# End-to-End Delay Guarantees for Multiple-Channel Schedulers

Jorge A. Cobb and Miaohua Lin
Department of Computer Science
The University of Texas at Dallas
Richardson, TX 75083-0688
{cobb,miaohua}@utdallas.edu

***Abstract*—Consider a network in which adjacent nodes exchange messages via multiple communication channels. Multiple channels between adjacent nodes are desirable due to their cost effectiveness and improved fault-tolerance. We consider the problem of providing deterministic quality of service guarantees in this network. We show that any scheduling protocol designed for a single channel can be converted into a multiple-channel scheduling protocol without significantly increasing the delay at the scheduling node. However, because there are multiple channels between adjacent nodes, the packets of a flow may be reordered. This in turn significantly increases the upper bound on the end-to-end delay of the flow. We show how this increase in delay can be avoided through the use of efficient sorting techniques.**

## I. Introduction

Packet scheduling protocols that provide deterministic quality of service guarantees flourished in the previous decade (for a survey, see [23]). 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 [16].

In [16], all tasks share a single resource. In the last few years, there has been significant work in the scheduling of periodic tasks over *multiple* resources [1], [2], [17]. Even though the theory of periodic task scheduling over multiple resources has began to show promise, there has been little work to develop packet scheduling protocols over multiple channels between network nodes. This is due in part to the belief that multiple channels between nodes is either not practical 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 inherent parallelism in the network. One cause 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 its nodes. The reason for this approach is that it often reduces equipment and trunk costs. That is, it is often more cost effective to put two components in parallel than to use one component that has twice the speed. In addition, it improves 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, to reflect the changes in network load over time [11]. If there is a significant load between two nodes in the network, it is possible that a single light-path may not provide enough bandwidth between them, which calls for the establishment of additional light-paths between these nodes. Thus, multiple communication channels may be established between two nodes (for more examples of multiple-channel systems, see [4].)

Given the evidence of multiple channels between nodes presented above, it is likely that multiple channels will continue to exist. Therefore, it is reasonable to assume that if a guaranteed quality of service protocol is deployed on a global scale, it will likely traverse at some point network nodes with multiple channels between them. Thus, the problem of scheduling packets for guaranteed quality of service in the presence of multiple channels must be studied.

In [4], the first scheduling protocol for guaranteed service over multiple channels is presented. The scheduling protocol assigns timestamps to packets in the same way as in weighted-fair-queuing [15], [18], and packets are forwarded to channels in order of increasing timestamp. However, no other scheduling protocols were considered, and the end-to-end delay of a series of multiple-channel nodes was not considered.

In this paper, we present general techniques to develop multiple-channel scheduling protocols. In particular, we show two techniques which take a single-channel scheduling protocol and convert it to a multiple-channel protocol. In addition, we consider the end-to-end delay of packets through a series of multiple-channel nodes. We observe that the addition of multiple channels may significantly increase the end-to-end delay due to packet reorder. We show how this increase in delay can be prevented through efficient sorting techniques.

Due to space limitations, some proofs have been omitted.

The omitted proofs may be found in [7].

## II. Single-Channel Network Model

In this section, we define the network model for single-channel scheduling, and also define the quality of service that the model assigns to each flow of packets. We base our model on the models of [5] and [10]. We present multiple-channel scheduling in Section III.

A *network* is a set of nodes interconnected by point-to-point communication channels. For every pair of nodes $a$ and $b$, there is at most one channel from $a$ to $b$ and at most one channel from $b$ to $a$. Every output channel in a node is equipped with a scheduler. From the input channels, the scheduler receives packets from flows whose path include the output channel of the scheduler. The scheduler then chooses the transmission order and transmission time of these packets over its output channel. This is shown in Figure 1.

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

| | |
|---|---|
| $R_f$ | bandwidth reserved for flow $f$. |
| $p_{f,i}$ | $i^{th}$ packet of $f$, $i \geq 1$. |
| $L_{f,i}$ | length of packet $p_{f,i}$. |
| $L_{f,i}^{max}$ | maximum of $L_{f,j}$, where $1 \leq j \leq i$. |
| $L_s^{max}$ | maximum packet length 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 bandwidth of scheduler $s$. |

Consider a scheduler $s$ and a flow $f$. We define the *start-time* $S_{s,f,i}$ and *finish-time* $F_{s,f,i}$ of packet $p_{f,i}$ at scheduler $s$ as follows [5][10]. 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 $s$, and $F_{s,f,i}$ is the time at which the last bit of $p_{f,i}$ is forwarded by $s$. More formally, let $f$ be an input flow of scheduler $s$. Then,

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

Each scheduler $s$ forwards the packets of each input flow $f$ at a rate of at least $R_f$. Therefore, for each packet $p_{f,i}$, its exit time from the scheduler is close to its start time, $S_{s,f,i}$. We refer to these schedulers as *start-time schedulers* [5], [6], [10], [14].

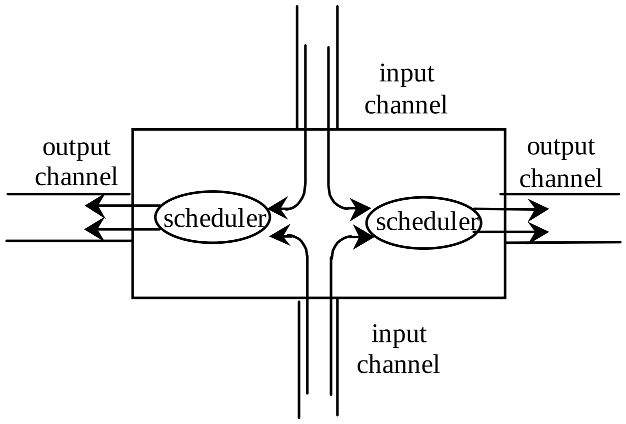

Fig. 1. Output channels and their schedulers.

*Definition 1:* 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 deadline* of $p_{f,i}$ at $s$. Throughout the paper, we assume all schedulers are start-time schedulers.

The start-time delay is broad enough to encompass 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-queuing protocols [18], [21]. Another example is the real-time channel model, [8], [24] 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 1:* 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 $i$,

$$S_{t,f,i} \leq S_{s,f,i} + \Delta_{s,f,i} \tag{1}$$

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

This bound was shown in [6], [10], and it also follows from the results in [24]. From induction and the definition of a start-time scheduler, we can obtain the following end-to-end delay bound.

*Corollary 1:* Let $t1, t2, \ldots, tk$ be a sequence of $k$ start-time schedulers traversed by flow $f$. For all $i$,

$$S_{tk,f,i} \leq S_{t1,f,i} + \sum_{x=1}^{k-1} \Delta_{tx,f,i}$$

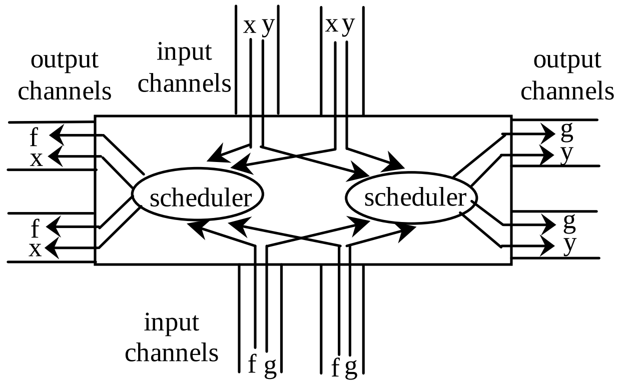

Fig. 2. Multiple output channels per node.

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

■

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

## III. $N$-Channel Network Model

We next enhance the network model to include multiple channels between nodes. This is shown in Figure 2. All the channels to the same neighboring node are managed by a single scheduler. For simplicity, we assume all the channels managed by the same scheduler have equal bandwidth.

*Definition 2:* A scheduler $s$ is an *$N$-channel scheduler* if it has a total of $N$ output channels. An $N$-channel scheduler has capacity $C$ if and only if each of its $N$ channels has capacity $C/N$. ■

Since all channels of a scheduler lead to the same node, and they all have the same bandwidth, it is irrelevant which channel is used to forward a packet. Hence, when any of the channels becomes idle, the scheduler will pick the next packet from its queue and forward it to the idle channel.

Note that packets from a flow can become reordered along their path to the destination. This is due to the fact that multiple output channels can become idle at the same time, and packets $p_{f,i}$ and $p_{f,(i+1)}$ can be forwarded to a pair of channels at the same time. If $L_{f,i} > L_{f,(i+1)}$, then $p_{f,(i+1)}$ will arrive earlier to the next node than $p_{f,i}$, and hence, be reordered.

Each node has the option to either process the packets of each flow in FCFS order, or sort the packets of each flow back to their original order. We will consider both of these cases. In both cases, the start-time of each packet is computed using the same formulae as before. However, the index of the packet may change from one scheduler to the next due to reorder.

## IV. $N$-Channel End-to-End Delay

As discussed above, an $N$-channel scheduler may reorder packets. Therefore, even if the scheduler is a start-time scheduler, we cannot apply Theorem 1, because this theorem assumes no packet reorder. We next present some general results about a scheduler which reorders packets. We then apply these results to the reorder introduced by an $N$-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 $s$ which may reorder packets. Let $f$ be an input flow of $s$, and denote the output flow corresponding to $f$ as flow $g$ (which may be reordered). Let $p_{g,j} = p_{f,i}$, that is, the $j^{th}$ packet of the output flow $g$ is the $i^{th}$ packet of flow $f$. 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,

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

■

Lemma 1 implies that if no packet after $p_{f,i}$ is reordered with $p_{f,i}$, then the increase in start-time of $p_{f,i}$ is the same as in Theorem 1. 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 an $N$-channel scheduler.

*Theorem 2:* Let $s$ be an $N$-channel start-time scheduler, $f$ be an input flow of $s$, and $t$ be the next scheduler of $f$ after $s$. Let $g$ be the output flow of $s$ corresponding to $f$, and let $p_{g,j} = p_{f,i}$, that is, the $j^{th}$ packet of the output flow $g$ is the $i^{th}$ packet of flow $f$. Then,

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

*Proof:* Notice that an $N$-channel reorders packets, but with limited reorder. That is, for a packet $p_{f,i}$, while it is being transmitted in a channel, packets from $f$ after $p_{f,i}$ could be transmitted. These packets can add to no more than $(N-1) \cdot L_{f,i}$ bytes. This is because we have only $N-1$ additional channels, and the packets in these channels must exit before $p_{f,i}$ exits. Since packets received after $p_{f,i}$ do not affect the start-time of $p_{f,i}$, we can consider these additional packets to be the last packets of the flow. Hence, from Lemma 1

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

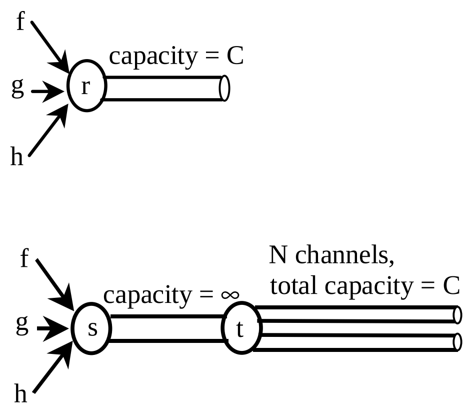

Fig. 3. N-Channel emulation of a 1-channel scheduler.

■

Theorem 2 shows that the packet reorder introduced by an $N$-channel scheduler affects the start-time of each packet at the next scheduler. The impact of this increase depends on the values of $N$ and $L_{f,i}/R_f$. It is likely that $N$ 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 the start-time.

## V. 1-Channel Emulation

We have considered the increase in start-time across an $N$-channel server. However, we have not considered how the scheduler chooses packets to be forwarded. Although scheduling protocols tailored specifically for multiple channels can be developed, one obvious question to ask is if existing scheduling protocols can be adapted to multiple channels. In particular, if *all* single-channel scheduling protocols can be adapted to multiple channels. We next address this question.

Consider Fig. 3. In this figure we have three schedulers. Scheduler $r$ is a 1-channel scheduler with capacity $C$. Scheduler $s$ is a 1-channel scheduler whose output channel has infinite capacity and whose input flows are the same as those of $r$. Scheduler $t$ is an $N$-channel scheduler of capacity $C$.

The behavior of these schedulers is as follows. Scheduler $s$ will forward to $t$ each packet at exactly the same time at which the same packet is forwarded by $r$. Notice that since the capacity of $s$ is infinity, this implies that $s$ is non-work conserving, and that once $s$ forwards a packet, the packet is received immediately by $t$. Also notice that after $s$ forwards a packet of size $L$ to $t$, it may not forward the next packet until $L/C$ seconds later. Finally, $t$ is work-conserving. It simply queues the packets received from $s$, and it forwards packets in FCFS order to its $N$ output channels. In a sense, schedulers $s$ and $t$ emulate the behavior of scheduler $r$, except that the packets are distributed over $N$ channels.

We next compare the behavior of these two systems.

*Theorem 3:* Let $r$ be a 1-channel scheduler with capacity $C$. Assume a 1-channel scheduler $s$ has capacity $\infty$ and has the same input flows as $r$. Let $s$ forward packets in the same order as $r$ and at exactly the same time as $r$. Assume $t$ is a work-conserving, FCFS, $N$-channel scheduler with capacity $C$. Let $t$ be after $s$. Then, for all input flows $f$ and for all $i$,

$$E_{t,f,i} - E_{r,f,i} \leq \frac{(N-1)\cdot L_t^{max}}{C} + \frac{(N-1)\cdot L_{f,i}}{C}.$$

*Proof:* If packet $p_{f,i}$ arrives at $t$ and is sent immediately to a channel, (i.e., no queuing at $t$), then the extra delay suffered at $t$ is at most $\frac{L_{f,i}}{C/N} - \frac{L_{f,i}}{C}$, which is equal to,

$$\frac{(N-1)\cdot L_{f,i}}{C}$$

Consider now that when $p_{f,i}$ arrives at $t$, all channels are occupied (i.e., $p_{f,i}$ is queued at $t$). Let $p_{g,j}$ be the latest packet before $p_{f,i}$ such that $p_{g,j}$ suffered no queuing at $t$ (i.e. there was one empty channel at the time $p_{g,j}$ arrives at $t$). Thus, from $A^+_{t,g,j}$ up to $A_{t,f,i}$ all channels are occupied.

Let the *channel backlog* be the sum over all channels of the number of bits which remain to be transmitted from the packet currently in the channel. Let $Q$ be the channel backlog at time $A^-_{t,g,j}$. Note that $Q \leq (N-1)\cdot L_t^{max}$, because there is one empty channel at this time, and each channel can have at most $L_t^{max}$ bits.

Let $B$ be the total number of bits in the sequence of packets arriving at $t$ starting from time $A_{t,g,j}$ and ending at time $A^-_{t,f,i}$. That is, we start with $p_{g,j}$ and end with the packet previous to $p_{f,i}$. Since packets arrive at $t$ at a rate of at most $C$, the length of the interval $[A_{t,g,j}, A_{t,f,i})$ is at least

$$\frac{B}{C}$$

Also, during $[A_{t,g,j}, A_{t,f,i})$, since all channels are busy during this interval, the number of bits forwarded is

$$C\cdot(A_{t,f,i} - A_{t,g,j}) \geq C\cdot\frac{B}{C} = B$$

Hence, at time $A^-_{t,f,i}$ the channel backlog plus the packets in the queue ahead of $p_{f,i}$ is at most $Q$ bits, that is, at most the channel backlog at time $A^-_{t,g,j}$.

These $Q$ bits prevent $p_{f,i}$ from going into a channel. How long is it before $p_{f,i}$ is forwarded to a channel? Notice that all channels will be busy until $p_{f,i}$ is forwarded to a channel. Hence, in the worst case, all $Q$ bits are in the channels, and they are evenly spread over all channels. Hence, $p_{f,i}$ begins transmission no later than time

$$A_{t,f,i} + \frac{\frac{Q}{N}}{\frac{C}{N}} = A_{t,f,i} + \frac{Q}{C}$$

The exit time, $E_{t,f,i}$, is thus,

$$E_{t,f,i} \leq A_{t,f,i} + \frac{Q}{C} + \frac{L_{f,i}}{\frac{C}{N}}$$

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

$$E_{t,f,i} - E_{r,f,i} \leq A_{t,f,i} + \frac{Q}{C} + \frac{L_{f,i}}{\frac{C}{N}} - (A_{t,f,i} + \frac{L_{f,i}}{C})$$

Thus,

$$E_{t,f,i} - E_{r,f,i} \leq \frac{Q}{C} + \frac{N \cdot L_{f,i}}{C} - \frac{L_{f,i}}{C}$$

Since $Q \leq (N-1) \cdot L_t^{max}$,

$$E_{t,f,i} - E_{r,f,i} \leq \frac{(N-1) \cdot L_t^{max}}{C} + \frac{N \cdot L_{f,i}}{C} - \frac{L_{f,i}}{C}$$

Reducing, we obtain,

$$E_{t,f,i} - E_{r,f,i} \leq \frac{(N-1) \cdot L_t^{max}}{C} + \frac{(N-1) \cdot L_{f,i}}{C}$$

■

The above theorem shows that we can implement an $N$-channel scheduler based upon any 1-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. It is less than the time required for one of the channels to transmit two packets. This is not significant if the channel bandwidth is large.

Note that the bound on the exit-time difference is similar to the bound obtained in [4]. However, the bound in [4] is specific to weighted-fair queuing. In our case, the bound applies to *any* scheduler, regardless of type. For example, a stop-and-go scheduler [12] and all the schedulers in the family of rate-proportional servers [19] exhibit this bound.

The behavior of both $s$ and $t$ can be combined into a single scheduler which emulates $r$. In this case, we say that $s$ emulates $r$.

*Definition 3:* An $N$-channel scheduler $s$ *emulates* a 1-channel scheduler $r$ if and only if all of the following hold.

- The capacity of both $r$ and $s$ is the same.
- $s$ forwards packets in the same order as $r$ does.
- A packet in $s$ is eligible to be forwarded when the packet is forwarded by $r$.
- A channel cannot be idle while there are eligible packets in $s$.

■

Thus far, we placed no restrictions on the type of scheduler $r$. Next, we focus on a start-time scheduler.

*Corollary 2:* Let $s$ be an $N$-channel scheduler which emulates a 1-channel start-time scheduler $r$. Let $C$ be their capacity.

1) For all input flows $f$ and all $i$,

$$\delta_{s,f,i} \leq \delta_{r,f,i} + \frac{(N-1) \cdot L_s^{max}}{C} + \frac{(N-1) \cdot L_{f,i}}{C}$$

2) Let $f$ be an input flow of $s$, $g$ be the output flow of $s$ corresponding to $f$ (which may be reordered), and $p_{g,j} = p_{f,i}$.

$$\begin{aligned} S_{t,g,j} \quad \leq \quad & S_{s,f,i} + \Delta_{r,f,i} + \frac{(N-1) \cdot L_s^{max}}{C} + \\ & \frac{(N-1) \cdot L_{f,i}}{C} + \frac{(N-1) \cdot L_{f,i}}{R_f} \end{aligned}$$

where $t$ is the next scheduler in the path of $f$ after $s$.

■

Part one of Corollary 2 shows that if the 1-channel scheduler being emulated is a start-time scheduler, then the resulting $N$-channel scheduler is also a start-time scheduler, whose start-time delay is slightly higher than that of the emulated scheduler. Part two shows the increase in the start-time of a packet as it traverses the $N$-channel scheduler. This includes the additional term

$$\frac{(N-1) \cdot L_{f,i}}{R_f}$$

due to the reordering of packets. As we mentioned earlier, we can reduce this term by efficiently sorting packets, which is discussed in Section VII.

## VI. Bounded Appetite Servers

A disadvantage of the emulation of a 1-channel scheduler is that the resulting $N$-channel scheduler is not work-conserving. For example, assume the scheduler and its channels are idle, and the scheduler receives two packets at the same time. After the scheduler forwards the first packet, it cannot forward the second packet until $L/C$ seconds later, where $L$ is the size of the first packet. We would like to consider work-conserving schemes to schedule packets over $N$ channels. In particular, we would like to be as general as possible, and to be close to existing protocols for 1-channel schedulers.

We consider schedulers which assign deadlines to packets and forward packets in order of increasing deadline. This 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 [13] and weighted fair queuing [15], [18], 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 $p_{g,j}$.

To guarantee that packets exit by their deadlines, we require the following bound on the occurrence of deadlines, which was introduced in [9].

*Definition 4:* A scheduler $s$ with capacity $C$ satisfies the *bounded appetite property* if and only if, for all intervals $[a,\, b]$, the packets which arrive during the interval and whose deadlines are at most $b$ sum to at most $(b-a)\cdot C$ bytes. That is,

$$\left(\sum f,i\;:\;(a \le A_{s,f,i} \le b) \wedge (D_{s,f,i} \le b)\;:\;L_{s,f,i}\right) \le (b-a)\cdot C$$

where $D_{s,f,i}$ is the deadline of packet $p_{f,i}$ at scheduler $s$. ■

Without loss of generality, we assume that $D_{s,f,i} \ge A_{s,f,i} + L_{f,i}/(C/N)$ for an $N$-channel server $s$. We next consider the exit time from a work conserving $N$-channel scheduler with bounded appetite.

*Theorem 4:* Consider an $N$-channel, work-conserving scheduler $s$ which satisfies the bounded appetite property and has capacity $C$. Let $s$ forward packets in order of their deadlines. Then, for all input flows $f$ and for all $i$,

$$E_{s,f,i} \le D_{s,f,i} + \frac{N\cdot L_s^{max}}{C}$$

*Proof:* The proof is similar to that of Theorem 3. Assume when packet $p_{f,i}$ arrives at $s$ it goes directly into a channel. Then,

$$E_{s,f,i} = A_{s,f,i} + \frac{L_{f,i}}{C/N} \le D_{s,f,i}$$

Consider now when $p_{f,i}$ arrives at $s$, and all channels are occupied (i.e., $p_{f,i}$ is queued at $s$). Let $\tau$ 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 $D_{s,g,j} > D_{s,f,i}$, and all packets forwarded in the interval $(\tau, A_{s,f,i}]$, have deadlines at most $D_{s,f,i}$.

Consider the first case. Since $p_{f,i}$ arrives at time $A_{s,f,i}$, only packets with deadlines at most $D_{s,f,i}$ are forwarded during the interval $[\tau, E_{s,f,i}]$. Furthermore, any packet forwarded during the interval $[\tau, E_{s,f,i}]$ must arrive at $\tau$ or later, because $p_{g,j}$ has no queuing delay.

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

$$(D_{s,f,i} - \tau)\cdot C \qquad (2)$$

bytes. Note that only these packets and the channel backlog at time $\tau^-$ can be forwarded before $p_{f,i}$.

The channel backlog at time $\tau^-$ can be at most $(N-1)\cdot L_s^{max}$ bits, since there was an empty channel when $p_{g,j}$ arrived. Furthermore, from the definition of $p_{g,j}$, all channels are busy until $p_{g,j}$ is forwarded.

The case which delays the forwarding of $p_{f,i}$ the most is when all channels *at the same time* finish transmitting their packets. In this way, no channel will become free until the very last moment. Hence, $p_{f,i}$ has an exit time as follows.

$$E_{s,f,i} \le \tau + \frac{(D_{s,f,i} - \tau)\cdot C}{C} + \frac{(N-1)\cdot L_s^{max}}{C}$$

Reducing we obtain,

$$E_{s,f,i} \le D_{s,f,i} + \frac{(N-1)\cdot L_s^{max}}{C}$$

A similar reasoning applies to the second case, except that $p_{g,j}$ is not counted in (2) above. Hence, packets in the channel backlog at time $\tau$ (including $p_{g,j}$) plus those packets counted in (2) may exit before $p_{f,i}$ is forwarded to a channel. The backlog at time $\tau$ is at most $N\cdot L_s^{max}$. Hence, the exit time is as follows.

$$E_{s,f,i} \le D_{s,f,i} + \frac{N\cdot L_s^{max}}{C}$$

■

It was shown in [9] that weighted fair queuing and virtual clock are bounded appetite scheduling protocols, and their deadlines satisfy the following.

$$D_{s,f,i} \le S_{s,f,i} + \frac{L_{f,i}}{R_f}$$

We therefore have the following corollary.

*Corollary 3:* Let $s$ be a work-conserving, $N$-channel server with capacity $C$. Let $s$ assign timestamps to packets as weighted fair queuing or virtual clock, and forward packets to the channels in order of increasing timestamp. Let $f$ be an input flow of $s$, let $g$ be the output flow corresponding to $f$ (which could be reordered) and let $p_{g,j} = p_{f,i}$. Then,

$$\begin{aligned}
\delta_{s,f,i} &\le \frac{L_{f,i}}{R_f} + \frac{N\cdot L_s^{max}}{C}\\
E_{s,f,i} &\le S_{s,f,i} + \frac{L_{f,i}}{R_f} + \frac{N\cdot L_s^{max}}{C}\\
S_{t,g,j} &\le S_{s,f,i} + \frac{L_{f,i}^{max}}{R_f} + \frac{N\cdot L_s^{max}}{C} + \\
&\quad \frac{(N-1)\cdot L_{f,i}}{R_f}
\end{aligned}$$

■

Corollary 3 provides a slightly tighter bound on the exit time of a packet than the bound given in [4] for a multiple-channel weighted fair-queuing scheduler. More importantly, however, is that the corollary may be applied to any scheduler which satisfies the bounded appetite property. In addition to the typical timestamp protocols of virtual clock [21] and weighted fair queuing [15], [18], protocols such as stop-and-go [12] and rate-controlled static priority queuing [25] have bounded appetite [9]. Although these latter protocols are not work-conserving, they define 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 scheduler at its eligibility time, then the above corollary also applies to these protocols.

## VII. Sorting Servers

We next attempt to reduce the increase in start-time shown in Theorem 2. This increase is caused by the reordering of packets through the $N$-channel scheduler. To reduce this start-time, the next scheduler must restore packets to their original order. We consider a couple of techniques to perform this ordering.

Both techniques assume that, for each packet $p_{f,i}$ at a scheduler $s$, there is an *eligibility time* $G_{s,f,i}$, and the scheduler 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.

$$S_{s,f,i} = max(G_{s,f,i}, F_{s,f,(i-1)})$$

$$F_{s,f,i} = S_{s,f,i} + \frac{L_{f,i}}{R_f}$$

The first technique is a jitter reduction technique, similar to the technique in [22]. If $s$ is a start-time scheduler, then each packet is timestamped with the difference between its exit time and its start-time deadline. That is, each packet $p_{f,i}$ is timestamped with the value $\lambda_{s,f,i}$, where

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

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

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

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

The above technique has the disadvantage of not being work-conserving. In particular, a flow cannot exceed its reserved rate and take advantage of unused bandwidth, since the schedulers along its path will only forward its packets at the reserved rate.

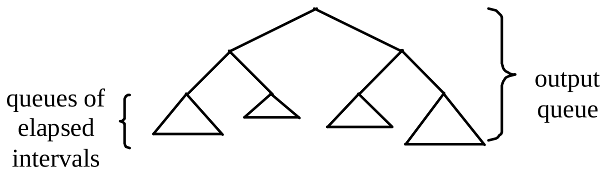

Fig. 4. Output queue of scheduler with eligibility times.

In our second technique, a packet at $t$ becomes eligible no earlier than its arrival time plus the maximum transmission time of one packet. In this way, when packet $p_{f,i}$ is eligible at $t$, all other packets $p_{f,j}$, $j < i$, have already been received at $t$. We assume that each packet $p_{f,i}$ includes its index $i$ in its header. Scheduler $t$ sorts the packets of $f$ in order of their indices before forwarding them to the output channels. Hence, the increase in start-time due to reorder is prevented.

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

- There exists a $\psi$ such that for all $f$ and $i$,

$$A_{t,f,i} + \frac{N \cdot L_s^{max}}{C} \leq G_{t,f,i} \leq A_{t,f,i} + \psi$$

- $t$ sorts packets of $f$ in order of increasing index, and forwards them in order to the output channels.
- A packet $p_{f,i}$ is not eligible at $t$ if there is a packet $p_{f,j}$, $j < i$, which is also not eligible.

Then, the start-time at $t$ is as follows.

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

■

The above technique, although not work-conserving, does allow the packets of a flow to be forwarded at a rate greater than the flow's reserved rate. However, it must be implemented efficiently and with a small value of $\psi$.

We borrow the sorting technique of [20]. Here, we assume packets are assigned timestamps, and are forwarded in timestamp order. The major problem is that packets cannot be added to the output queue until their eligibility time. If one packet from each flow becomes eligible at the *same time,* than all these packets must be added at once to the output queue. If there are $M$ flows, this takes O($M \cdot \log M$) time, which is excessive.

To avoid this, time is divided into adjacent fixed-size intervals of length $\theta$. For each interval $(n \cdot \theta,\ (n+1) \cdot \theta]$, there is a packet priority queue, arranged in timestamp order. The queue of the interval only contains packets which become eligible during this interval. At time $(n + 1) \cdot \theta$, the queue elapses, and it is inserted into the output queue.

The output queue is a balanced search tree, whose leaves are the roots of elapsed interval queues (see Figure 4). To limit the size of the output queue, only one packet per flow can be in an interval queue (whether the interval queue has elapsed or not). This packet is the next one to transmit from the flow. Thus, the output queue has at most $M$ leaves, and each interval queue has at most $M$ packets.

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

In our scheme, we require $\theta$ to be at least $(N \cdot L_t^{max})/C$. If a packet arrives at any time during the $n^{th}$ interval, $(n \cdot \theta,\ (n+1) \cdot \theta]$, it is added to the queue of the next interval. Thus, it becomes eligible at time $(n+2) \cdot \theta$, which implies $\psi \leq \theta$.

Note that once a packet $p_{f,i}$ becomes eligible and its interval queue is added to the output queue, all packets $p_{f,j}$, $j < i$, have been received. Only the packet with the smallest index is kept in an interval queue, and hence, packet $p_{f,i}$ is indeed the next packet to transmit from $f$.

However, there is an issue which is not present in [20]. It is possible that a packet $p_{f,j}$, $j < i$, is received while packet $p_{f,i}$ is contained in an interval queue. If this is the case, $p_{f,i}$ is removed from its interval queue, and added to the regular queue of packets of flow $f$ (discussed below). Then, packet $p_{f,j}$ is added to the appropriate interval queue.

Only the packet of $f$ with smallest index is stored in an interval queue. Other packets are stored in the regular queue of $f$. Since the scheduler must sort these packets by index, and reorder is limited, we assume the queue of $f$ is actually two queues. The first is a small queue of $N$ packets[1] sorted by index. An incoming packet is placed in the FIFO queue only if the small queue is full. When a packet is removed from the small queue, the next packet is removed from the FIFO queue and added to the small queue. Due to the limited reorder, and since the $N$ most recent packets are in the small queue, the packet with smallest index (other than the one in the interval queue) is always be found in the small queue.

In conclusion, processing an arrival or departure of a packet can be done in O($\log M$) time. Also, moving an interval queue to the output queue requires only O($\log M$) time. Thus, $\theta$ can be small, close to the transmission time of a packet.

[1]This assumes equal packet sizes. If not, the queue must be able to contain packets adding up to $N \cdot L_t^{max}$ bytes.

## 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 scheduling protocol designed for a single channel can be converted into a multiple-channel scheduling protocol without significantly increasing the delay at the scheduling node. This technique is inherently non-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 multiple-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 for the flow. We have shown that this increase in delay can be eliminated through the use of efficient sorting techniques.

There are several possible venues for future work in multiple-channel scheduling protocols. First, we have assumed all channels of a scheduler have the same bandwidth. This can be relaxed without significantly changing the theorems. In addition, the term

$$\frac{(N-1) \cdot L^{max}}{C}$$

in the theorems can actually be reduced to half. We did not present this reduction due to lack of space. We will present these improvements in the journal version of the paper. Finally, we will explore if there is any significant advantage in designing scheduling protocols specifically for multiple channels, as opposed to transforming a single-channel scheduler into a multiple-channel scheduler as done in this paper.

## References

[1] Baruah, S., Cohen, N, Plaxton, G, Varvel, D., "Proportionate Progress: A Notion of Fairness in Resource Allocation", *Algorithmica*, 15:600-625, 1996.

[2] Baruah, S., Gherke, J., Plaxton, G., "Fast Scheduling of Periodic Tasks on Multiple Resources", *Proc. Intl. Parallel Processing Symposium*, 1995.

[3] Bennet, J.C.R., Partridge, C., Shectman, N., "Packet Reordering is not Pathological Network Behavior", *IEEE/ACM Trans. on Networking*, Vol 7 No 6, Dec. 1999.

[4] Blanquer J.M., Ozden B., "Fair Queuing for Aggregated Multiple Links", *Proceedings of the ACM SIGCOMM Conference*, 2001.

[5] Cobb J., "An In-Depth Look at Flow Aggregation", *Proceedings of the IEEE International Conference on Network Protocols*, 1999.

[6] Cobb J., Gouda M., "Flow Theory", *IEEE/ACM Transactions on Networking*, October 1997.

[7] Cobb J., Lin M., "End-to-End Delay Guarantess for Multi-Channel Schedulers", Technical Report, Department of Computer Science, The University of Texas at Dallas, UTDCS-01-02.

[8] 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.

[9] Figueira N., Pasquale J., "A Schedulability Condition for Deadline-Ordered Service Disciplines", *IEEE/ACM Transactions on Networking*, Vol. 5 No. 2, April 1997.

[10] Figueira N., Pasquale J., "Leave-in-Time: A New Service Discipline for Real-Time Communications in a Packet-Switching Data Network", *Proceedings of the ACM SIGCOMM Conference*, 1995.

[11] Fumagalli, A., Cai, J., Chlamtac, I., "A Token Based Protocol for Integrated Packet and Circuit Switching in WDM", *Proceedings of the IEEE GLOBECOM Conference*, 1998.

[12] Golestani, S.J., "A Framing Strategy for Congestion Management", *IEEE Journal on Selected Areas in Communications*, Vol. 9 No. 7, Sept. 1991.
[13] Golestani, S. J., "A Self-Clocking Fair-Queuing Scheme for Broadband Applications", *Proc. of the IEEE INFOCOM 1994 Conference*.
[14] Goyal P, Lam S., Vin H., "Determining End-to-End Delay Bounds in Heterogeneous Networks", *Proceedings of the NOSSDAV workshop*, 1995.
[15] Keshav S., "A Control Theoretic Approach to Flow Control", *Proceedings of the 1991 ACM SIGCOMM Conference*.
[16] Liu C., Layland J., "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment", *Journal of the ACM*, Vol. 20, Jan 1973.
[17] Moir, M., Ramamurthy, S., "Fair Scheduling of Fixed and Migrating Periodic Tasks on Multiple Resources", *IEEE Real-Time Systems Symposium*, 1999.
[18] 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.
[19] Stiliadis, D. Varma, A., "Rate-proportional Servers: a Design Methodology for Fair Queuing Algorithms" *IEEE/ACM Transactions on Networking*, Vol. 6 No., 2 , April 1998.
[20] Stiliadis, D. Varma, A., "A General Methodology for Designing Efficient Traffic Scheduler Shaping Algorithms" *Proc. of the INFOCOM 1997 Conference*.
[21] Xie G., Lam S., "Delay Guarantee of Virtual Clock Server", *IEEE/ACM Transactions on Networking*, Dec. 1995.
[22] Verma D. C., Zhang H., Ferrari D., "Delay Jitter for Real-Time Communication in a Packet Switched Network", *Proc. of the TRICOM 1991 Conference*.
[23] Zhang H., "Service Disciplines for Guaranteed Performance Service in Packet-Switching Networks", *Proceedings of the IEEE*, Vol. 93, No. 10, Oct. 1995.
[24] Zheng Q., Shin K.G., "On the Ability of Establishing Real-Time Channels in Point-to-Point Packet-Switched Networks", *IEEE Transactions on Comm.*, Vol 42, No. 2/3/4, 1994.
[25] Zhang H., Ferrari D., "Rate-Controlled Static Priority Queuing", *Proceedings of the INFOCOM 1993 Conference*.

## APPENDIX

*Lemma 2:* Consider a scheduler $t$ with input flow $f$. If for any packet $p_{f,j}$, where $1 \le j < i$, 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 $f$, 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 $\hat{F}_{t,f,(j-1)} \le F_{t,f,j}$. We have four cases to consider.

1) $\hat{A}_{t,f,j} \le F_{t,f,(j-2)}$
In this case, in the reordered flow $p_{f,j}$ is the next packet after $p_{f,(j-2)}$. Hence, from the definition of $S$,

$$\hat{S}_{t,f,j} = F_{t,f,(j-2)}$$

$$\hat{F}_{t,f,j} = F_{t,f,(j-2)} + L_{f,j}/R_f \tag{3}$$

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

a) $A_{t,f,(j-1)} \le \hat{F}_{t,f,j}$
Because in the reordered flow $p_{f,j}$ is the packet previous to $p_{f,(j-1)}$, from the definition of $S$,

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

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

From equation (3),

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

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

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

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

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 $S$,

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

$$\hat{F}_{t,f,(j-1)} = A_{t,f,(j-1)} + L_{f,(j-1)}/R_f \tag{4}$$

In the ordered flow, from the definition of $S$,

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

Hence, combining the above with Equation (4), $\hat{F}_{t,f,(j-1)} < F_{t,f,j}$.

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 $S$,

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

$$\hat{F}_{t,f,j} = \hat{A}_{t,f,j} + L_{f,j}/R_f \tag{5}$$

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 $S$,

$$\begin{aligned}\hat{S}_{t,f,(j-1)} &= \hat{F}_{t,f,j} \\ &= \{\text{from Equation (5)}\} \\ &\quad \hat{A}_{t,f,j} + L_{f,j}/R_f\end{aligned}$$

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 array, from the definition of $S$, $F$

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

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 $L_q/R_f$. ■

The proof of Lemma 3 may be found in [7].

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

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

Hence, for all packets of $g$, the corresponding packet of $h$ arrives no later than the packet of $g$. Note that from the results of [5], [10], 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 $p_{f,k}$.

Only packets $p_{g,1}$ up to $p_{g,j}$ affect the computation of $S_{t,g,j}$. Hence, we obtain a new flow $h'$ which contains only the packets of $h$ corresponding to $p_{g,1}$ up to $p_{g,j}$. Note that $p_{f,k}$, which is $p_{g,j}$, is the last packet of $g$. Also, since we assumed that $p_{f,k}$ 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 $p_{f,k}$ itself) to the arrival times of the corresponding packets of $g$. This results in flow $h''$. From the lemma, the start time of $p_{f,k}$ in $h''$ does not increase.

Finally, the arrival time of $p_{f,k}$ 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,

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

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

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

■

The proofs of Corollaries 2 and 3 may be found in [7].

We next present the proof of Theorem 5.

*Proof:* Since the maximum delay a packet $p_{f,i}$ will experience before being considered eligible is $\psi$, and since $s$ is a start-time scheduler, that means that the packet will be eligible no later than time

$$S_{s,f,i} + \delta_{s,f,i} + \psi \tag{6}$$

From the lower bound on $\psi$, all packets with smaller index have been received when $p_{f,i}$ is received. Furthermore, from the assumptions of the theorem, all of these packets are eligible. Hence, $p_{f,i}$ is ready to be considered for scheduling at the time in (6).

Hence, $s$ plus the extra delay introduced before the packet becomes eligible can be considered as a single start-time scheduler $\hat{s}$ where $\delta_{\hat{s},f,i} = \delta_{s,f,i} + \psi$. Hence, the start-time at $t$ follows from Theorem 1. ■