

A Theory of Multi-Channel Schedulers for Quality of Service

Jorge A. Cobb Miaohua Lin

Department of Computer Science
The University of Texas at Dallas
Richardson, TX 75083-0688
{cobb,miaohua}@utdallas.edu

Abstract—A computer network consists of a set of computing nodes interconnected via communication channels. It is commonly assumed that, for each pair of network nodes u and v , there is at most one channel from u to v . However, it is often desirable to have multiple channels between nodes. That is, for every pair of network nodes u and v , there may be multiple channels from u to v . In this paper, we consider the problem of providing deterministic quality of service guarantees when there are multiple channels between nodes. We show that any packet scheduling protocol that operates over a single channel can be modified to operate over multiple channels. In addition, this transformation increases the packet delay through the node by only a small amount. However, having multiple channels between nodes may cause packet reorder. This reorder significantly increases the upper bound on end-to-end delay. We show how this increase in delay is avoided through the use of efficient sorting techniques.

I. INTRODUCTION

A computer network provides quality of service if there is an upper bound on the delay of packets through the network. This delay bound can be either probabilistic or deterministic. In this paper, we focus on deterministic delay bounds. Research on packet scheduling protocols that provide deterministic delay bounds flourished in the previous decade (for a survey, see [27]). Many of these protocols are based, one way or another, on earlier work on task scheduling. In particular, they are based on the techniques given in the landmark paper of Liu and Layland on periodic task scheduling [19].

In [19], all tasks share a single resource. However, in the last few years, there have been breakthroughs in the scheduling of tasks over *multiple* resources [1], [2], [20]. Even though task scheduling over multiple resources has been successful, there has been little work on packet scheduling over multiple channels between network nodes. This is due in part to the belief that multiple channels between nodes is either impractical or uncommon. However, there is significant evidence to the contrary.

In a recent paper [3], it was argued that packet reordering is not a “pathological” problem, but rather a normal occurrence. That is, packets are reordered not only due to route changes (which are rare), but also due to parallelism in the network. One reason for this parallelism is the aggressive deployment of parallel channels between nodes. As stated in [3], in a survey of 38 major service providers in 1997, only two had no parallel channels between their nodes. One reason for parallel channels is that they often reduce equipment and trunk costs. That is, it is often more cost effective to use two components in parallel

than to use one component with twice the capacity. In addition, parallel channels improve fault-tolerance.

Another technology that provides multiple channels between nodes is the establishment of light-paths in wave-division multiplexed (WDM) optical networks. Although the establishment of light-paths is usually semi-permanent, recent work allows the establishment of light-paths on-demand [14]. If there is a significant load between two nodes in the network, it is possible that a single light-path may not provide enough capacity between these nodes. In this case, additional light-paths may be established between them (for more examples of multi-channel systems, see [4].)

Based on the evidence presented above, it is likely that multiple channels between nodes will continue to exist. Therefore, if quality of service is deployed on a global scale, it will likely encounter network nodes with multiple channels between them. Hence, the problem of scheduling packets over multiple channels must be studied.

The first packet scheduling protocol that provides deterministic quality of service over multiple channels is presented in [4]. The scheduling protocol assigns timestamps to packets in the same manner as weighted-fair-queuing [18], [21], and packets are forwarded to channels in order of increasing timestamp. However, no other scheduling protocols were considered, and the end-to-end delay over a series of nodes was not determined.

In [10], we presented general techniques to develop scheduling protocols for nodes with multiple channels. However, the techniques were restricted to the case where all channels of a node have equal capacity. In this paper, we generalize the techniques to allow channels of different capacity. In particular, we present two techniques to transform a scheduling protocol that operates over a single channel into a scheduling protocol that operates over multiple channels. In addition, we consider the end-to-end delay of packets through a series of nodes with multiple channels. We observe that the packet reorder caused by multiple channels may significantly increase the end-to-end delay. We show how this increase in delay can be prevented through the use of efficient sorting techniques.

II. MULTI-CHANNEL SCHEDULERS

In this section, we define our network model and protocol notation. We begin with the usual network model where there is only a single channel between nodes. We then extend our model to multiple channels between nodes.

A *network* is a set of computers interconnected by point-to-point communication channels. The network may be viewed

as a graph, where each computer is a node in the graph and each channel is a directed edge in the graph. Thus, we use the terms computer and node interchangeably. A network is a *single-channel network* iff, for every pair of nodes u and v , there is at most one channel from u to v . Nodes u and v are *neighbors* iff there is a channel from u to v or there is a channel from v to u .

A *flow* is a sequence of packets that traverse the network, starting at the source node of the flow and ending at the destination node of the flow. A network may be traversed by multiple flows. The network path of each flow is fixed, and network resources are reserved for each flow. This reservation of resources ensures a bounded end-to-end packet delay.

In a single-channel network, every output channel in a node is equipped with a scheduler. From the input channels of the node, each scheduler receives packets from flows that traverse the output channel of the scheduler. The scheduler then chooses the transmission order and transmission time of these packets over its output channel.

As an example, consider Figure 1. Figure 1(a) shows a network with five nodes. There are four flows in the network, f , g , x , and y . Each channel is labelled with the flows that traverse it. Figure 1(b) shows the center node in detail, including the scheduler of each of its output channels, and the path through the node taken by each of the four flows.

We say a packet is *forwarded* to the output channel when its first bit is transmitted over the output channel. We say a packet *exits* a scheduler when the last bit of the packet is transmitted by the output channel, and hence, the output channel becomes idle at this moment. To simplify our discussion, we ignore channel propagation delays, since they simply add a constant delay to each packet. We say a scheduler is *work-conserving* if it does not allow its output channel to remain idle while its queue is non-empty.

We adopt the following notation for each flow f and each scheduler s along the path of f .

- R_f rate (bits/sec.) reserved for flow f .
- $p_{f,i}$ i^{th} packet of f , $i \geq 1$.
- $L_{f,i}$ length of packet $p_{f,i}$ (bits).
- $L_{f,i}^{\max}$ maximum of $L_{f,j}$, where $1 \leq j \leq i$.
- L_s^{\max} maximum packet length of all flows at s .
- $A_{s,f,i}$ arrival time of $p_{f,i}$ at scheduler s .
- $E_{s,f,i}$ exit time of $p_{f,i}$ from s .
- C_s output channel capacity (bits/sec.) of scheduler s .

We next enhance the network model to include multiple channels between nodes.

A *multi-channel network* is a network in which there exists a pair of nodes, u and v , such that the number of channels from u to v is greater than one. An example of a multi-channel network is shown in Figure 2(a). It is similar to the network in Figure 1(a), except that there are two channels between each pair of nodes. Figure 2(b) shows the center node in detail.

All output channels that lead to the same neighboring node are managed by a single scheduler. Therefore, all packets with a common next-hop node have a common scheduler. Because all output channels of a scheduler lead to the same node, the scheduler distributes the packets of each flow among all its

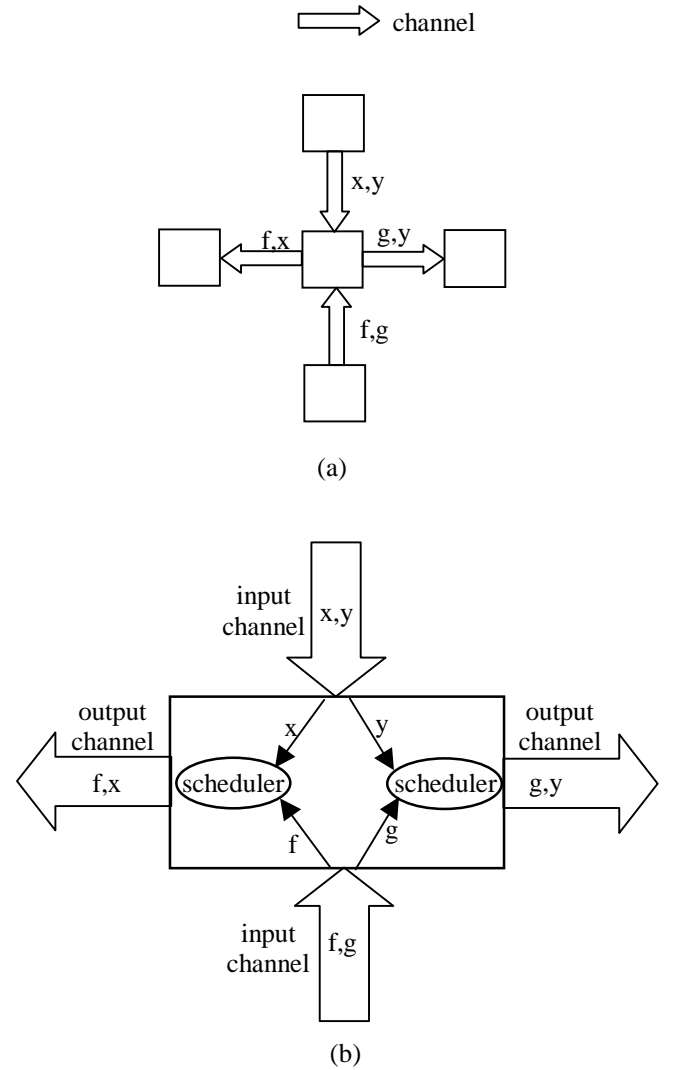


Fig. 1. Output channels and their schedulers.

output channels.

A scheduler is said to be a *multi-channel scheduler* if it manages multiple output channels, or a *single-channel scheduler* if it manages a single output channel.

For a multi-channel scheduler s , we adopt the following notation.

- N_s number of output channels managed by s .
- $c_{s,i}$ capacity of i^{th} output channel of s , $1 \leq i \leq N_s$.
- c_s^{\min} minimum channel capacity of s , i.e., $\min\{c_{s,i} \mid 1 \leq i \leq N_s\}$.
- c_s^{\max} maximum channel capacity of s , i.e., $\max\{c_{s,i} \mid 1 \leq i \leq N_s\}$.
- C_s total capacity of scheduler s , i.e., $\sum_{i=1}^{N_s} c_{s,i}$.

We assume that each scheduler forwards the packets of each flow in the order in which they are received. However, note that packets from a flow may be reordered along their path to the destination. To illustrate this, assume that multiple output channels of a scheduler are idle. In addition, assume that packets $p_{f,i}$ and $p_{f,(i+1)}$ are forwarded to a pair of idle

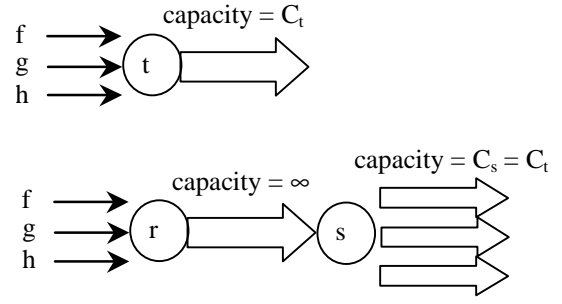
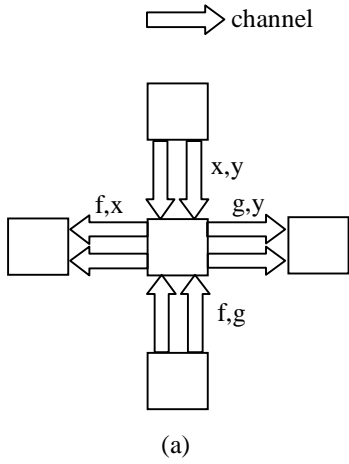


Fig. 3. Multi-channel emulation of a single-channel scheduler.

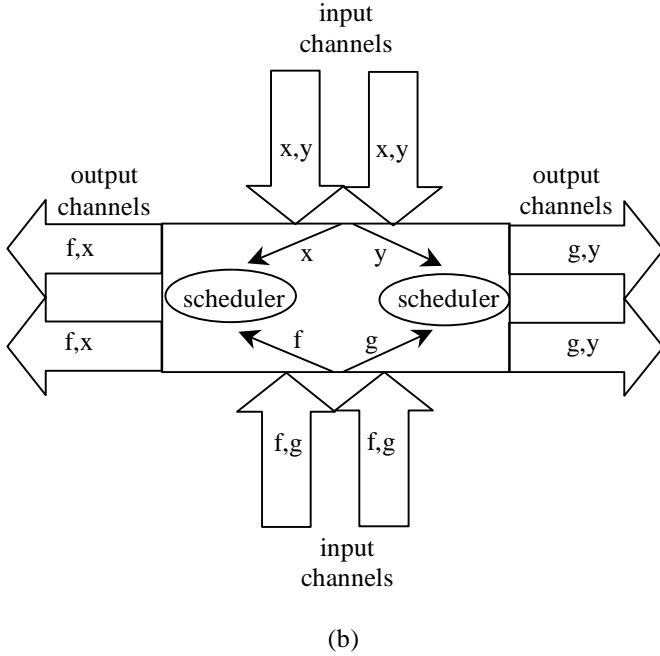


Fig. 2. Multiple output channels per node.

channels at the same time. If $L_{f,i} > L_{f,(i+1)}$, then $p_{f,(i+1)}$ will arrive to the next node earlier than $p_{f,i}$, and hence, these two packets become reordered.

Due to reorder, we say a flow g is derived from flow f if both flows have the same set of packets. That is, if g and f differ, then they differ only in the order of their packets.

When the packets of a flow are received at a node, the node has two options before transferring the packets to their scheduler. The node may simply transfer the packets in the order in which they are received, or the node may sort the packets back into their original order. We will consider both of these cases in later sections.

III. SINGLE-CHANNEL EMULATION

Single-channel schedulers have been studied extensively in the literature, and a large number of these schedulers

have been developed [27]. The obvious question to ask is if existing single-channel schedulers can be transformed into multi-channel schedulers. In this section, we present a simple technique by which *all* single-channel schedulers can be transformed into multi-channel schedulers.

Consider Figure 3. In this figure we have three schedulers. Scheduler t is a single-channel scheduler. Scheduler r is a single-channel scheduler whose output channel has infinite capacity and whose input flows are the same as those of t . Scheduler s is a multi-channel scheduler whose output capacity equals the output capacity of t .

The behavior of these schedulers is as follows. Scheduler r forwards each packet at exactly the same time at which the same packet is forwarded by t . Because the capacity of r is infinite, once r forwards a packet, the packet is received immediately by s . Notice that after r forwards a packet of size L to s , it may not forward the next packet until at least L/C_t seconds later. Therefore, r is not work-conserving. Finally, s is work-conserving. It simply queues the packets received from r , and it forwards these packets in FCFS order to its output channels. Scheduler s treats all its output channels alike, even though their capacity may not be equal. Thus, if a channel becomes idle, and the queue of s is not empty, then the next packet from the queue is forwarded over this channel. In a sense, schedulers r and s emulate the behavior of scheduler t , except that the packets are distributed over multiple channels.

We next compare the behavior of these two systems.

Theorem 1: Let t be a single-channel scheduler with capacity C_t . Assume a single-channel scheduler r has an output channel with infinite capacity, and r has the same input flows as t . Let r forward packets in the same order as t and at exactly the same time as t . Assume s is a work-conserving, FCFS, multi-channel scheduler such that $C_s = C_t$. Let the input of s be the output of r . Then, for all input flows f and for all i ,

$$E_{s,f,i} - E_{t,f,i} \leq \frac{(N_s - 1) \cdot L_s^{max} - L_{f,i}}{C_s} + \frac{L_{f,i}}{c_s^{min}}$$

Proof:

The *channel backlog* of a scheduler is the sum of the number of bits that remain to be transmitted from packets currently in a channel. The *total backlog* is the channel backlog plus the queue size.

If packet $p_{f,i}$ arrives at s and is sent immediately to a channel, (i.e., no queuing at s), then the extra delay suffered at s is at most

$$\frac{L_{f,i}}{c_s^{min}} - \frac{L_{f,i}}{C_t}$$

The desired result follows from $C_t = C_s$.

Assume now that when $p_{f,i}$ arrives at s , all channels are busy (i.e., $p_{f,i}$ is queued at s). Let $p_{g,j}$ be the latest packet before $p_{f,i}$ such that $p_{g,j}$ suffered no queuing at s (i.e., there was one empty channel at the time $p_{g,j}$ arrives at s). Thus, all channels are busy during the time interval $[A_{s,g,j}, A_{s,f,i}]$.

Let β be the channel backlog of s at time $A_{s,g,j}^-$. Note that there is one empty channel at time $A_{s,g,j}^-$. Thus, the queue of s is empty, and the total backlog is also equal to β . Note also that $\beta \leq (N_s - 1) \cdot L_s^{max}$, because there is one empty channel, and each channel can have at most L_s^{max} bits.

Let λ be the total number of bits in the sequence of packets arriving at s during the time interval $[A_{s,g,j}, A_{s,f,i}]$. That is, we start with $p_{g,j}$ and end with the packet previous to $p_{f,i}$.

Since packets arrive at s at a rate of at most C_t , which equals C_s , the length of the time interval $[A_{s,g,j}, A_{s,f,i}]$ is as follows.

$$(A_{s,g,j} - A_{s,f,i}) \geq \frac{\lambda}{C_s}$$

Also, during the time interval $[A_{s,g,j}, A_{s,f,i}]$, since all channels are busy, the number of bits forwarded is equal to

$$C_s \cdot (A_{s,f,i} - A_{s,g,j})$$

From the above two relations, the total number of bits forwarded during the interval $[A_{s,g,j}, A_{s,f,i}]$ is at least λ . That is, it is at least the total number of bits arriving during the interval. Therefore, the total backlog at time $A_{s,f,i}^-$ is at most the total backlog at time $A_{s,g,j}^-$, i.e., at most β .

The total backlog of β bits prevent $p_{f,i}$ from being forwarded into a channel. We next consider the length of time before $p_{f,i}$ is forwarded. Note that the worst delay for $p_{f,i}$ occurs when all channels finish clearing the backlog at the same time. This is because if a channel finishes earlier than others, then $p_{f,i}$ is forwarded via the idle channel. Because all channels are busy until $p_{f,i}$ is forwarded, and because the backlog is cleared by all channels at time same time, $p_{f,i}$ begins transmission no later than time

$$A_{s,f,i} + \frac{\beta}{C_s}$$

In the worst case, the slowest channel is assigned to $p_{f,i}$. Hence, $E_{s,f,i}$, is as follows.

$$E_{s,f,i} \leq A_{s,f,i} + \frac{\beta}{C_s} + \frac{L_{f,i}}{c_s^{min}}$$

The additional delay compared to a single channel scheduler is as follows.

$$E_{s,f,i} - E_{t,f,i} \leq A_{s,f,i} + \frac{\beta}{C_s} + \frac{L_{f,i}}{c_s^{min}} - (A_{s,f,i} + \frac{L_{f,i}}{C_s})$$

Thus,

$$E_{s,f,i} - E_{t,f,i} \leq \frac{\beta}{C_s} + \frac{L_{f,i}}{c_s^{min}} - \frac{L_{f,i}}{C_s}$$

We argued that $\beta \leq (N_s - 1) \cdot L_s^{max}$. Therefore,

$$E_{s,f,i} - E_{t,f,i} \leq \frac{(N_s - 1) \cdot L_s^{max}}{C_s} + \frac{L_{f,i}}{c_s^{min}} - \frac{L_{f,i}}{C_s}$$

Reducing the above,

$$E_{s,f,i} - E_{t,f,i} \leq \frac{(N_s - 1) \cdot L_s^{max} - L_{f,i}}{C_s} + \frac{L_{f,i}}{c_s^{min}}$$

■

Corollary 1: Under the same assumptions as those in Theorem 1, plus the additional restriction that all output channels of s have the same capacity, i.e., $c_s^{max} = c_s^{min} = \frac{C_s}{N_s}$, then, for all input flows f and for all i ,

$$E_{s,f,i} - E_{t,f,i} \leq \frac{(N_s - 1) \cdot L_s^{max}}{C_s} + \frac{(N_s - 1) \cdot L_{f,i}}{C_s}$$

■

Theorem 1 shows that we can implement a multi-channel scheduler based upon any single-channel scheduler. The penalty for doing so is an increase in the exit time of each packet. Notice, however, that the increase is quite small. From Corollary 1, if the output channels of s have similar capacities, then the difference in the exit times from s and t is less than the transmission time of two packets in a channel of s . This is because the transmission time of a packet is at most $\frac{N_s \cdot L_s^{max}}{C_s}$. This is not significant if the channel capacity is large.

The behavior of both r and s can be combined into a single scheduler that emulates t .

Definition 1: A multi-channel scheduler s emulates a single-channel scheduler t if and only if all of the following hold.

- $C_t = C_s$.
- Scheduler s forwards packets in the same order as scheduler t .
- A packet in s is forwarded only if it has already been forwarded by t .
- An output channel in s cannot remain idle while there are packets in s that have been forwarded by t .

■

We next consider the specific case where t is a PGPS scheduler. Below, $W_{t,f}(\tau)$ is the number of bits of flow f that scheduler t has forwarded during the interval $[0, \tau]$.

Corollary 2: Let G be a fluid GPS server, P be a PGPS scheduler that simulates G , and s be a multi-channel scheduler that emulates P . Let all channels of s have equal capacity. Then, for any time τ , for any input flow f , and for any i , $i \geq 1$,

$$E_{s,f,i} - E_{G,f,i} \leq \frac{N \cdot L_s^{max}}{C_s} + \frac{(N_s - 1) \cdot L_{f,i}}{C_s}$$

$$W_{G,f}(\tau) - W_{s,f}(\tau) \leq N_s \cdot L_s^{max} + (N_s - 1) \cdot L_{f,i}$$

Proof: Note that the maximum packet size and total capacity of P , G , and s are the same, so we drop the subscripts of these quantities. Consider first the exit time. The relationship between the exit times from P and G is as follows [18], [21].

$$E_{P,f,i} - E_{G,f,i} \leq \frac{L^{max}}{C}$$

From Corollary 1,

$$E_{s,f,i} - E_{P,f,i} \leq \frac{(N_s - 1) \cdot L^{max}}{C} + \frac{(N_s - 1) \cdot L_{f,i}}{C}$$

We combine the above two we obtain the desired result.

$$\begin{aligned} & E_{s,f,i} - E_{G,f,i} \\ & \leq \frac{(N_s - 1) \cdot L^{max}}{C} + \frac{(N_s - 1) \cdot L_{f,i}}{C} + \frac{L^{max}}{C} \\ & = \frac{N_s \cdot L^{max}}{C} + \frac{(N_s - 1) \cdot L_{f,i}}{C} \end{aligned}$$

We next consider W . For terseness, we define

$$\alpha = \frac{(N_s - 1) \cdot L^{max}}{C} + \frac{(N_s - 1) \cdot L_{f,i}}{C}$$

Also, for simplicity, we assume $W(\tau) = 0$ if $\tau < 0$.

Consider the difference between $W_{P,f}(\tau)$ and $W_{s,f}(\tau)$. From Corollary 1, every packet exits from s no later than the corresponding exit time from P plus α . Notice that the output channel of P has greater capacity than the single output channel over which the same packet is sent in s . Hence, any bit of a packet will exit from s no later than the time it would exit from P plus α . Hence,

$$W_{P,f}(\tau - \alpha) \leq W_{s,f}(\tau)$$

The following is a well known result of PGPS [18], [21].

$$W_{G,f}(\tau - \alpha) \leq W_{P,f}(\tau - \alpha) + L^{max}$$

Thus,

$$W_{G,f}(\tau - \alpha) - L^{max} \leq W_{s,f}(\tau) \quad (1)$$

Furthermore, in the worst case, G has only packets of f queued, and thus it can transmit $\alpha \cdot C$ bits from f in α seconds. Hence,

$$W_{G,f}(\tau) - W_{G,f}(\tau - \alpha) \leq \alpha \cdot C \quad (2)$$

Combining (1) with (2),

$$W_{G,f}(\tau) - W_{s,f}(\tau) \leq \alpha \cdot C + L^{max}$$

The right hand side is as follows after replacing α by its defined value.

$$\left(\frac{(N_s - 1) \cdot L^{max}}{C} + \frac{(N_s - 1) \cdot L_{f,i}}{C} \right) \cdot C + L^{max}$$

Simplifying we obtain the desired result.

$$N_s \cdot L^{max} + (N_s - 1) \cdot L_{f,i}$$

The multi-channel emulation of a PGPS server yields an algorithm that is somewhat different from the multi-channel PGPS algorithm presented in [4] (called MSFQ). Nonetheless, the same bound on the exit time of a packet given in Corollary 2 is precisely the same bound given for MSFQ in [4]. The bound on the number of bits served is a little higher, however. Nonetheless, as mentioned above, the result of Theorem 1 is not restricted to emulating PGPS servers. It can be applied to the emulation of *any* packet scheduler, regardless of type. For example, the theorem can be applied to the emulation of first-come-first-serve, weighted round-robin [22], rate-controlled static-priority [29], or the emulation of any scheduler in the family of rate-proportional servers [23].

IV. BOUNDED APPETITE SCHEDULERS

A disadvantage of the emulation of a single-channel scheduler by a multi-channel scheduler is that the latter is not work-conserving. For example, assume the channels of the multi-channel scheduler are idle, and two packets are received at the same time. After forwarding one of these packets, the next packet cannot be forwarded until L/C seconds later, where L is the size of the forwarded packet and C the total capacity of the scheduler. Another disadvantage is that the scheduler must have a timer that expires L/C seconds after transmitting a packet of size L . This may increase the implementation complexity of the scheduler.

In this section, we investigate the transformation of a work-conserving single-channel scheduler into a work-conserving multi-channel scheduler. We desire two properties of this transformation. First, it must encompass a large number of single-channel schedulers. Second, the resulting multi-channel scheduler should be as close as possible to the original single-channel scheduler.

We consider schedulers that assign deadlines to packets and forward packets in order of increasing deadline. The deadline of packet $p_{f,i}$ at scheduler s is denoted $D_{s,f,i}$. Assigning deadlines to packets can be done by assigning a timestamp to each packet, and then forwarding packets in order of increasing timestamp. However, the timestamp need not be exactly equal to the deadline. For example, in self-clocking fair queuing [16] and weighted fair queuing [18], [21], the timestamp assigned to each packet is not equal to its intended deadline. However, we require that if a packet $p_{f,i}$ has a timestamp greater than the timestamp of another packet $p_{g,j}$, then the deadline of $p_{f,i}$ is greater than the deadline of $p_{g,j}$.

We say that a scheduler is *deadline ordered* if it forwards packets in order of increasing deadline. Without loss of generality, we assume that $D_{s,f,i} > A_{s,f,i}$.

We next consider the exit time from a work-conserving, deadline ordered, multi-channel scheduler. To guarantee that packets exit by their deadlines, we must bound the occurrence of deadlines. We use the bound introduced in [12].

Let $Z_s(a, b)$ be the total number of bytes that arrive to s during interval $[a, b]$ and whose deadline is at most b . That is,

$$Z_s(a, b) = \left(\sum f, i : a \leq A_{s,f,i} \wedge D_{s,f,i} \leq b : L_{s,f,i} \right)$$

Definition 2: A scheduler s with capacity C_s satisfies the *bounded appetite property* for a constant θ if and only if, for all intervals $[a, b]$ in which $Z_s(a, b) > 0$,

$$Z_s(a, b) \leq ((b - a) \cdot C_s - \theta)$$

Theorem 2: Consider a multi-channel, work-conserving, deadline-ordered scheduler s . Let s have bounded appetite with constant θ . Then, for all input flows f and for all i , one of the following holds.

$$\begin{aligned} E_{s,f,i} & \leq D_{s,f,i} + \frac{N_s \cdot L_s^{max} - L_{s,f,i} - \theta}{C_s} + \frac{L_{s,f,i}}{C_s^{min}} \\ E_{s,f,i} & \leq A_{s,f,i} + \frac{L_{f,i}}{C_s^{min}} \end{aligned}$$

Proof: The proof is similar to that of Theorem 1. Assume that when packet $p_{f,i}$ arrives at s , there is an idle channel and $p_{f,i}$ is immediately forwarded. Then,

$$E_{s,f,i} \leq A_{s,f,i} + \frac{L_{f,i}}{c_s^{min}}$$

Consider now that when $p_{f,i}$ arrives at s , all channels are busy (i.e., $p_{f,i}$ is queued at s). Let τ be the latest time, but no later than $A_{s,f,i}$, such that one of the following holds:

- 1) A packet $p_{g,j}$ was forwarded directly to an output channel ($p_{g,j}$ has no queuing delay at s) and all packets forwarded in the interval $[\tau, A_{s,f,i}]$, including $p_{g,j}$, have deadlines at most $D_{s,f,i}$.
- 2) A packet $p_{g,j}$ was forwarded, where $D_{s,g,j} > D_{s,f,i}$, all packets forwarded in the interval $(\tau, A_{s,f,i}]$ have deadlines at most $D_{s,f,i}$, and all channels are busy during this interval.

Consider the first case. Any packet forwarded before $p_{f,i}$ must arrive after time τ and have a deadline of at most $D_{s,f,i}$. This is because $p_{g,j}$ has no queuing delay at time τ , and furthermore, only packets with deadlines at most $D_{s,f,i}$ are forwarded during the interval $[\tau, A_{s,f,i}]$.

Since the deadline of a packet is greater than its arrival time, only packets arriving in the interval $[\tau, D_{s,f,i}]$ can arrive after τ and have a deadline of at most $D_{s,f,i}$. From the bounded appetite property, the number of bytes in these packets add to at most

$$(D_{s,f,i} - \tau) \cdot C_s - \theta \quad (3)$$

Note that both packets $p_{g,j}$ and $p_{f,i}$ are counted in (3) above. Also, note that only the packets in (3) and the channel backlog at time τ^- prevent $p_{f,i}$ from being forwarded. The channel backlog at time τ^- can be at most $(N_s - 1) \cdot L_s^{max}$ bits, since there was an empty channel when $p_{g,j}$ arrived.

From the definition of $p_{g,j}$, all channels are busy until $p_{f,i}$ is forwarded. The case that delays $p_{f,i}$ the most is when all channels at the same time finish transmitting their packets. Hence, $p_{f,i}$ has an exit time as follows.

$$E_{s,f,i} \leq \tau + \frac{(D_{s,f,i} - \tau) \cdot C_s - \theta - L_{s,f,i}}{C_s} + \frac{(N_s - 1) \cdot L_s^{max}}{C_s} + \frac{L_{s,f,i}}{c_s^{min}}$$

Reducing we obtain,

$$E_{s,f,i} \leq D_{s,f,i} + \frac{(N_s - 1) \cdot L_s^{max} - \theta - L_{s,f,i}}{C_s} + \frac{L_{s,f,i}}{c_s^{min}}$$

A similar reasoning applies to the second case. The main difference is that in the first case, $D_{s,g,j} \leq D_{s,f,i}$, and hence, $p_{g,j}$ is counted in (3). This is no longer true in the second case. Hence, packets in the channel backlog at time τ (including $p_{g,j}$) plus those packets counted in (3) may exit s before $p_{f,i}$ is forwarded to a channel. The backlog at time τ is at most $N \cdot L_s^{max}$. Hence, the exit time is as follows.

$$E_{s,f,i} \leq D_{s,f,i} + \frac{N_s \cdot L_s^{max} - \theta - L_{s,f,i}}{C_s} + \frac{L_{s,f,i}}{c_s^{min}} \quad \blacksquare$$

Corollary 3: Under the same conditions as Theorem 2, with the additional restriction that all channels of s have equal capacity, $c_s^{min} = c_s^{max} = \frac{C_s}{N_s}$, for all input flows f and all i , one of the following holds.

$$E_{s,f,i} \leq D_{s,f,i} + \frac{N_s \cdot L_s^{max} - \theta}{C_s} + \frac{(N_s - 1) \cdot L_{s,f,i}}{C_s}$$

$$E_{s,f,i} \leq A_{s,f,i} + \frac{L_{s,f,i}}{c_s^{min}} \quad \blacksquare$$

Theorem 2 shows an upper bound on the exit time from a multi-channel, work-conserving, deadline-ordered scheduler with bounded appetite. Notice that it is possible that the multi-channel scheduler simply assigns to each packet the same deadline that a bounded-appetite single-channel scheduler. Thus, the multi-channel scheduler may assign deadlines in the same way as any of the following protocols, all of which have been shown to have bounded appetite [12]: virtual clock [25], time-shift scheduling [8]¹, weighted fair queuing [18], [21], stop-and-go [15]², and the whole family of rate-proportional servers [23]³.

We next consider the specific case where the packet timestamp of a multi-channel scheduler s is the same as the packet timestamp of a PGPS scheduler.

Corollary 4: Let G be a GPS server and P be a PGPS scheduler that simulates G . Let s be a multi-channel, work-conserving, deadline-ordered scheduler that assigns the same timestamp to each packet as P does. Let $C_s = C_p$, and all channels of s have the same capacity. Then, for any time τ , for any input flow f , and for any i , $i \geq 1$,

$$E_{s,f,i} - E_{G,f,i} \leq \frac{N_s \cdot L_s^{max}}{C_s} + \frac{(N_s - 1) \cdot L_{f,i}}{C_s}$$

$$W_{G,f}(\tau) - W_{s,f}(\tau) \leq N_s \cdot L_s^{max} + (N_s - 1) \cdot L_{f,i}$$

Proof: Consider first the exit time. In [12], it was shown that PGPS is a bounded-appetite scheduler with $\theta = 0$ and $D_{P,f,i} = E_{G,f,i}$ ⁴. Thus, since $D_{s,f,i} = D_{P,f,i} = E_{G,f,i}$, and since $C_s = C_p$, s is also a bounded appetite scheduler with $\theta = 0$. Thus, from Corollary 3,

$$E_{s,f,i} \leq E_{G,f,i} + \frac{N_s \cdot L_s^{max}}{C_s} + \frac{(N_s - 1) \cdot L_{s,f,i}}{C_s}$$

which is the desired result.

The proof of the bound on W is the same as that in Corollary 2. \blacksquare

The assignment of deadlines in a multi-channel scheduler s as given in Corollary 4 yields the same algorithm as MSFQ

¹Time-shift scheduling was shown to be a member of the rate-proportional family of schedulers [5].

²Although stop-and-go is not work-conserving, it defines an eligibility time for each packet. After the eligibility time of the packet has elapsed, the packet is scheduled according to its deadline. Thus, if we consider each packet as arriving into the multi-channel scheduler at its eligibility time, then Theorem 2 also applies to stop-and-go deadlines.

³Although not shown in [12], it is easy to show that all rate-proportional schedulers have bounded appetite. The deadline is the time at which the system potential reaches the potential of the packet.

⁴Although P does not compute $E_{G,f,i}$, the timestamps P assigns to packets ensure that packets are forwarded in order of their exit time from G [12].

[4]. The bound on the exit time of a packet given in Corollary 4 is precisely the same bound given for MSFQ in [4]. The bound on the number of bits served is a little higher, however. Nonetheless, as mentioned above, Theorem 2 is not restricted to GPS deadlines. It can be applied to any assignment of deadlines that satisfy the bounded appetite property.

V. SINGLE-CHANNEL END-TO-END DELAY

In this section, we review the end-to-end quality of service model of a single-channel network. We base our model on the models of [6], [13]. We present the end-to-end quality of service model of a multi-channel network in Section VI.

Consider scheduler s and an input flow f of s . We define the *start-time* $S_{s,f,i}$ of packet $p_{f,i}$ at scheduler s as follows [6], [13]. Assume s were to forward the packets of f at exactly R_f bits/sec.. Then, $S_{s,f,i}$ is the time at which the first bit of $p_{f,i}$ is forwarded by s . More formally,

$$\begin{aligned} S_{s,f,1} &= A_{s,f,1} \\ S_{s,f,i} &= \max \left(A_{s,f,i}, S_{s,f,(i-1)} + \frac{L_{f,(i-1)}}{R_f} \right), \\ &\text{for every } i, i > 1 \end{aligned}$$

The first equation above is obvious. For the second equation, the first bit of packet $p_{f,i}$ begins to be transmitted after packet $p_{f,(i-1)}$ exits s , that is, after time $S_{s,f,(i-1)} + L_{f,(i-1)}/R_f$.

The start-time is measured assuming s forwards the bits of f at exactly the rate R_f . However, the true forwarding rate may be different. Nonetheless, on average, s forwards the packets of f at a rate of at least R_f . Therefore, the exit time of packet $p_{f,i}$ from s is close to its start time, $S_{s,f,i}$. If s has this property, we say s is a *start-time scheduler* [6], [7], [13], [17].

Definition 3: A scheduler s is a *start-time scheduler* if and only if, for every input flow f of s , and every $i, i \geq 1$,

$$E_{s,f,i} \leq S_{s,f,i} + \delta_{s,f,i}$$

for some constant $\delta_{s,f,i}$. ■

We refer to $\delta_{s,f,i}$ as the *start-time delay* of packet $p_{f,i}$ at scheduler s , and we refer to $S_{s,f,i} + \delta_{s,f,i}$ as the *start-time exit bound* of $p_{f,i}$ at s . We assume the values of δ are such that the exit bounds are increasing. That is, for all i , $S_{s,f,i} + \delta_{s,f,i} < S_{s,f,(i+1)} + \delta_{s,f,(i+1)}$.

The start-time delay is broad enough to represent the delay provided by many scheduling protocols. For example, by choosing $\delta_{s,f,i} = L_{f,i}/R_f + L_s^{max}/C_s$, $\delta_{s,f,i}$ becomes the delay of virtual-clock and weighted-fair-queueing protocols [21], [25]. Another example is the real-time channel model, [11], [28] where each flow has constant packet size and constant packet delay. This is represented above by having $\delta_{s,f,i}$ and $L_{f,i}$ be constant for all i .

We next consider the delay of a packet across a sequence of schedulers. Because the start-time of a packet determines its exit time from a scheduler, a bounded end-to-end delay requires a bounded per-hop increase in the start-time of the packet. This bound is as follows.

Theorem 3: Let s be a start-time scheduler, f be an input flow of s , and t be the next scheduler after s . Then, for all i ,

$$S_{t,f,i} \leq S_{s,f,i} + \Delta_{s,f,i}$$

where $\Delta_{s,f,i} = \max_{1 \leq x \leq i} \{\delta_{s,f,x}\}$. ■

This bound was shown in [7], [13], and it also follows from the results in [28]. From induction and the definition of a start-time scheduler, we can obtain the following end-to-end exit bound.

Corollary 5: Let t_1, t_2, \dots, t_k be a sequence of k start-time schedulers traversed by flow f . For all i ,

$$\begin{aligned} S_{t_k,f,i} &\leq S_{t_1,f,i} + \sum_{x=1}^{k-1} \Delta_{t_x,f,i} \\ E_{t_k,f,i} &\leq S_{t_k,f,i} + \delta_{t_k,f,i} \end{aligned}$$

Notice that the above bounds are independent of the particular scheduling technique, and the capacity of the channel between the schedulers. The only requirement is that the schedulers are start-time schedulers.

VI. MULTI-CHANNEL END-TO-END DELAY

As discussed in Section II, a multi-channel scheduler may reorder packets. Therefore, even if the scheduler is a start-time scheduler, we cannot apply Theorem 3, because this theorem assumes no packet reorder. We next present some general results about a scheduler that reorders packets. We then apply these results to the reorder introduced by a multi-channel scheduler.

We begin with a lemma that bounds the start-time of packets in the presence of limited reorder.

Lemma 1: Consider a start-time scheduler s that may reorder packets. Let f be an input flow of s , and g be the output flow of s derived from f (i.e., g is f with some reorder). Let $p_{g,j} = p_{f,i}$, that is, the j^{th} packet of g is the i^{th} packet of f . Finally, let t be the next scheduler after s .

1) Assume $p_{f,i}$ is the last packet of flow f . Then,

$$S_{t,g,j} \leq S_{s,f,i} + \Delta_{s,f,i}$$

2) Assume $p_{f,(i+m)}$ is the last packet of flow f . Then,

$$S_{t,g,j} \leq S_{s,f,i} + \Delta_{s,f,i} + \frac{\sum_{k=i+1}^{i+m} L_{f,k}}{R_f}$$

The proof Lemma 1 is given in the appendix. This lemma implies that if no packet after $p_{f,i}$ is reordered with $p_{f,i}$, then the start-time increase of $p_{f,i}$ is the same as in Theorem 3. Furthermore, if packets $p_{f,(i+1)}$ up to $p_{f,(i+m)}$ are reordered with $p_{f,i}$, then the start-time of $p_{f,i}$ increases in proportion to the number of bytes in these packets. We next apply this lemma to the limited reorder introduced by a multi-channel scheduler.

Theorem 4: Let s be an multi-channel start-time scheduler, f be an input flow of s , and t be the next scheduler after s . Let

g be the output flow of s derived from f , and let $p_{g,j} = p_{f,i}$. Finally, let $c_s^1 = c_s^{min}$. Then,

$$S_{t,g,j} \leq S_{s,f,i} + \Delta_{s,f,i} + \frac{\frac{L_{f,i}}{c_s^{min}} \cdot \sum_{n=2}^{N_s} c_s^n}{R_f}$$

Proof: Notice that a multi-channel scheduler reorders packets, but with limited reorder. That is, while packet $p_{f,i}$ is being transmitted in a channel, packets from f with index greater than i could be transmitted. For a given channel, for these packets to arrive before $p_{f,i}$, their transmission must take less time than the transmission of $p_{f,i}$. In the worst case, $p_{f,i}$ is transmitted over the slowest channel of capacity c_s^{min} . Thus, it takes $L_{f,i}/c_s^{min}$ seconds to transmit this packet. During this time, channel n can transmit a total of

$$\frac{L_{f,i}}{c_s^{min}} \cdot c_s^n$$

bytes of packets whose index is greater than i . Hence, the packets with index greater than i that arrive to t before $p_{f,i}$ can add to at most

$$\frac{L_{f,i}}{c_s^{min}} \cdot \sum_{n=2}^{N_s} c_s^n$$

bytes.

The result follows from Lemma 1. ■

Corollary 6: Under the same conditions as in Theorem 4, except that all channels of s have the same capacity, the increase in start-time is as follows.

$$S_{t,g,j} \leq S_{s,f,i} + \Delta_{s,f,i} + \frac{(N_s - 1) \cdot L_{f,i}}{R_f}$$

Assume a flow traverses a path of multi-channel schedulers. The start-time of a packet of the flow increases at each hop, and an upper bound on this per-hop increase is given above. Thus, the exit time of the packet is just the start-time of the packet at the first scheduler, plus the increase in start-time at each hop, plus the start-time delay at the last scheduler of the path.

Corollary 6 shows that the packet reorder introduced by a multi-channel scheduler affects the start-time of each packet at the next scheduler. The impact of this increase depends on the values of N_s and $L_{f,i}/R_f$. It is likely that N_s will be quite small, with only a few channels between nodes. However, the term $L_{f,i}/R_f$ may be significant, and for some applications, even a per-hop delay equal to $L_{f,i}/R_f$ is too large. Thus, the impact of packet reorder is significant. In Section VII, we show how packets can be efficiently sorted to prevent this increase in start-time.

We conclude this section by combining Theorem 4 above with the results of Sections III and IV.

Corollary 7: Let s be an multi-channel scheduler, f be an input flow of s , and t be the next scheduler after s . Let g be the output flow of s derived from f , and let $p_{g,j} = p_{f,i}$. Finally, let $c_s^1 = c_s^{min}$.

- 1) Let s emulate a start-time single-channel scheduler r . Then, s is a start-time scheduler, where

$$\delta_{s,f,i} = \delta_{r,f,i} + \frac{(N_s - 1) \cdot L_s^{max} - L_{f,i}}{C_s} + \frac{L_{f,i}}{c_s^{min}}$$

- 2) Let s be a deadline-ordered server with bounded appetite and constant θ . Then, s is a start-time scheduler, where

$$\delta_{s,f,i} = D_{s,f,i} + \frac{N_s \cdot L_s^{max} - L_{s,f,i} - \theta}{C_s} + \frac{L_{s,f,i}}{c_s^{min}} - S_{s,f,i}$$

- 3) In both cases above, the start-time of $p_{f,i}$ at scheduler t is as follows.

$$S_{t,g,j} \leq S_{s,f,i} + \Delta_{s,f,i} + \frac{\frac{L_{f,i}}{c_s^{min}} \cdot \sum_{n=2}^{N_s} c_s^n}{R_f}$$

■

VII. SORTING SCHEDULERS

We next attempt to reduce the end-to-end delay bound of a series of multi-channel schedulers. This large delay bound is caused by the reordering of packets through multiple channels. To reduce the delay bound, each scheduler must restore the original order of the packets of each flow. In this section, we present three techniques to restore this order. We first consider an obvious sorting technique, and show how it fails to reduce the delay bound. Then, we introduce two additional techniques that, although more complex, are effective in reducing the delay bound.

Assume the index of each packet is included in the packet's header. I.e., packet $p_{f,i}$ contains i in its header. The obvious technique to order packets is to sort them as they arrive. Therefore, the queue of each flow f is sorted by packet index. Furthermore, the next packet of f to be forwarded is the packet with the smallest index. However, note that the scheduler may not always forward the packets of f by increasing index. For example, assume that packet $p_{f,(i+1)}$ arrives to the scheduler an instant before packet $p_{f,i}$. It is possible that $p_{f,(i+1)}$ is chosen to be forwarded by the scheduler immediately upon arrival, and thus, it is forwarded before $p_{f,i}$ is received.

For simplicity, we consider the virtual clock protocol. Given that packets arrive out of order, the timestamp assignment method of virtual clock cannot be used. In particular, the timestamp of a packet is not fixed, because it may change with the arrival of packets of the same flow with a smaller index. Thus, to simplify our presentation, we make two assumptions. First, each flow f has a fixed packet size L_f . Second, the scheduler uses flow timestamps [9], as opposed to packet timestamps. With flow timestamps, a packet does not receive a timestamp when it is received. Instead, a single timestamp T_f is maintained for each flow f . We explain flow timestamps in more detail below.

The flow timestamp T_f is updated as follows.

- When a packet from f is received and inserted into the queue of f :
 - If the queue of f is empty, T_f is updated to $\max(A_{f,i}, T_f) + \frac{L_f}{R_f}$.

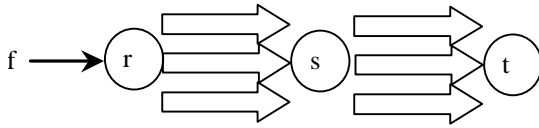


Fig. 4. Sorting schedulers example.

- If the queue of f is not empty, T_f remains unchanged.
- When the packet at the head of the queue is forwarded:
 - If the queue of f is empty, T_f remains unchanged.
 - If the queue of f is not empty, T_f is updated to $T_f + \frac{L_f}{R_f}$.

It is easy to show that if a flow has a non-empty queue, then the exit time of its next packet is at most $T_f + L^{max}/C$. Furthermore, it can also be shown that this exit bound is at most the exit bound of the virtual clock protocol [9].

We next consider an example to show that, even though the queue of f is maintained sorted by index, the end-to-end delay bound is similar to that given in Section VI.

Consider the schedulers in Figure 4. Flow f traverses schedulers r , s , and t . The index of each packet of f corresponds to the order in which the packet was created, and not the order in which it arrives at a scheduler. Schedulers r and s have three channels. Two of these channels have equal capacity, and the remaining channel has a capacity slightly lower than the other two. Since r is a multi-channel scheduler, it may reorder packets while forwarding them to s .

Consider the following sequence of events at scheduler s .

- 1) Scheduler r forwards $p_{f,i}, p_{f,(i+1)}$, and $p_{f,(i+2)}$ at the same time. Packet $p_{f,i}$ is forwarded over the channel with lesser capacity.
- 2) Packets $p_{f,(i+1)}$ and $p_{f,(i+2)}$ arrive at s at time τ . Assume these two packets are forwarded immediately upon arrival. Let $T_{s,f}$ be the flow timestamp of f at s . Let

$$T_{s,f} = \tau + \frac{L_f}{R_f}$$

when $p_{f,(i+1)}$ became the head of the queue of f , and

$$T_{s,f} = \tau + \frac{2 \cdot L_f}{R_f}$$

when $p_{f,(i+2)}$ became the head of the queue of f .

- 3) Let $p_{f,i}$ arrive at s at time $\tau + \epsilon$. Since the queue of f is empty at this time, timestamp $T_{s,f}$ is updated as follows.

$$T_{s,f} \approx \tau + \frac{3 \cdot L_f}{R_f}$$

- 4) Let $p_{f,(i+3)}$ and $p_{f,(i+4)}$ arrive at s after $p_{f,i}$ and before $p_{f,i}$ is forwarded.
- 5) Let $p_{f,i}, p_{f,(i+3)}$, and $p_{f,(i+4)}$ be forwarded from s at the same time, where $p_{f,i}$ is forwarded over the channel with lesser capacity. Let the exit time of $p_{f,i}$ from s be

$$T_{s,f} \approx \tau + \frac{3 \cdot L_f}{R_f}$$

Thus, the exit time of $p_{f,(i+3)}$ and $p_{f,(i+4)}$ is slightly less than this.

Note that scheduler s maintains the queue of f sorted by index. Nonetheless, the exit time of $p_{f,i}$ from s is $\tau + \frac{3 \cdot L_f}{R_f}$, i.e., $\frac{2 \cdot L_f}{R_f}$ larger than its exit time if r were a single-channel scheduler.

A similar sequence of events may occur at scheduler t , as follows. First, we assume packets $p_{f,(i+1)}$ and $p_{f,(i+2)}$ depart t soon after they are received. Next, we consider packets $p_{f,(i+3)}$ and $p_{f,(i+4)}$.

Because $p_{f,(i+3)}$ and $p_{f,(i+4)}$ arrive to t before $p_{f,i}$, the queue of f is empty at this time. We may therefore assume they are forwarded immediately upon arrival. Furthermore, recall that $p_{f,(i+3)}$ and $p_{f,(i+4)}$ arrive at time $\tau + \frac{3 \cdot L_f}{R_f}$. Thus, when $p_{f,(i+3)}$ becomes the head of the queue of f , we have

$$T_{s,f} \approx \tau + \frac{4 \cdot L_f}{R_f}$$

Next, when $p_{f,(i+4)}$ becomes the head of the queue of f , we have

$$T_{s,f} \approx \tau + \frac{5 \cdot L_f}{R_f}$$

Packet $p_{f,i}$ arrives after $p_{f,(i+3)}$ and $p_{f,(i+4)}$ are forwarded. Thus, when $p_{f,i}$ becomes the head of the queue of f , we have

$$T_{s,f} \approx \tau + \frac{6 \cdot L_f}{R_f}$$

Hence, $p_{f,i}$ may be delayed at t until this time.

We have shown that the per-hop delay of $p_{f,i}$ is actually $\frac{3 \cdot L_f}{R_f}$, as opposed to the per-hop delay of $\frac{L_f}{R_f}$ in single-channel schedulers. Therefore, simply sorting the queue of each flow by index is insufficient. To overcome this, we present two more advanced sorting techniques.

Our sorting techniques require that packets have a small additional delay when entering a scheduler. Therefore, we assume that, for each packet $p_{f,i}$ at scheduler s , there is an *eligibility time* $G_{s,f,i}$. Scheduler s will not forward $p_{f,i}$ before this time. Since a packet has not truly ‘‘arrived’’ to the scheduler until its eligibility time, we redefine the start-time as follows.

$$S_{s,f,i} = \max \left(G_{s,f,i}, G_{s,f,(i-1)} + \frac{L_{f,(i-1)}}{R_f} \right)$$

We place the restriction that packets should become eligible in their sorted order, that is, $G_{s,f,i} \leq G_{s,f,(i+1)}$ for all i .

A. Jitter-Reduction Sorting-Schedulers

The first technique is a jitter reduction technique, similar to the technique in [26]. If s is a start-time scheduler, then each packet is labelled with the difference between its exit time and its start-time exit bound. That is, each packet $p_{f,i}$ is labelled with the value $\lambda_{s,f,i}$, where

$$\lambda_{s,f,i} = S_{s,f,i} + \delta_{s,f,i} - E_{s,f,i}$$

If t is the next scheduler after s , then the eligibility time is as follows.

$$G_{t,f,i} = A_{t,f,i} + \lambda_{s,f,i} = E_{s,f,i} + \lambda_{s,f,i} = S_{s,f,i} + \delta_{s,f,i}$$

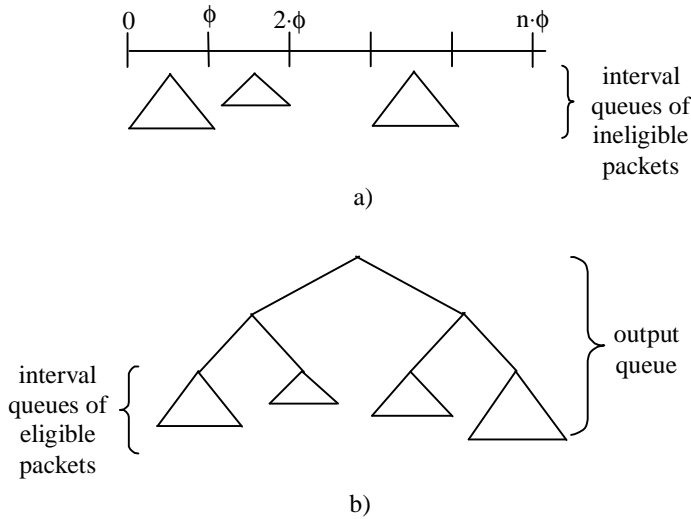


Fig. 5. Output queue of scheduler with eligibility times.

By using the above technique, each packet $p_{f,i}$ becomes eligible at t precisely at its start-time exit bound at s , i.e., at time $S_{s,f,i} + \delta_{s,f,i}$. In all start-time scheduling protocols in the literature, the start-time exit bounds increase with each new packet of the same flow. Hence, packets will become eligible at t in sorted order.

From the above, we can represent the behavior of s and the additional delay λ by a single-channel scheduler s' , where the start-time delay is the same at s and s' . That is, $\delta_{s,f,i} = \delta_{s',f,i}$, for all f and i . Each packet $p_{f,i}$ exits s' precisely at its start-time exit bound. This implies that when a flow traverses a path of multi-channel jitter-reduction schedulers, this is equivalent to traversing a path of single-channel schedulers. In this case, the end-to-end delay is simply obtained from Corollary 5.

One problem to be addressed is that up to M packets can become eligible *at the same time*, where M is the number of flows at the scheduler. Since inserting a packet into an output queue ordered by timestamp takes $O(\log M)$ time, it takes $O(M \cdot \log M)$ time to process all the packets that become eligible. This overhead is too large for a high-speed network implementation. To resolve this, we present a technique similar to the one presented in [24]. This reduces to $O(\log M)$ time the overhead of managing the eligibility of packets. We assume that each packet is assigned a timestamp, and packets are forwarded in order of increasing timestamp.

The data structures needed are shown in Figure 5. Time is divided into small intervals of constant size, as shown in Figure 5(a). Each time interval has a queue of packets. We refer to these queues as *interval queues*. Each interval queue contains packets that become eligible during its interval, and these packets are sorted by timestamp. Furthermore, at most one packet from each flow can be in an interval queue.

Let ϕ be the size of the time intervals. At time $n \cdot \phi$, the queue of interval $((n-1) \cdot \phi, n \cdot \phi]$ becomes eligible, and its interval queue is transferred to the output queue of the scheduler. The output queue has the form of a balanced search tree. The leaves of this tree are the roots of elapsed interval

queues (see Figure 5(b)).

Each flow has a regular queue of packets, independent of the interval queues. This regular queue is sorted by packet index (or equivalently, by eligibility time). The packet with the smallest index is located both in the regular queue and in an interval queue. All other packets of the flow are only in the regular queue.

Note that since each interval queue has at most one packet from each flow, each interval queue has at most M packets, and the output queue has at most M leaves.

When a packet from the output queue is transmitted, the packet is removed from its interval queue and from the regular queue of the flow. Then, the packet at the head of the regular queue of the flow is examined. If this packet is not eligible, it is added to the queue of the appropriate interval. If it is eligible, it is inserted into the output queue as an interval queue of size one.

Receiving a packet is similar. When $p_{f,i}$ is received, its eligibility time is computed. If $p_{f,i}$ does not have the smallest index of f , it is simply inserted into the regular queue of f . Otherwise, let $p_{f,j}$ be the packet of f in an interval queue. This packet is removed from its interval queue. Then, if $p_{f,i}$ is eligible, it is added to the output queue as an interval queue of size one. If $p_{f,i}$ is not eligible, it is added to the appropriate interval queue.

Note that this technique introduces an additional delay of ϕ to the incoming packets. Therefore, the per-hop delay bound of each flow grows by ϕ . However, ϕ can be kept small, close to the transmission time of a packet.

The above technique of sorting packets based on their eligibility time has a couple of disadvantages. The first disadvantage is that the per-hop delay of packet $p_{f,i}$ at s is $\delta_{s,f,i}$, regardless of how lightly loaded the scheduler is. This prevents an application from taking advantage of a low per-hop delay during light network loads. Another disadvantage is that the timestamp of each packet must be computed upon arrival, before the packet is inserted into an interval queue. This timestamp should be the timestamp the packet would receive if it arrived precisely at its eligibility time. Although this is possible with flow timestamps [9], it is not possible in all protocols. For example, in protocols such as WFQ [18], [21], the timestamp that a packet receives when it becomes eligible cannot be computed at the time of the packet's arrival. This is because the timestamp of the packet depends on packets from other flows, and these packets may not have yet arrived to the scheduler.

B. Fixed-Delay Sorting-Schedulers

Our second technique remedies the shortcomings of the first technique. Here, a packet $p_{f,j}$ at scheduler t becomes eligible when all other packets $p_{f,i}$, $i < j$, have already been received at t and are eligible. By delaying packets in this manner, we have the following increase in start-time.

Theorem 5: Let s be a start-time multi-channel scheduler, f be an input flow of s , and t be the next scheduler of f after s . In addition, we assume the following.

- Eligibility times at t are non-decreasing, i.e., for all i , $G_{t,f,i} \leq G_{t,f,(i+1)}$.

• There exists a ψ_t such that for all i , $G_{t,f,i} - A_{t,f,i} \leq \psi_t$. Then, the start-time at t is as follows.

$$S_{t,f,i} \leq S_{s,f,i} + \Delta_{s,f,i} + \psi_t$$

Corollary 8: Let t_1, t_2, \dots, t_k be a sequence of k start-time schedulers traversed by flow f . Each of these schedulers satisfies the conditions of Theorem 5. Then, for all i ,

$$S_{t_k,f,i} \leq S_{t_1,f,i} + \sum_{x=1}^{k-1} (\Delta_{t_x,f,i} + \psi_{t_{(x+1)}})$$

$$E_{t_k,f,i} \leq S_{t_k,f,i} + \delta_{t_k,f,i}$$

Proof: Since s is a start-time scheduler, we have

$$A_{t,f,i} \leq S_{s,f,i} + \delta_{s,f,i}$$

We are given that $G_{t,f,i} \leq A_{t,f,i} + \psi_t$. Hence,

$$G_{t,f,i} \leq S_{s,f,i} + \delta_{s,f,i} + \psi_t \quad (4)$$

Because a packet is not eligible unless it is received, and since eligibility times are increasing, the packets of f become eligible in order of their index. Furthermore, from (4) above, scheduler s plus the delay at t before the packet becomes eligible has the same behavior as a single-channel start-time scheduler s' , where $\delta_{s',f,i} = \delta_{s,f,i} + \psi_t$. Therefore, the start-time at t follows from Theorem 3, and the end-to-end delay follows from Corollary 5. ■

The above technique, although not work-conserving, allows a flow to take advantage of a lightly loaded network. In this case, the packets of a flow may be forwarded at a rate greater than the reserved rate of the flow, and with a per-hop delay lower than the start-time delay δ . However, the technique must be implemented efficiently and with a small value of ψ . We next consider a specific implementation.

At scheduler t , time is divided into intervals of constant size. Let ϕ_t be the length of an interval. We require ϕ_t to be at least the maximum transmission time of a packet, that is,

$$\phi_t \geq \frac{L_s^{max}}{c_s^{min}}$$

Let *interval* n denote the interval $((n-1) \cdot \phi_t, n \cdot \phi_t]$.

Each flow has a queue of packets for each interval. The queue of flow f for interval n is denoted $Q_{t,f,n}$. Packets of f that arrive during interval n are stored in $Q_{t,f,n}$. In addition, $Q_{t,f,n}$ is ordered by packet index. Note that the maximum size of $Q_{t,f,n}$ is $\phi_t \cdot C_s$.

In order to determine the eligibility time of a packet, we must determine whether all packets with lesser indices have been received or not. We next address this issue.

Let $p_{f,j}$ be the packet of f with smallest index that arrived during interval n . Consider any packet $p_{f,i}$, where $i < j$. Due to reorder, $p_{f,i}$ may arrive during interval $n+1$. However, $p_{f,i}$ cannot arrive during interval $n+2$. This is because of the limited reorder and of the lower bound on ϕ_t . Thus, at time $(n+1) \cdot \phi_t$, all packets of f with index less than j have been received.

```

for  $n = 2$  to  $\infty$ ,
    wait until  $clock \geq n \cdot \phi_t$ 
    for every  $f$ , where  $Q_{t,f,(n-1)} \neq \emptyset$ ,
        while  $Q_{t,f,(n-1)} \neq \emptyset$ ,
            let  $p_{f,i}$  have the smallest index in interval  $n-1$ .
            let  $p_{f,j}$  have the smallest index in interval  $n$ .
            if  $i < j$ , then
                forward  $p_{f,i}$  to the scheduler
            else
                forward  $p_{f,j}$  to the scheduler
            end if
        end while
    end for

```

Fig. 6. Transferring packets from their interval queues to the scheduler.

In some cases, we may reduce the time when $p_{f,j}$ becomes eligible as follows. Recall that $p_{f,j}$ is received during interval n . Assume packet $p_{f,k}$, $j < k$, is received during interval $n-1$. This implies that by time $n \cdot \phi_t$, all packets with index less than k have been received. Hence, by time $n \cdot \phi_t$, all packets with index less than j have also been received.

We combine the two observations above to determine when a packet $p_{f,j}$ becomes eligible, as follows.

- 1) Let n be the interval during which packet $p_{f,j}$ arrives.
- 2) If the current time is at least $(n+1) \cdot \phi_t$, then $p_{f,j}$ is eligible.
- 3) If the current time is at least $n \cdot \phi_t$, and there is a packet $p_{f,k}$ that arrived during interval $n-1$, such that $j < k$, then $p_{f,j}$ is eligible.

The sorting technique is therefore as follows. When a packet is received, it is inserted into the appropriate interval queue. At each time $n \cdot \phi_t$, where $n \geq 1$, eligible packets are removed from their interval queues and are transferred to the scheduler. The algorithm to determine which packets become eligible is shown in Figure 6. Since a packet cannot become eligible during the interval in which it is received, inserting and removing packets from an interval queue do not interfere with each other. Thus, inserting and removing packets from interval queues can be done in parallel.

A packet truly “arrives” at the scheduler when it is removed from its interval queue. It is at this time that the scheduler processes the packet in the usual way. E.g., the scheduler assigns a timestamp to the packet and inserts the packet into the output queue. Note that delaying packets in the above manner does not affect the scheduling algorithm in any way, except for a small additional delay in the arrival of packets. Thus, the output queue of the scheduler is unaffected by this technique.

The overhead introduced by the sorting technique is very small. The interval size ϕ_t is approximately the transmission time of a packet. In addition, notice that the processing time of each packet increases only by a constant amount. This is because interval queues are bounded. Finally, let us make the sensible assumption that packets can be inserted into the interval queues and transferred from the interval queues to the scheduler as fast as packets are received. Then, the total

additional delay introduced by the sorting technique is at most $3 \cdot \phi_t$. Thus, $\psi_t \leq 3 \cdot \phi_t$, and the per-hop delay bound of each flow grows by $3 \cdot \phi_t$. This is greater than the increase of ϕ_t of Section VII-A. However, it is applicable to a larger class of schedulers.

VIII. CONCLUDING REMARKS

In this paper, we have considered the problem of providing deterministic quality of service guarantees in a network with multiple channels between nodes. We have shown that any single-channel scheduling protocol can be converted into a multi-channel scheduling protocol without significantly increasing the delay at the scheduling node. This technique is inherently not work-conserving. In addition, we have shown that for schedulers with bounded appetite, any work-conserving single-channel scheduling protocol can be converted to a work-conserving multi-channel scheduling protocol. In addition, due to multiple channels between nodes, the packets of a flow may be reordered. This in turn significantly increases the upper bound on end-to-end delay. We have shown that this increase in delay can be eliminated through the use of efficient sorting techniques.

There are several possible topics for future work in multi-channel scheduling protocols. We discuss a couple of topics below.

We have restricted ourselves to the design of multi-channel schedulers based on existing single-channel schedulers. The bound on the packet exit time of our multi-channel schedulers is very close to the corresponding bound of single-channel schedulers. Thus, we do not believe that a significant reduction in exit time is possible by designing protocols specifically tailored for multiple channels. Nonetheless, a slight improvement may be possible.

In our network model, the packets of each flow are distributed among all the output channels of the scheduler. Another approach would be to have a scheduler for each output channel, and have the packets of each flow be forwarded over a single output channel. Thus, the problem of multiple channels has been reduced to the well-known problem of scheduling over a single channel. However, we have the additional problem of choosing the output channel for each flow. An incorrect assignment of flows to output channels may waste network resources.

Consider the following two examples. First, it is possible that no channel has enough capacity to support a new flow, yet the sum of the remaining capacities of all the channels may be enough to support the flow. As another example, consider those protocols that allow flows to temporarily exceed their packet rate (e.g., WFQ [18], [21]). This allows some flows to make use of capacity unused by other flows. With multiple channels, it is possible that a channel has high utilization, but other channels, although fully reserved, have low utilization. If a flow is forwarded only over a single channel, this prevents the flow in the high utilization channel from taking advantage of the unused capacity in the other channels.

ACKNOWLEDGMENTS

This work was supported in part by the Texas Advanced Research Program through grant number 009741-0139-1999.

REFERENCES

- [1] Baruah, S., Cohen, N., Plaxton, G., Varvel, D., "Proportionate Progress: A Notion of Fairness in Resource Allocation", *Algorithmica*, **15**, pp. 600-625, 1996.
- [2] Baruah, S., Gherke, J., Plaxton, G., "Fast Scheduling of Periodic Tasks on Multiple Resources", in: *Proceedings of the International Parallel Processing Symposium*, 1995.
- [3] Bennet, J.C.R., Partridge, C., Shtetman, N., "Packet Reordering is not Pathological Network Behavior", *IEEE/ACM Transactions on Networking*, **7**(6), December 1999.
- [4] Blanquer J.M., Ozden B., "Fair Queuing for Aggregated Multiple Links", in: *Proceedings of the ACM SIGCOMM Conference*, 2001.
- [5] Cobb J., "Universal Timestamp Scheduling for Real-Time Networks", *Computer Networks*, **31**, 1999.
- [6] Cobb J., "An In-Depth Look at Flow Aggregation", in: *Proceedings of the IEEE International Conference on Network Protocols*, 1999.
- [7] Cobb J., Gouda M., "Flow Theory", *IEEE/ACM Transactions on Networking*, **5**(5), October 1997.
- [8] Cobb J., El-Nahas A., Gouda M., "Time-Shift Scheduling: Fair Scheduling of Flows in High-Speed Networks", *IEEE/ACM Transactions on Networking*, **6**(3), June 1998.
- [9] Cobb J., Gouda M., El-Nahas A., "Flow Timestamps", in: *Proceedings of the Annual Joint Conference on Information Sciences*, 1995.
- [10] Cobb J., M. Lin, "End-to-End Delay Guarantees for Multiple-Channel Schedulers", in: *Proceedings of the IEEE International Workshop on Quality of Service*, 2002.
- [11] D. Ferrari, D. Verma, "A Scheme for Real-Time Channel Establishment in Wide-Area Networks", *IEEE Journal on Selected Areas in Communications*, **8**(3), April 1990.
- [12] Figueira N., Pasquale J., "A Schedulability Condition for Deadline-Ordered Service Disciplines", *IEEE/ACM Transactions on Networking*, **5**(2), April 1997.
- [13] Figueira N., Pasquale J., "Leave-in-Time: A New Service Discipline for Real-Time Communications in a Packet-Switching Data Network", in: *Proceedings of the ACM SIGCOMM Conference*, 1995.
- [14] Fumagalli, A., Cai, J., Chlamtac, I., "A Token Based Protocol for Integrated Packet and Circuit Switching in WDM", in: *Proceedings of the IEEE GLOBECOM Conference*, 1998.
- [15] Golestani, S.J., "A Framing Strategy for Congestion Management", *IEEE Journal on Selected Areas in Communications*, **9**(7), Sept. 1991.
- [16] Golestani, S. J., "A Self-Clocking Fair-Queuing Scheme for Broadband Applications", in: *Proceedings of the IEEE INFOCOM 1994 Conference*.
- [17] Goyal P, Lam S., Vin H., "Determining End-to-End Delay Bounds in Heterogeneous Networks", in: *Proceedings of the NOSSDAV workshop*, 1995.
- [18] Keshav S., "A Control Theoretic Approach to Flow Control", in: *Proceedings of the ACM SIGCOMM Conference*, 1991.
- [19] Liu C., Layland J., "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment", *Journal of the ACM*, **20**, January 1973.
- [20] Moir, M., Ramamurthy, S., "Fair Scheduling of Fixed and Migrating Periodic Tasks on Multiple Resources", in: *Proceedings of the IEEE Real-Time Systems Symposium*, 1999.
- [21] Parekh A. K. J., Gallager R., "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single Node Case", *IEEE/ACM Transactions on Networking*, **1**(3):344-357, June 1993.
- [22] Shreedhar, M. and Varghese, G., "Efficient Fair Queuing Using Deficit Round Robin", in: *Proceedings of the ACM SIGCOMM Conference* 1995.
- [23] Stiliadis, D. Varma, A., "Rate-proportional Servers: a Design Methodology for Fair Queuing Algorithms", *IEEE/ACM Transactions on Networking*, **6**(2), April 1998.
- [24] Stiliadis, D. Varma, A., "A General Methodology for Designing Efficient Traffic Scheduler Shaping Algorithms", in: *Proceedings of the INFOCOM Conference*, 1997.
- [25] Xie G., Lam S., "Delay Guarantee of Virtual Clock Server", *IEEE/ACM Transactions on Networking*, **3**(6) December 1995.
- [26] Verma D. C., Zhang H., Ferrari D., "Delay Jitter for Real-Time Communication in a Packet Switched Network", in: *Proceedings of the TRICOM Conference*, 1991.

- [27] Zhang H., "Service Disciplines for Guaranteed Performance Service in Packet-Switching Networks", *Proceedings of the IEEE*, **93**,(10), October 1995.
- [28] Zheng Q., Shin K.G., "On the Ability of Establishing Real-Time Channels in Point-to-Point Packet-Switched Networks", *IEEE Transactions on Communications*, **42**(2/3/4), 1994.
- [29] Zhang H., Ferrari D., "Rate-Controlled Static Priority Queuing", in: *Proceedings of the INFOCOM Conference*, 1993.

APPENDIX

PROOF OF LEMMA 1

We present a couple of lemmas before presenting the proof of Lemma 1.

Lemma 2: Consider a scheduler t with input flow f . If for any packet $p_{f,j}$, where $1 \leq j < i$, we modify $A_{t,f,j}$ to be at most its previous value, then $S_{t,f,i}$ cannot increase.

Proof: We will reduce the arrival time of packet j incrementally and show that $S_{t,f,i}$ cannot increase.

Consider first reducing $A_{t,f,j}$, but without reducing it below $A_{t,f,(j-1)}$. From the definition of S , reducing the arrival time of a packet $p_{f,j}$ does not increase $S_{t,f,j}$, and by a simple induction on the definition of S , it does not increase $S_{t,f,i}$ for any $i, i > j$.

Consider reducing now $A_{t,f,j}$ below $A_{t,f,(j-1)}$ but not below $A_{t,f,(j-2)}$. For simplicity, we keep the same indices in both cases, i.e., in the reordered case, $p_{f,j}$ arrives to t before $p_{f,(j-1)}$.

We use a hat accent to denote the values after the reduction in $A_{t,f,j}$ and we use non-accented values to denote the values without the reduction in $A_{t,f,j}$. Thus, $A_{t,f,j}$ is the original arrival time of $p_{f,j}$ and $\hat{A}_{t,f,j}$ is the reduced arrival time of $p_{f,j}$.

From the definition of S , and $p_{f,(j-1)}$ being the packet previous to $p_{f,(j+1)}$ in the reordered flow,

$$\hat{S}_{t,f,(j+1)} = \max(\hat{F}_{t,f,(j-1)}, A_{t,f,(j+1)})$$

Without reorder, from the definition of S ,

$$S_{t,f,(j+1)} = \max(F_{t,f,j}, A_{t,f,(j+1)})$$

From the above, we must show that $\hat{F}_{t,f,(j-1)} \leq F_{t,f,j}$. We have four cases to consider.

1) $\hat{A}_{t,f,j} \leq F_{t,f,(j-2)}$

In this case, in the reordered flow, $p_{f,j}$ is the next packet after $p_{f,(j-2)}$. Hence, from the definition of S ,

$$\begin{aligned} \hat{S}_{t,f,j} &= F_{t,f,(j-2)} \\ \hat{F}_{t,f,j} &= F_{t,f,(j-2)} + L_{f,j}/R_f \end{aligned} \quad (5)$$

Within this case we have the following two subcases.

a) $A_{t,f,(j-1)} \leq \hat{F}_{t,f,j}$

Because in the reordered flow $p_{f,j}$ is the packet previous to $p_{f,(j-1)}$, from the definition of S ,

$$\begin{aligned} \hat{S}_{t,f,(j-1)} &= \hat{F}_{t,f,j} \\ \hat{F}_{t,f,(j-1)} &= \hat{F}_{t,f,j} + L_{f,(j-1)}/R_f \end{aligned}$$

From equation (5),

$$\hat{F}_{t,f,(j-1)} = F_{t,f,(j-2)} + L_{f,j}/R_f + L_{f,(j-1)}/R_f$$

In the ordered flow, from the definition of S , F increases by at least L/R_f with each packet. Hence,

$$F_{t,f,j} \geq F_{t,f,(j-2)} + L_{f,j}/R_f + L_{f,(j-1)}/R_f$$

Thus, $\hat{F}_{t,f,(j-1)} \leq F_{t,f,j}$

b) $A_{t,f,(j-1)} > \hat{F}_{t,f,j}$

Because in the reordered flow $p_{f,j}$ is the packet previous to $p_{f,(j-1)}$, from the definition of S ,

$$\hat{S}_{t,f,(j-1)} = A_{t,f,(j-1)}$$

$$\hat{F}_{t,f,(j-1)} = A_{t,f,(j-1)} + L_{f,(j-1)}/R_f \quad (6)$$

In the ordered flow, from the definition of S ,

$$\begin{aligned} F_{t,f,j} &= S_{t,f,j} + L_{f,j}/R_f \\ &\geq F_{t,f,(j-1)} + L_{f,j}/R_f \\ &= S_{t,f,(j-1)} + L_{f,(j-1)}/R_f + L_{f,j}/R_f \\ &\geq A_{t,f,(j-1)} + L_{f,(j-1)}/R_f + L_{f,j}/R_f \end{aligned}$$

Hence, combining the above with Equation (6), $\hat{F}_{t,f,(j-1)} < F_{t,f,j}$.

2) $\hat{A}_{t,f,j} > F_{t,f,(j-2)}$

Since in the reordered flow $p_{f,j}$ is the next packet after $p_{f,(j-2)}$, from the definition of S ,

$$\hat{S}_{t,f,j} = \hat{A}_{t,f,j}$$

$$\hat{F}_{t,f,j} = \hat{A}_{t,f,j} + L_{f,j}/R_f \quad (7)$$

Within, this case, we have the following two subcases.

a) $A_{t,f,(j-1)} \leq \hat{F}_{t,f,j}$

In the reordered flow, since $p_{f,(j-1)}$ follows $p_{f,j}$, from the definition of S ,

$$\begin{aligned} \hat{S}_{t,f,(j-1)} &= \hat{F}_{t,f,j} \\ &= \{\text{from Equation (7)}\} \\ &\quad \hat{A}_{t,f,j} + L_{f,j}/R_f \end{aligned}$$

Also from the definition of S ,

$$\hat{F}_{t,f,(j-1)} = \hat{A}_{t,f,j} + L_{f,j}/R_f + L_{f,(j-1)}/R_f$$

In the ordered flow, from the definition of S and F ,

$$\begin{aligned} F_{t,f,j} &= S_{t,f,j} + L_{f,j}/R_f \\ &\geq F_{t,f,(j-1)} + L_{f,j}/R_f \\ &= S_{t,f,(j-1)} + L_{f,(j-1)}/R_f + L_{f,j}/R_f \\ &\geq A_{t,f,(j-1)} + L_{f,(j-1)}/R_f + L_{f,j}/R_f \end{aligned}$$

Note that $A_{t,f,(j-1)} > \hat{A}_{t,f,j}$ (due to the reorder), and hence $F_{t,f,j} > \hat{F}_{t,f,(j-1)}$.

b) $A_{t,f,(j-1)} > \hat{F}_{t,f,j}$

In the reordered flow, since $p_{f,(j-1)}$ follows $p_{f,j}$, from the definition of S ,

$$\hat{S}_{t,f,(j-1)} = A_{t,f,(j-1)}$$

$$\hat{F}_{t,f,(j-1)} = A_{t,f,(j-1)} + L_{f,(j-1)}/R_f$$

In the ordered flow, from the definition of S ,

$$\begin{aligned} F_{t,f,j} &> F_{t,f,(j-1)} \\ &= S_{t,f,(j-1)} + L_{f,(j-1)}/R_f \\ &\geq A_{t,f,(j-1)} + L_{f,(j-1)}/R_f \end{aligned}$$

Hence, $\hat{F}_{t,f,(j-1)} < F_{t,f,j}$. ■

Lemma 3: Consider a scheduler s and an input flow f . Assume we insert an additional packet q into f after packet $p_{f,(j-1)}$ and before packet $p_{f,j}$. Then, the start-time of $p_{f,i}$ for all $i, i \geq j$, increases by at most L_q/R_f .

Proof: Even though we insert a packet into f , for simplicity, let us maintain the packet indices as before the packet was inserted. Therefore, the sequence of packet arrivals is $p_{f,(j-1)}, q, p_{f,j}$. Values corresponding to the case where q is inserted have a hat accent. Non-accented values either do not change with the insertion of q or correspond to the case where q is not inserted. We first show that $S_{f,j}^s$ increases by at most L_q/R_f , i.e.,

$$\hat{S}_{f,j}^s \leq S_{f,j}^s + L_q/R_f$$

We have two cases.

- 1) $F_{f,(j-1)}^s < A_{f,j}^s$
From the definition of S ,

$$\hat{S}_q^s = \max(F_{f,(j-1)}^s, A_q^s)$$

Since $A_q^s \leq A_{f,j}^s$ and $F_{f,(j-1)}^s < A_{f,j}^s$,

$$\hat{S}_q^s \leq A_{f,j}^s$$

Thus, for packet $p_{f,j}$ which follows q ,

$$\begin{aligned} \hat{S}_{f,j}^s &= \max(\hat{F}_q^s, A_{f,j}^s) \\ &= \max(\hat{S}_q^s + L_q/R_f, A_{f,j}^s) \\ &\leq \max(A_{f,j}^s + L_q/R_f, A_{f,j}^s) \\ &= A_{f,j}^s + L_q/R_f \end{aligned}$$

Note that since $F_{f,(j-1)}^s < A_{f,j}^s$ in f , then without inserting q ,

$$S_{f,j}^s = A_{f,j}^s$$

Hence, $\hat{S}_{f,j}^s \leq S_{f,j}^s + L_q/R_f$.

- 2) $F_{f,(j-1)}^s \geq A_{f,j}^s$

In this case, from the definition of S ,

$$S_{f,j}^s = \max(F_{f,(j-1)}^s, A_{f,j}^s) = F_{f,(j-1)}^s$$

Since $A_q^s \leq A_{f,j}^s \leq F_{f,(j-1)}^s$, from the definition of S ,

$$\hat{S}_q^s = \max(F_{f,(j-1)}^s, A_q^s) = F_{f,(j-1)}^s$$

For the next packet $p_{f,j}$ after q ,

$$\begin{aligned} \hat{S}_{f,j}^s &= \max(\hat{F}_q^s, A_{f,j}^s) \\ &= \max(\hat{S}_q^s + L_q/R_f, A_{f,j}^s) \\ &= \max(F_{f,(j-1)}^s + L_q/R_f, A_{f,j}^s) \\ &= F_{f,(j-1)}^s + L_q/R_f \end{aligned}$$

Hence, combining with the above, $\hat{S}_{f,j}^s \leq S_{f,j}^s + L_q/R_f$. ■

We next consider the increase in S for packets after $p_{f,j}$. From the definition of S , for all $i, i > 1$,

$$S_{f,i}^s = \max(S_{f,(i-1)}^s + L_{f,(i-1)}/R_f, A_{f,i}^s)$$

If $S_{f,(i-1)}^s$ increases by a value α , then $S_{f,i}^s$ increases by at most α . We have shown that $S_{f,j}^s$ increases by at most L_q/R_f . Hence, by a simple induction proof, for all $i, i \geq j$, $S_{f,i}^s$ increases by at most L_q/R_f . ■

We next present the proof of Lemma 1.

Proof: Consider first part one. Let h correspond to a possible output of s , where each packet of f is delayed its maximum. That is, for all i , $p_{h,i} = p_{f,i}$, and

$$A_{t,h,i} = S_{s,f,i} + \delta_{s,f,i}$$

Notice that packet order is preserved, since we made the assumption in Section V that start-time exit bounds are increasing. Also, note that for all packets of g , the corresponding packet of h arrives no later than the packet of g . Furthermore, from the results of [6], [13], for all i ,

$$S_{t,h,i} \leq S_{s,f,i} + \Delta_{s,f,i}$$

We manipulate flow h until we obtain our desired flow g . Each manipulation cannot increase the start time of packet $p_{f,i}$.

Only packets $p_{g,1}$ up to $p_{g,j}$ affect the computation of $S_{t,g,j}$. Thus, we assume g simply consists of packets $p_{g,1}$ up to $p_{g,j}$. Let h' be obtained from h by deleting all those packets which are not in g . I.e., h' contains only packets $p_{g,1}$ up to $p_{g,j}$. Note that $p_{f,i}$, which is $p_{g,j}$, is the last packet of g . Also, since we assumed that $p_{f,i}$ is the last packet of f , it is also the last packet of h and h' .

It is obvious from a simple induction proof and the definition of S , that removing any packet from a flow cannot increase the value of S for any packet in the flow. Hence, the start-times of the packets of h' have not increased beyond those of the corresponding packets of h .

Next, note that the packets of g and h' are not in the same order. Through a repeated application of Lemma 2, we can reduce the arrival times of the packets of h' (except the last one, $p_{f,i}$), and make their arrival times equal to the corresponding packets of g . This results in flow h'' . From the lemma, the start time of $p_{f,i}$ in h'' does not increase.

Finally, the arrival time of $p_{f,i}$ must also be reduced to the arrival time of $p_{g,j}$. From the definition of S , reducing the arrival time of a packet does not increase its start-time. Hence,

$$S_{t,g,j} \leq S_{t,h,i} \leq S_{s,f,i} + \Delta_{s,f,i}$$

Consider now part two. If no packet after $p_{f,i}$ is reordered with $p_{f,i}$, the start time of $p_{f,i}$ at t is the same as in part one. However, the m packets after $p_{f,i}$ could be reordered with $p_{f,i}$ and arrive to t before $p_{f,i}$. By a repeated application of Lemma 3 and part one we obtain,

$$S_{t,g,j} \leq S_{t,h,i} \leq S_{s,f,i} + \Delta_{s,f,i} + \frac{\sum_{k=i+1}^{i+m} L_{f,k}}{R_f}$$

■