{r_f} + \sum_{j=1}^{H-1}\pi_j + \sum_{j=1}^{H}\beta_{f,j}\right)$$

The bound on the network throughput derived in Theorem 2 for a CSGT network differs from that provided by a core-stateful network of fair servers (Section 3), by a constant term $E_1 = r_f * [D^{max} - \sum_{j=1}^{H}(\gamma_{f,j} - \beta_{f,j})] + l_f$. The bound on application throughput differs by the additional term $E_2 = (W-1) * l_f$.

Observe that for a CSGT network derived from CSVC, $\beta_{f,j} = l^{max}/C_j$. Further, for most fair schedulers, $\gamma_{f,j} \geq l^{max}/C_j$. Therefore, $E_1 \leq r_f * D^{max} + l_f$, which is primarily governed by D^{max}, the maximum latency on the reverse path.



(a) Non-zero throughput time-scale



(b) Sequencer buffer space requirement

Figure 2: Evaluation of a CSGT Network

5 Evaluation of CSGT Networks

5.1 The Throughput Guarantee

Theorem 2 states that when measured over any time interval larger than $\gamma_{f,H}^{net}$, the network guarantees a non-zero throughput at a rate equal to the reserved rate. As discussed in Section 3, the smaller the value of $\gamma_{f,H}^{net}$, the better the network can support applications with stringent timeliness requirements. To evaluate the throughput guarantee of a CSGT network numerically, we compute $\gamma_{f,H}^{net}$ for example networks, where link capacities are $100Mbps$ and link propagation latencies are $1ms$. In figure 2(a), we plot $\gamma_{f,H}^{net}$ against the number of hops on the end-to-end path of a sample flow with a reserved bit-rate of $10Mbps$. D^{max} is varied from a multiple of 1 to 3 of the end-to-end propagation latency on the reverse path. For comparison, we also plot $\gamma_{f,H}^{net}$ for a representative core-stateful network of fair WF^2Q+ [3] servers. We observe the following:

1. When D^{max} is equal to the end-to-end link propagation latency, the throughput guarantee of the CSGT networks is similar to that of the core-stateful WF^2Q+ networks.
2. $\gamma_{f,H}^{net}$ increases with D^{max}. Therefore, the throughput guarantee of a CSGT network improves by provisioning low-delay feedback channels. However, even when D^{max} is three times the end-to-end propagation latency, the application is guaranteed a non-zero throughput over any time interval larger than $150ms$.

These observations imply that by provisioning low-delay feedback channels, a CSGT network can provide non-zero throughput guarantees at very short time-scales, and similar to those in core-stateful networks.

5.2 Sequencer Buffer Space vs. Maximum Throughput

Recall that there is a tradeoff between the amount of buffer space required at the sequencer and the maximum rate that the source is allowed to achieve. We numerically characterize this tradeoff for the same example network as above (link capacity = $100Mbps$, link propagation latency = $1ms$). In figure 2(b), we plot the minimum sequencer buffer space required to allow sources to achieve a given maximum bit-rate, R^{max} (varied from 2 to 5 times the reserved rate). We observe that:

1. To enable a flow, with a reserved bit-rate of up to $10Mbps$, to achieve an end-to-end bit-rate of up to 5 times that ($50Mbps$), less than $1MB$ of sequencer buffer space is sufficient.
2. The buffer space requirement grows slower with increase in the bit-rate of the sample flow (for reference, we plot a line where the buffer requirement grows at the same rate as the reserved bit-rate).

These observations imply that a small amount of sequencer buffer space is sufficient to allow flows to utilize bandwidth up to multiple times their reserved rate. Further, a CSGT network can reduce the total buffer requirement at the sequencer by aggregating into a single large flow, all micro-flows that traverse the same path between a pair of edge routers.

5.3 Deployment Considerations

A CSGT network introduces two main overheads—namely, additional state in every packet, and additional traffic due to feedback messages—in order to provide throughput guarantees in a core-stateless architecture. To address the first overhead, several techniques for encoding state in packets efficiently have been discussed in [15]. We expect the overhead

due to feedback traffic to also be small because of two reasons. First, the size of each feedback packet is small; additionally, the packetization overhead can further be reduced by acknowledging the receipt of multiple sequence numbers in the same feedback message. In applications that involve a bidirectional transfer of data, in fact, the feedback can be *piggy-backed* onto data packets transmitted on the reverse path. Second, feedback messages are transmitted only when data packets depart the network *much earlier* than their deadlines. This happens only when sufficient spare bandwidth is available on the forward path. While it would be useful to quantify precisely the expected amount of feedback traffic generated in practice, the lack of real testbeds and traffic characteristics prevents us from doing so.

6 Summary

End-to-end throughput guarantees is an important network service semantics to offer. Existing network architectures either provide throughput guarantees at the cost of introducing the complexity of per-flow state maintenance in all routers, or do not provide throughput guarantees at finite time-scales. In this paper, we propose the Core-stateless Guaranteed Throughput (CSGT) network architecture—the *first* work-conserving network architecture that provides throughput guarantees to individual flows over finite time-scales, but without maintaining per-flow state in core routers. We develop the architecture in two steps. First, we show that for a network to provide end-to-end throughput guarantees, it must also provide end-to-end delay guarantees. Second, we demonstrate that two mechanisms —tag re-use and source rate control— when integrated with a work-conserving, core-stateless network that provides end-to-end delay guarantees, lead to the design of CSGT network that provides end-to-end throughput bounds within an *additive constant* of what is attained by a core-stateful network of fair rate servers.

References

[1] 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. to appear in IEEE/ACM Transactions on Networking.

[2] 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.

[3] J.C.R. Bennett and H. Zhang. Hierarchical Packet Fair Queueing Algorithms. In *IEEE/ACM Transactions on Networking*, volume 5, pages 675–689, Oct 1997.

[4] Z. Cao, Z. Wang, and E. Zegura. Rainbow Fair Queueing: Fair Bandwidth Sharing Without Per-Flow State. In *Proceedings of IEEE INFOCOM*, March 2000.

[5] A. Clerget and W. Dabbous. TUF: Tag-based Unified Fairness. In *Proceedings of IEEE INFOCOM*, April 2001.

[6] 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.

[7] S.J. Golestani. A Self-Clocked Fair Queueing Scheme for High Speed Applications. In *Proceedings of INFOCOM'94*, 1994.

[8] P. Goyal and H.M. Vin. Generalized Guaranteed Rate Scheduling Algorithms: A Framework. Technical Report TR-95-30, Department of Computer Sciences, The University of Texas at Austin, 1995. Available via URL http://www.cs.utexas.edu/users/dmcl.

[9] P. Goyal, H.M. 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.

[10] J. Kaur and H.M. Vin. Core-stateless Guaranteed Rate Scheduling Algorithms. In *Proceedings of IEEE INFOCOM*, volume 3, pages 1484–1492, April 2001.

[11] J. Kaur and H.M. Vin. Core-stateless Guaranteed Throughput Networks. *Technical Report TR-01-47, Department of Computer Sciences, University of Texas at Austin*, November 2001.

[12] S. Keshav. On Efficient Implementation of Fair Queuing. *Journal of Internetworking Research*, 2:157–173, September 1995.

[13] 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.

[14] S. Shenker, L. Zhang, and D. Clark. A Scheduling Service Model and a Scheduling Architecture for an Integrated Services Packet Networks. Available via anonymous ftp from ftp://ftp.parc.xerox.com/pub/archfin.ps, 1995.

[15] I. Stoica. Stateless Core: A Scalable Approach for Quality of Service in the Internet. *PhD thesis, Carnegie Mellon University, Pittsburgh, PA*, December 2000.

[16] 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'98*, Sept 1998.

[17] I. Stoica and H. Zhang. Providing Guaranteed Services Without Per Flow Management. In *Proceedings of ACM SIGCOMM'99*, Sept 1999.

[18] H. Zhang. Service Disciplines For Guaranteed Performance Service in Packet-Switching Networks. *Proceedings of the IEEE*, 83(10), October 1995.

[19] H. Zhang and S. Keshav. Comparison of Rate-Based Service Disciplines. In *Proceedings of ACM SIGCOMM*, pages 113–121, August 1991.

[20] L. Zhang. VirtualClock: A New Traffic Control Algorithm for Packet Switching Networks. In *Proceedings of ACM SIGCOMM'90*, pages 19–29, August 1990.

[21] 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*, Dec 2000.

A Proof of Theorem 1

Consider $t_1 = a^1_{f,1}$ and $t_2 = d^k_{f,H}$. Then $W_{f,H}(a^1_{f,1}, d^k_{f,H}) = \sum_{i=1}^{k} l^i_f$. Substituting into the throughput bound, we get: $d^k_{f,H} \leq a^1_{f,1} + \frac{\Phi}{r_f} + \sum_{i=1}^{k} \frac{l^i_f}{r_f}$. Since the source transmits at least at its reserved rate, $EAT_1(p^k_f) = a^1_{f,1} + \sum_{i=1}^{k-1} \frac{l^i_f}{r_f}$. Therefore, $d^k_{f,H} - EAT_1(p^k_f) \leq \frac{\Phi + l^k_f}{r_f} \leq \frac{\Phi + l^{max}_f}{r_f}$, for all k. Therefore, the network provides a delay guarantee to flow f.

B Proof of Lemma 1

We refer to packets that are assigned re-used tags as *future* packet. We assume that a future packet is removed from the sequencer re-ordering buffers as soon as all packets with smaller sequence numbers arrive.

Let $B(t)$ denote the occupancy of the sequencer re-ordering buffers of a flow f at time t. Let t_i denote the time instant at which the i^{th} future packet is removed from these buffers. For ease of analysis, we assume that even if all packets with smaller sequence numbers reach the sequencer before it, a future packet is still buffered and removed (in such as scenario, at time instances t_i^- and t_i respectively). Let t_0 denote the initial time at which the source starts transmitting packets, and $B(t_0) = 0$. It follows that, $B(t_i) \leq B(t_i^-) - 1$ (more than one future packets may be removed from the buffers at the same instant). The buffer occupancy in any time interval $[t_i, t_{i+1})$ is non-decreasing with time (since no packets are removed in this interval). Therefore, the maximum buffer occupancy in this interval is given by $B(t_{i+1}^-)$.

Consider the first server (ingress node) at a time t_i. Consider all future packets transmitted by this server, that have not departed the sequencer (by time t_i). Of these, let b_1 denote the future packet with the smallest sequence number. Consider the set of all packets with smaller sequence numbers than packet b_1, that have not yet (at t_i) reached the sequencer. Among these, let p' have the largest tag-vector—then all of these packets would reach the sequencer at most by time $(F_H(p') + \beta_{f,H})$, and the packet b_1 would not need to be buffered after that.

Let t' be the time at which packet p' arrives at the first server, and is assigned a tag-vector. Since all future packets that have not departed the sequencer by time t_i, have a larger sequence number than p', they are transmitted from the first server after t'. Let B" be the total number of future packets that get transmitted from the first server in the interval $(t', F_1(p')]$, with *smaller* tag-vectors than p'. Since p' has not reached the sequencer by t_i, any future packets transmitted after t' with *larger* tag-vectors than p' have also not reached the sequencer. Hence, $B(t_i) \leq B$", and B" is the maximum number of future packets that would need to be buffered at the sequencer before p' gets delivered, that is, $B(t_{i+1}^-) \leq B$".

Due to source flow control, observe that: $t' \geq F_1(p') - W * l_f/r_f$. The number of distinct tag-vectors that lie in the interval $(t', F_1(p')]$ is given by: $W' = \left\lfloor \frac{(F_1(p'))^- - t'}{l_f/r_f} \right\rfloor \leq W - 1$. Let T^{min} denote a lower bound on the round-trip time—the time difference between the transmission of a packet at the first server and the arrival of its acknowledgment at the first server. The maximum number of tags that can become re-usable from the interval $(t', F_1(p')]$ after time t', which is an upper bound on $B(t_{i+1}^-)$, is given by:

$$\leq \begin{cases} W' + (W' - \lceil k^{min} \rceil) + ... + (W' - \lceil N' k^{min} \rceil), \\ \qquad\qquad \text{if } T^{min} \geq \frac{l_f}{r_f} \\ \left\lfloor \frac{1}{k^{min}} \right\rfloor + \left\lfloor \frac{2}{k^{min}} \right\rfloor + \ldots + \left\lfloor \frac{W'}{k^{min}} \right\rfloor, \text{ if } T^{min} < \frac{l_f}{r_f} \end{cases}$$

where $k^{min} = \frac{T^{min}}{l_f/r_f}$, and $N' = \left\lfloor \frac{W'-1}{k^{min}} \right\rfloor$. The right-hand-side of the above is an increasing function of W'. Since $W' \leq W - 1$,

$$B(t_{i+1}^-) \leq \begin{cases} (N+1)(W-1) - (N)(N+1)\frac{k^{min}}{2}, \\ \qquad\qquad \text{if } T^{min} \geq \frac{l_f}{r_f} \\ \frac{W(W-1)}{2k^{min}}, \qquad \text{if } T^{min} < \frac{l_f}{r_f} \end{cases}$$

where $N = \left\lfloor \frac{W-2}{k^{min}} \right\rfloor$. Since this upper bound on $B(t_{i+1}^-)$ is independent of i, it is an upper bound on the maximum buffer occupancy in all time intervals $[t_i, t_{i+1}), i \geq 0$. Therefore, if the provisioned buffer space, B, is at least as large as given by this upper bound, no packets are lost at the sequencer re-ordering buffers.

C Condition on W to Allow Large Throughput to be Achieved

Suppose $r' > r_f$ is the bottleneck bandwidth available to the flow f at time t_0 (and thereafter). Without loss of generality, assume that the first server is the bottleneck server[7]. We derive a condition on W such that if there is only a single packet at the bottlenecked first server (and in the network) at a time t_0, then there continues to be at least one packet at the bottlenecked server at all future times (given that the source has packets to transmit). If this is the case, then the bottleneck bandwidth available to flow f is not wasted.

Suppose packet p_f^1 arrives at the server queue at time t_0. Then it can be shown that packet p_f^k, (where packets $p_f^1, \ldots, p_f^k$ are transmitted back-to-back) would incur a maximum delay of $(\sum_{j=1}^{H-1} \pi_j + \sum_{j=1}^{H} \beta_{f,j} + (H+k-1) * l_f/r')$ before it departs from the network. The acknowledgment for packet p_f^k would reach the ingress node after an additional delay of at most D^{max}. The corresponding S_1 of this packet at the ingress node would be $S_1(p_f^k) = t_0 + (k-1) l_f/r_f$. Therefore, the tag-vector of packet p_f^k would definitely be available for re-use at the first server, if k satisfies: $\sum_{j=1}^{H-1} \pi_j + \sum_{j=1}^{H} \beta_{f,j} + (H+k-1) * l_f/r' + D^{max} \leq (k-1) * l_f/r_f$. This implies:

$$k \geq \frac{\sum_{j=1}^{H-1} \pi_j + \sum_{j=1}^{H} \beta_{f,j} + D^{max} + H * \frac{l_f}{r'}}{\frac{l_f}{r_f} - \frac{l_f}{r'}} + 1 \qquad (9)$$

[7] Suppose this is not the case. Then even if the first server transmits packets once only every l_f/r' time units, the bottleneck server will remain backlogged.

Now observe that, if the bottlenecked first server remains backlogged till the time the acknowledgment for p_f^k arrives, then the subsequent acknowledgments (spaced l_f/r' apart) clock the transmission of new packets, and the bottleneck server, that transmits a packet every l_f/r' units, would remain backlogged subsequently.

Further, due to source flow control, the tags for packet p_f^k would become available for re-use at most by the time $W + m_k$ packets arrive for transmission at the first server after t_0, where m_k is given by the term:

$$\left\lfloor \frac{\sum_{j=1}^{H-1}\pi_j + \sum_{j=1}^{H}\beta_{f,j} + (H+k-1)\frac{l_f}{r'} + D^{max}}{l_f/r_f} \right\rfloor \quad (10)$$

Therefore, if the time it takes for the first server to transmit $W + m_k$ packets at the rate r', is at least as large as the maximum time it takes for the acknowledgment of packet p_f^k to arrive after t_0, the first server always remains backlogged with packets to transmit. That is, the following condition would ensure that the server remains continuously backlogged: $(W+m_k) * \frac{l_f}{r'} \geq \sum_{j=1}^{H-1}\pi_j + \sum_{j=1}^{H}\beta_{f,j} + D^{max} + (H+k-1) * \frac{l_f}{r'}$. From (9) and (10), this yields:

$$(W-1)l_f \geq r'(\sum_{j=1}^{H-1}\pi_j + \sum_{j=1}^{H}\beta_{f,j} + D^{max}) + H\, l_f \quad (11)$$

If, on the other hand, W does not satisfy condition (11) for the available bottleneck bandwidth r', then it can be seen, using a similar argument as above, that the throughput rate that the source can sustain, $R^{max}(W, r')$, is given by:

$$\simeq r_f + \frac{\left(1 - \frac{r_f}{r'}\right)(W-1) * l}{\sum_{j=1}^{H-1}\pi_j + \sum_{j=1}^{H}\beta_{f,j} + D^{max} + H * \frac{l_f}{r'}}$$

D Proof of Lemma 2

In the following, a *non-preemptive* scheduling algorithm is one that does not preempt the transmission of a lower priority packet even after a higher priority packet arrives. On the other hand, a *preemptive* scheduling algorithm always ensures that the packet in service is the packet with the highest priority by possibly preempting the transmission of a lower priority packet. A non-preemptive algorithm is considered *equivalent* to a preemptive algorithm if the priority assigned to all the packets is the same in both. The following lemma is stated and proved in [8].

Lemma 3 *If PS is a work conserving preemptive scheduling algorithm, NPS its equivalent non-preemptive scheduling algorithm and the priority assignment of a packet is not changed dynamically, then*

$$L_{NPS}(p^k) - L_{PS}(p^k) \leq \frac{l^{max}}{C}$$

where $L_{PS}(p^k)$ and $L_{NPS}(p^k)$ denote the time a packet leaves the server when PS and NPS scheduling algorithms are employed, respectively. Also, l^{max} is the maximum length of a packet and C is the capacity of the server.

Lemma 4 *If the j^{th} server's capacity is not exceeded, then the time at which packet p_f^k departs a Preemptive CSGT server, denoted by $L^j_{PCSGT}(p_f^k)$, is*

$$L^j_{PCSGT}(p_f^k) \leq F_j(p_f^k) \qquad j \geq 1 \quad (12)$$

Proof: Let $S_j(p_f^k) = F_j(p_f^k) - l_f^k/r_f$. At server j, define the quantity $R_{f,j}(t)$ for flow f as follows:

$$= \begin{cases} r_f \text{ if } \exists k \ni (a_{f,j}^k \leq t) \wedge (S_j(p_f^k) < t \leq F_j(p_f^k)) \\ 0 \quad \text{otherwise} \end{cases} \quad (13)$$

Let S be the set of flows served by server j. Then server j with capacity C_j is defined to have *exceeded its capacity* at time t if $\sum_{n\in S} R_{n,j}(t) > C_j$. Let $K_{f,j}(t_1, t_2)$ be the set of all flow f packets that arrive at server j in interval $[t_1, t_2]$ and have deadlines no greater than t_2. For packet p_f^k, let $T_{f,j}^k = F_j(p_f^k) - \max(a_{f,j}^k, S_j(p_f^k))$. The proof of Lemma 4 is by induction on j.

Base Case : j = 1. From (2) and (5), we have: $F_1(p_f^k) \geq a_{f,1}^k + l_f^k/r_f^k$. Then it can be observed from (4) and (13) that:

$$\int_{t_1}^{t_2} R_{f,1}(t)dt = \sum_{i\in K_f(t_1,t_2)} (r_{f,1}^i * T_{f,1}^i)$$
$$\geq \sum_{i\in K_f(t_1,t_2)} (r_{f,1}^i * \frac{l_f^i}{r_f^i}) \geq \sum_{i\in K_f(t_1,t_2)} l_f^i$$

Therefore, the cumulative length of all flow f packets that arrive in interval $[t_1, t_2]$ and have deadline value no greater than t_2, denoted by $AP_f(t_1, t_2)$, is given as $AP_f(t_1, t_2) \leq \int_{t_1}^{t_2} R_{f,1}(t)dt$.

We now prove the lemma by contradiction. Assume that for packet p_f^k, $L^1_{PCSGT}(p_f^k) > F_1(p_f^k)$. Also, let t_0 be the beginning of the busy period in which p_f^k is served and $t_2 = F_1(p_f^k)$. Let t_1 be the least time less than t_2 during the busy period such that no packet with deadline value greater than t_2 is served in the interval $[t_1, t_2]$ (it can be shown that such a t_1 exists). Clearly, all the packets served in the interval $[t_1, t_2]$ arrive in this interval (else they would have been served earlier than t_1) and have deadline value less than or equal to t_2. Since the server is busy in the interval $[t_1, t_2]$ and packet p_f^k is not serviced by t_2, we have: $\sum_{f\in S} AP_f(t_1, t_2) > C_1(t_2 - t_1)$. Since $AP_f(t_1, t_2) \leq \int_{t_1}^{t_2} R_{f,1}(t)dt$, we have:

$$\int_{t_1}^{t_2} \sum_{f\in S} R_{f,1}(t)dt > C_1(t_2 - t_1) \quad (14)$$

Since the server capacity is not exceeded, $\sum_{f \in S} R_{f,1}(t) \leq C_1$. Hence, $\int_{t_1}^{t_2} \sum_{f \in S} R_{f,1}(t)dt \leq C_1(t_2 - t_1)$. This contradicts (14) and hence the base case is proved.

Induction Hypothesis : Assume (12) holds for $1 \leq j \leq m$.

Induction Step : We will show that (12) holds for $1 \leq j \leq m+1$.

From (3) and the Induction Hypothesis, we get: $F_{m+1}(p_f^k) \geq a_{f,m+1}^k + l_f^k/r_f^k$. The induction step can now be proved in exactly the same manner as the base case. Therefore, from induction, the lemma follows. ■

Since Preemptive CSGT is work conserving and does not dynamically change the priority of a packet, Lemma 2 follows immediately from Lemma 4 and Lemma 3.

E Proof of Theorem 2

Since D^{max} is the maximum latency after which a packet from the egress node reaches the ingress node, the following condition is *sufficient* to ensure that the tags of packet p_f^k will be reused by the ingress node:

$$d_{f,H}^k + D^{max} \leq F_1(p_f^k) - \frac{l_f^k}{r_f} \tag{15}$$

Consider any tag value F at the *last* server H. Let k_F denote the sequence number of the last packet that is ever assigned the deadline tag F at server H. Then the departure time of packet $p_f^{k_F}$ from the last server, $d_H^{k_F}$, must not satisfy condition (15)—otherwise, tag F would be re-used for another packet, and $p_f^{k_F}$ does not qualify to be the last packet to be assigned the tag F. We therefore have:

$$F - d_H^{k_F} < D^{max} + \frac{l_f^{k_F}}{r_f^{k_F}} + \sum_{j=1}^{H-1}(\beta_{f,j} + \pi_j + \max_{1 \leq i \leq k_F} \frac{l_f^i}{r_f})$$

Let $\widehat{T}$ be the upper bound on the right hand side of the above for all F. That is, for a source with equal-sized packets, let $\widehat{T} = D^{max} + H\frac{l_f}{r_f} + \sum_{j=1}^{H-1} \pi_j + \sum_{j=1}^{H-1} \beta_{f,j}$.

Now consider any time interval (t_1, t_2). From the definition of $\widehat{T}$, we know that given any $F \geq (t_1 + \widehat{T})$ that has been assigned as a deadline tag at the last server H, at least one packet with this tag F will be delivered after time t_1 (this is true even for $t_1 = a_{f,1}^1$, since $F_1(p_f^1) \leq a_{f,1}^1 + \widehat{T}$). Further, from Lemma 2, we know that any packet with an assigned deadline tag F will be delivered no later than $F + \beta_{f,H}$. Therefore, for *every* $F \in [t_1 + \widehat{T}, t_2 - \beta_{f,H}]$ that has been assigned as a tag to any packet, *at least* one packet with that deadline tag will depart the last server in the time interval (t_1, t_2). Let $B(t_0, t)$ represent the total number of bits in packets that are *last* assigned deadlines that lie in any time interval $(t_0, t]$. Then, $W_{f,H}(t_1, t_2) \geq B(t_1 + \widehat{T}, t_2 - \beta_{f,H})$.

Let $t_3 = t_1 + \widehat{T}$ and $t_4 = t_2 - \beta_{f,H}$. Let $t_3' \geq t_3$ be the smallest time instant that coincides with a deadline—let the corresponding packet be $p_f^{k_3}$. Let $t_4' \leq t_4$ be the largest time instant that coincides with a deadline—let the corresponding packet be $p_f^{k_4}$. Then, since the source transmits at least at its reserved rate, the total number of bits in packets with deadline in the range (t_3', t_4') is given by: $B(t_3', t_4') = r_f(t_4' - t_3')$.

Now note that $t_3' < t_3 + l_f/r_f$. This is so because otherwise, $F(p_f^{k_3 - 1}) \in [t_3, t_3')$, which violates the definition of t_3'. Similarly, $t_4' > t_4 - l_f/r_f$. This is so because otherwise, $F(p_f^{k_4+1}) \in (t_4', t_4]$, which violates the definition of t_4'. Therefore, we have: $t_4' - t_3' > (t_4 - t_3) - 2l_f/r_f$. Therefore, we get: $W_{f,H}(t_1, t_2) \geq B(t_3, t_4) \geq B(t_3', t_4') = r_f(t_4' - t_3') > r_f(t_4 - t_3) - 2l_f$. This implies,

$$W_{f,H}(t_1, t_2) > r_f(t_2 - t_1) - r_f * D^{max} - r_f\left((H+2)\frac{l_f}{r_f} + \sum_{j=1}^{H-1}\pi_j + \sum_{j=1}^{H}\beta_{f,j}\right)$$

Next, consider a time instant $t_5 \in (t_3, t_4)$ that coincides with a deadline tag at the last server. Let $p_f^{k_5}$ be the last packet that is assigned the deadline t_5. Then $p_f^{k_5}$ is received at the sequencer in the interval (t_1, t_2). If $p_f^{k_5}$ is not a future packet, it departs the sequencer (for simplicity, we assume that packets depart the sequencer instantly). If not, it has to be buffered till all packets with smaller sequence numbers reach the sequencer as well. Among all these packets with smaller sequence numbers than $p_f^{k_5}$, let p' have the largest deadline (if there are more than one such packets, let p' be the last packet to be assigned that deadline).

Packet $p_f^{k_5}$ arrives for transmission at the first server latest by $F_1(p_f^{k_5}) - l_f^{k_5}/r_f$. Let t' be the time at which p' arrives for transmission at the first server. Since p' has a smaller sequence number than $p_f^{k_5}$, $t' < F_1(p_f^{k_5}) - l_f^{k_5}/r_f$. Further, due to source flow control, $F_1(p') \leq t' + W * l_f/r_f$. Therefore, $F_1(p') < F_1(p_f^{k_5}) + (W-1) * l_f/r_f \Rightarrow F_H(p') < t_5 + (W-1) * l_f/r_f$.

Since p' is guaranteed to be delivered at the sequencer by $F_H(p') + \beta_{f,H}$, $p_f^{k_5}$ will also be delivered by this time. This implies that, for any deadline in the interval $(t_1 + \widehat{T}, t_2 - \beta_{f,H} - (W-1) * l_f/r_f)$, the last packet to be assigned that deadline will depart *the sequencer* in the time interval (t_1, t_2). Therefore, the number of bits that depart the sequencer in the time interval (t_1, t_2), $W_f^{app}(t_1, t_2)$, is given by: $W_f^{app}(t_1, t_2) \geq B(t_3, t_4 - (W-1) * \frac{l_f}{r_f}) \geq B(t_3', t_4' - (W-1) * \frac{l_f}{r_f}) > r_f(t_4 - t_3) - 2r_f\frac{l_f}{r_f} - (W-1)l_f$. This implies,

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