# Workconserving vs. Non-workconserving Packet Scheduling: An Issue Revisited [*]

*Jörg Liebeherr* *Erhan Yilmaz*

Department of Computer Science
University of Virginia
Charlottesville, VA 22903
email: {`jorg,erhan`}`@cs.virginia.edu`

`Computer Science Report No. CS-99-05`

## Abstract

Many packet schedulers for QoS networks are equipped with a rate-control mechanism. The function of a rate-control mechanism (also called rate controller) is to buffer packets from flows which exceed their negotiated traffic profile. It has been established that rate controllers lead to reduced buffer requirements at packet switches, and do not increase the worst-case delays in a deterministic service. On the other hand, rate controllers make a scheduler non-workconserving, and, thus, may yield higher average end-to-end delays. In this study, we show that by properly modifying a rate-controller, one can design a scheduler which balances buffer requirements against average delays. We present a scheduler, called Earliness-based Earliest Deadline First (EEDF), which achieves such a balancing using a tunable rate control mechanism. In simulation experiments, we compare EEDF with a rate-controlled EDF scheduler and a workconserving version of EDF.

[*]This work is supported in part by the National Science Foundation under Grants NCR-9624106 (CAREER) and ANIR-9730103.

## 1 Introduction

The selection of the packet scheduling discipline which operates at output ports of switches is one of the key design criteria for a Quality-of-Service (QoS) network. In the last decade, numerous packet scheduling disciplines have been proposed for QoS networks [1, 15, 20, 23].

Packet scheduling disciplines can be classified as workconserving or non-workconserving. A workconserving scheduler always transmits a packet when there is traffic waiting. A non-workconserving packet scheduling discipline may leave a link idle, even if there is a backlog of traffic.

Non-workconserving scheduling disciplines have been proposed for use in a QoS network for different reasons: Slotting or framing, delay jitter control, and rate control. Examples of schedulers which use slotting and framing include Stop-and-Go Queueing [14] and Hierarchical Round Robin [3]. In a similar way as time-division multiplexers, these scheduling disciplines divide time into periods of fixed length, so-called *slots* or *frames*, and assign periods to flows. If a packet misses an assigned slot, the packet must wait until the beginning of the next available slot. Thus, if a packet arrives to an empty scheduler, the scheduler will stay idle until the next available slot. A second group of non-workconserving schedulers is used to implement a service with bounded *delay jitter*, where delay jitter of a flow refers to the range of end-to-end delays of packets from this flow.[1] Examples include Earliest-Deadline-Due–Delay Jitter (EDD-DJ) [17] and Rate-Control Static-Priority–Delay Jitter (RCSP-DJ) [21]. To enforce lower bounds on delays, schedulers with delay jitter control have a holding mechanism which buffers packets until the bound on the minimum delay is satisfied.

Neither framing nor delay jitter control are widely used in current QoS networks. The primary reason for employing non-workconserving packet scheduling disciplines in todays networks is to perform *rate control* or *shaping* of packets entering a scheduler. The algorithms for admission control and provisioning of buffer and link capacity are based on a characterization of a flow's traffic, called *traffic profile* or *TSpec* [4]. However, since the traffic pattern of a flow can get distorted inside the network, the traffic of a flow may no longer adhere to its original profile. A *rate controller*[2] is a mechanism which enforces that traffic from a flow which is forwarded to a scheduler conforms to its original profile. This is achieved by buffering all packets exceeding the profile. Clearly, a rate controller makes a scheduler non-workconserving. Examples of scheduling disciplines which employ rate controllers are Rate-Controlled Static Priority (RCSP) [22, 18] and Rate-Controlled Earliest Deadline First (RC-EDF) [12]. For a service which guarantees bounds on the worst-case delay of packets, so-called *deterministic service*, a scheduler with a rate controller can satisfy the same set of delay bounds as a workconserving packet scheduler [19].

There are many arguments for and against non-workconserving schedulers (see [15], pp. 227-228). A disadvantage of non-workconserving packet scheduling disciplines over workconserving algorithms is that they may increase average end-to-end delays. However, one can argue that

[1]More precisely, in this context, the maximum delay jitter is the difference between the maximum and minimum end-to-end packet delay.

[2]Rate controllers are also referred to as *regulators* and *shapers*. We will use the terms interchangeably.

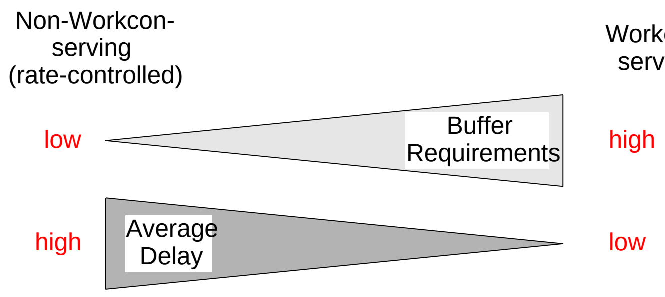


Figure 1: Illustration of trade-off.

this is a non-issue for applications which are only interested in worst-case delay guarantees. Also, the argument that rate controllers waste bandwidth can be disputed since a scheduler can always transmit best-effort traffic. An argument against jitter control is that it can be made obsolete through buffering at the receiver. An argument against both jitter and rate controllers is that they require maintaining per-flow state information at switches. An advantage of rate controllers is that they reduce the total amount of buffer space needed for a flow at switches [7, 8]. Overall, the jury is still out on the advantages of using workconserving over non-workconserving disciplines, or vice versa. This study will contribute to the debate by arguing that it may not always be meaningful to draw a clear line between workconserving and (rate-controlled) non-workconserving scheduling disciplines.

In this paper we point out that there is a design space between rate-controlled (non-workconserving) and workconserving schedulers which can be exploited by a switch designer. We present a modified Earliest-Deadline-First scheduler, called *Earliness-based EDF (EEDF)*, which enables a switch to trade off lower buffer requirements for higher average end-to-end delays (see Figure 1). To our knowledge, this study is the first which argues that, from a performance perspective, workconserving and and non-workconserving (rate-controlled) schedulers are two extreme points on a spectrum of feasible scheduler designs.

The remainder of this paper is structured as follows. In Section 2 we present an example which illustrates the relationship between average delays and buffer requirements in workconserving and rate-controlled schedulers. In Section 3 we quantify an expression for the so-called *earliness*, defined to be the maximum waiting time in a rate controller. In Section 4 we present the EEDF scheduler. In Section 5 we discuss simulation experiments which compare the EEDF scheduler with workconserving and rate-controlled versions of EDF. We present conclusions in Section 6.

## 2 Effects of Per-Node Traffic Shaping

In Figure 2 we show a scheduler without and with a rate controller. A workconserving scheduler never stays idle if there is a packet waiting to be sent. If a scheduler has a rate controller, the rate controller monitors arrivals to the scheduler and enforces for each flow that traffic arrivals conform to the traffic profile of the flow. A rate controller makes a scheduler non-workconserving since a

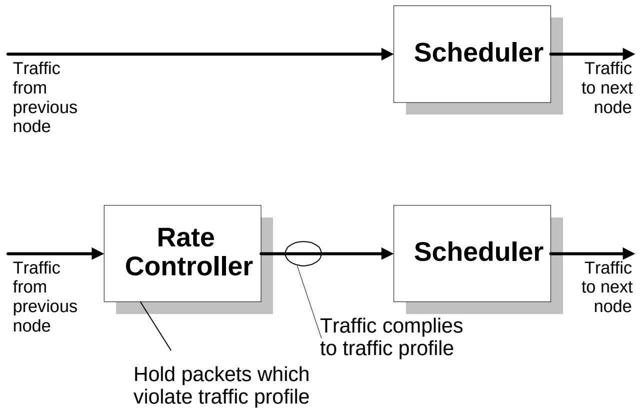

Scheduler
Traffic from previous node
Traffic to next node
Rate Controller
Scheduler
Traffic from previous node
Traffic to next node
Traffic complies to traffic profile
Hold packets which violate traffic profile

Figure 2: Scheduler without and with rate controller.

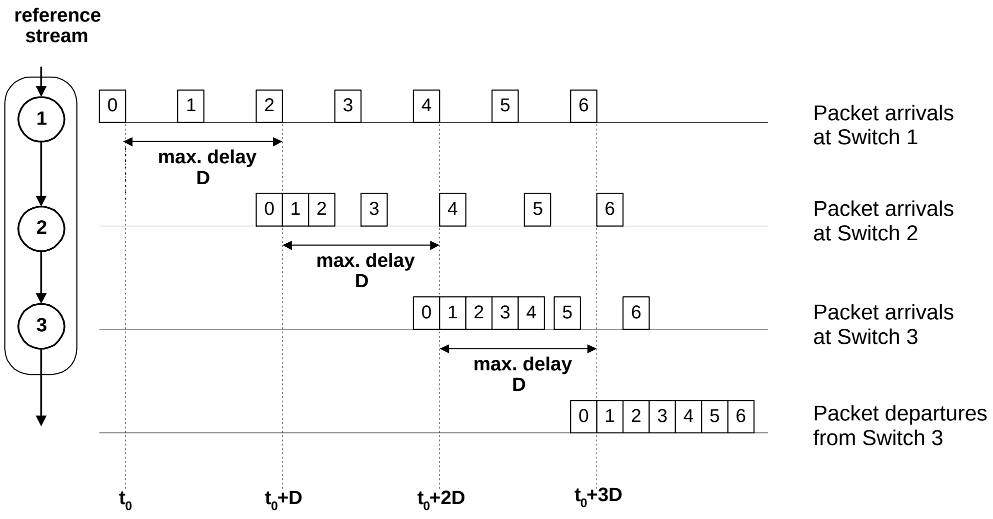


Figure 3: Packet arrivals at work-conserving scheduler.

packet may be kept in the rate controller even though the link is left idle.

Figure 3 depicts an example of an arrival scenario at a workconserving scheduler. The example demonstrates how traffic is distorted, resulting in increased burstiness and increased buffer requirements. In the figure, a reference traffic stream sends packets which are carried through a sequence of switches, depicted as circles. We assume that, at each switch, the maximum delay of a packet satisfies a delay bound $D$. (By convention, the delay bound includes the transmission delay of a packet. We assume that propagation delays are negligible) Figure 3 shows a feasible scenario of seven packet arrivals. The horizontal lines are time lines and packet arrivals are depicted as boxes. A packet is not considered arrived unless the last bit of the packet has arrived.

In the example, seven packets arrive, equally spaced, to Switch 1. The first packet from the reference flow experiences the maximum delay D. We assume that, at all switches, packets from the reference flow which are queued behind the first packet are transmitted back-to-back following the first packet. Subsequent packet arrivals do not receive any queueing delay. Note that such a scenario can occur, if, at each switch, packet 0 is delayed by higher-priority traffic, and there is no additional higher-priority traffic once packet 0 has been transmitted.

In the figure, we see that the first packet, packet 0, departs from Switch 1 and arrives to Switch 2 at time $t_0 + D$. At time $t_0 + D$, packets 1 and 2 have accumulated behind packet 0. Per our assumptions, packets 1 and 2 are transmitted immediately following packet 0, separated only by the transmission delay. Thus, we see a burst of three packets arriving at Switch 2.

At Switch 2, we assume again that packet 0 experiences the maximum delay D. When packet 0 is transmitted, at time $t_0 + 2D$, five packets from the reference flow have accumulated in the queue, and arrive as a burst to the next switch. When the packet depart from Switch 3, starting at time $t_0 + 3D$, the burst has grown to include all seven packets. An important observation to be made is that the burst size increases at each switch.

In Figure 4 we show the same arrival scenario for a non-workconserving scheduler with a rate controller. At Switches 2 and 3, the traffic from the reference flow is reshaped to the initial periodic arrival pattern. Figure 4 shows the arrivals at the rate controller as well as at the scheduler. At time $t_0 + D$, when packets 0, 1, and 2 arrive as a burst at Switch 2, packets 1 and 2 are buffered in the rate controller. The rate controller reconstitutes the periodic arrival pattern. Thus, the spacing of packets arrivals to the scheduler of Switch 2 is identical to the spacing at Switch 1. An effect of the rate controller is an increase of delay for all packets but packet 0.

The example illustrates the trade-off of workconserving and non-workconserving schedulers. With workconserving schedulers, the burstiness of the reference stream, and, hence, the buffer requirements, increase with the number of switches traversed. In contrast, rate controllers bound the burst size of a flow and, therefore, yield lower buffer requirements. However, a consequence of buffering is that the average delay of packets is increased. In the depicted example, assuming that the transmission time of a packet is 1 and the delay bound is D=6 packet transmission times, the average delay of packets is 21 in Figure 3 and 23.3 in Figure 4. In [19] it was shown that, in a deterministic service which gives bounds on delays, rate controllers do not require larger worst-case

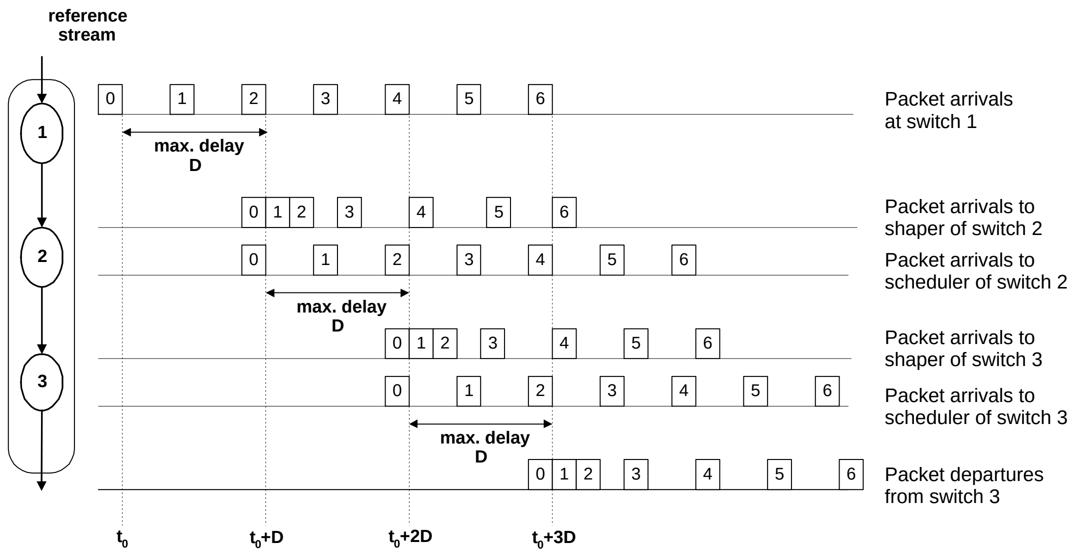

reference stream
1
2
3
0
1
2
3
4
5
6
max. delay D
Packet arrivals at switch 1
Packet arrivals to shaper of switch 2
Packet arrivals to scheduler of switch 2
Packet arrivals to shaper of switch 3
Packet arrivals to scheduler of switch 3
Packet departures from switch 3
t0
t0+D
t0+2D
t0+3D

Figure 4: Packet arrivals at rate-controlled scheduler.

delays. (In our example, the worst-case delay of D at each switch is experienced by packet 0.)

To overcome the drawback that a packet is held in the rate controller even when the scheduler is idle, different approaches have been taken. For example, in [21] and [9] the scheduler transmits best-effort traffic, if there is no packet with service guarantees in the scheduling queue. In [13], the scheduler simply picks some packet held in the rate controller for transmission if the scheduler queue is empty. Note, however, that the latter technique may, in the worst-case, suffer from a burstiness increase similar to a workconserving scheduler.

## 3 Quantifying the Effects of Traffic Shaping

In this section, we present an expression for the amount of time that an arriving packet may wait in the rate controller of a non-workconserving scheduler, referred to as *earliness.* The theoretical basis for the presentation in this section is Cruz's network calculus [7, 8, 9] and [13]. We assume that the underlying network service is a deterministic service with guarantees on worst-case delays.

Let $A_i^{in}[t,t+\tau]$ and $A_i^{out}[t,t+\tau]$, respectively, denote the arrivals and departures in time interval $[t,t+\tau]$ from flow $i$ at a switch on its path. Arrivals to the first switch on the route are assumed to be bounded by a right-continuous, non-decreasing, sub-additive function $A_i^*$ as follows:

$$A_i^{in}[t,t+\tau] \le A_i^*(\tau) \qquad\qquad \forall t, \forall \tau, \forall i \tag{1}$$

We refer to $A_i^*$ as traffic constraint function and say that $A_i^{in}$ *conforms* to $A_i^*$, written as $A_i^{in} \preceq A_i^*$. Subadditivity implies that $A_i^*(t_1+t_2) \le A_i^*(t_1) + A_i^*(t_2)$, for all $t_1, t_2$. $A_i^*$ characterizes the traffic profile of flow $i$.

As an example, in the $(\sigma,\rho)$-model [8], where traffic on a flow $i$ is described by a burst parameter $\sigma_i$ and a rate parameter $\rho_i$, the traffic constraint function is given by $A_i^*(t) = \sigma_i + \rho_i t$.

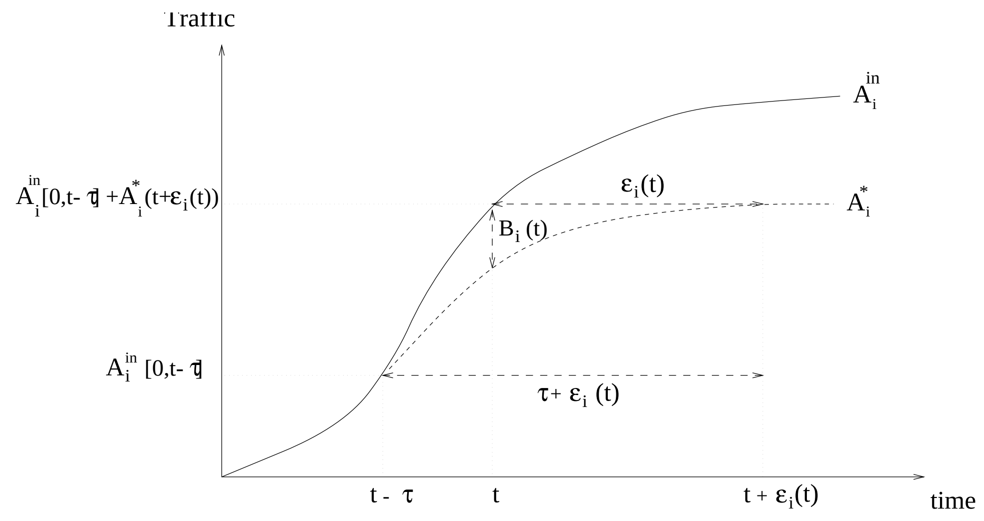


Figure 5: Relationship between input $A_i^{in}$, traffic constraint function $A_i^*$, backlog $B_i$, and earliness $\varepsilon_i$.

There are two mechanisms which can enforce that traffic on a flow $i$ entering a scheduler conforms to a given constraint function $A_i^*$:

1. A *traffic conditioner* (or *policer*) at the entrance of the network may reject traffic if it does not conform to $A_i^*$. Alternatively, the conditioner could mark packets as belonging to a lower-priority service class, e.g., best effort.

2. A *rate controller* (or *traffic shaper*) enforces that the traffic from flow $i$ which enters a scheduler conforms to $A_i^*$. Packets which exceed $A_i^*$ are buffered by the rate controller.

According to Cruz [8], if a switch satisfies a delay bound, the following relationship can be established between the input and output traffic on a switch.

**Theorem 1 (Cruz [8])** *Given $A_i^{in} \preceq A_i^*$ at a switch. If the switch guarantees a delay bound of $D_i$ for traffic from flow $i$, then*

$$A_i^{out}(t, t+\tau) \leq A_i^*(\tau + D_i) \qquad , \textit{ for all } t, \tau \geq 0 \tag{2}$$

We refer to the waiting time of a packet in the rate controller as *earliness*. This terminology alludes to the fact that any packet held in the rate controller arrived 'early' with respect to its traffic constraint function. We use $\varepsilon_i(t)$ to denote the earliness of an arrival from flow $i$ at time $t$. In Figure 5 we illustrate the relationship between traffic arrivals, traffic constraint function, backlog, and earliness. The earliness is formally defined as:[3]

[3]Using the convolution operator from Cruz's network calculus [9] and changing notation so that $A_i^{in}(t) \equiv A_i^{in}[0,t]$ Eqn. (3) can be rewritten as: $\varepsilon_i(t) = \inf\{x \mid A_i^{in}(t) \leq A_i^{in} * A_i^*(t+x)\}$.

$$\varepsilon_i(t) = \inf\{x \mid A_i^{in}[0,t] \le \inf_{0\le\tau\le t}\{A_i^{in}[0,t-\tau] + A_i^*(\tau+x)\} \quad (3)$$

If the inverse of $A_i^*$ exists, the earliness can be directly computed as given in the next theorem.

**Theorem 2 (Georgiadis et. al. [13])** *Given a flow with arrivals $A_i^{in}$ to a rate-controlled packet scheduler, where the rate controller enforces $A_i^*$. Let $t = A_i^{*,-1}(y)$ be the inverse function of $y = A_i^*(t)$. Then*

$$\varepsilon_i(t) = \sup_{0\le\tau\le t}\{A_i^{*,-1}(A_i^{in}(t-\tau,t]) - \tau\} \quad (4)$$

An important insight from Theorem 2 is that the complexity of implementing a rate controller is directly related to the complexity of computing the inverse of the traffic constraint function.

In the Appendix we show a proof, which is simpler than the one shown in [13]. Also, while the proof in [13] is done for concave functions, our proof is formulated for the more general class of subadditive constraint functions.

## 4 Earliness-based EDF (EEDF)

We will present a version of an Earliest-Deadline-First (EDF) scheduler which exploits the concept of earliness to balance the pros and cons of workconserving and non-workconserving scheduling. The scheduler is referred to as *Earliness-based EDF(EEDF)*.

In EDF, each packet is timestamped with a *deadline* set equal to the sum of its arrival time and a delay bound, and packets are transmitted in increasing order of deadlines [5, 10, 11, 12, 13, 16, 24]. For a service where all packets satisfy worst-case end-to-end delay bounds [6, 10], EDF has been shown to have *optimal efficiency*, in that it, among all scheduling disciplines, supports the most sessions with delay guarantees [12, 16]. The admission control conditions for EDF, which verifies that all packet from every flow satisfy their delay bounds are as follows:

**Theorem 3 (Liebeherr, Wrege, Ferrari [16])** *A set of flows $\mathcal{C}$ with delay bound $D_i$ for flow $i \in \mathcal{C}$ satisfies all delay bounds in an EDF scheduler if and only if for all $t \ge D_1$:*

$$R \cdot t \ge \sum_{i\in\mathcal{C}} A_i^*(t - D_i) + \max_{D_j > t} L_j \quad (5)$$

*where $R$ is the transmission rate of the scheduler, and $L_j$ is the maximum transmission time of a packet from flow $j$.*

More efficient versions of the admission control conditions have been derived in [5, 11].

In the past, both work-conserving and non-workconserving versions of EDF have been employed. The Tenet protocol suite [2] uses a workconserving version of EDF, referred to as *Delay-EDD*. Here, the deadline $dl_i(t)$ for a packet arrival at time $t$ of a flow $i$ with delay bound $D_i$ is calculated as: [4]

$$dl_i(t) = t + \varepsilon_i(t) + D_i$$

[4]The actual definition of Delay-EDD is somewhat different, as it assumes only peak-rate enforcement of arriving traffic.

Thus, upon arrival, Delay-EDD extends the deadline of a packet by its earliness. The deadline of an early packet ($\varepsilon_i(t) > 0$) does not depend on its earliness.

In [12], a non-workconserving version of EDF with a rate controller is proposed. The resulting scheduler is referred to as *Rate-controlled EDF (RC-EDF)*. The rate controller holds all packets with positive earliness. For a packet with arrival time $t$, the scheduler operates as follows:

```
If $\varepsilon_i(t) > 0$
    Hold packet in rate controller for $\varepsilon_i(t)$ time units
At time  $t + \varepsilon_i(t)$: $dl_i(t) = t + D_i$
```

Our modified version of EDF is a compromise between the workconserving Delay-EDD and the non-workconserving RC-EDF disciplines, which attempts to consolidate the advantages and drawbacks of the workconserving and non-workconserving versions. The scheduler, called *Earliness-based EDF (EEDF)* is an RC-EDF scheduler with a (tunable) threshold parameter $\varepsilon_i^*$. If the earliness of a packet is less than $\varepsilon_i^*$, the scheduler operates identically to Delay-EDD. If the earliness exceeds $\varepsilon_i^*$, the packet is kept in a rate controller until the earliness is $\leq \varepsilon_i^*$. The operations performed by an EEDF scheduler upon arrival of a packet at time $t$ are as follows:

```
If $\varepsilon_i(t) > \varepsilon_i^*$
    Hold packet in rate controller for $\varepsilon_i(t) - \varepsilon_i^*$ time units
    At time  $t + \varepsilon_i(t) - \varepsilon_i^*$: $dl_i(t) = t + D_i + \varepsilon_i^*$
Else
    $dl_i(t) = t + D_i + \varepsilon_i(t)$
```

By decreasing $\varepsilon_i^*$, the EEDF scheduler becomes more non-workconserving. EEDF with $\varepsilon_i^* = 0$ is identical to RC-EDF. The result of increasing $\varepsilon_i^*$ is that less packets are held in the rate controller. EEDF with $\varepsilon_i^* = \infty$ is identical to Delay-EDD.

Note that, for the same sequence of packet arrivals from a flow, the deadline $dl_i(t)$ assigned to packets are identical in all versions of the scheduler. As a consequence, the admission control conditions in Theorem 3 hold for Delay-EDD, RC-EDF, and EEDF.

## 5 Simulation Experiments

In this section, we compare the performance of EEDF with a workconserving Delay-EDD and a non-workconserving RC-EDF scheduler. According to our goals stated in Section 1, and as illustrated in Figure 1, we want to gain insight into the trade-off of average delay and buffer requirements in these schedulers. We want to see how well EEDF can exploit the design space between workconserving and non-workconserving schedulers by varying parameter $\varepsilon_i^*$. Increasing $\varepsilon_i^*$ will decrease the average delay of packets, but also increase the burstiness, and, with it, the buffer requirements, of flows.

We have implemented a packet-level simulation to quantify the trade-off using a simulation package developed at CMU. The simulated network consists of 10 switches in a tandem configuration,

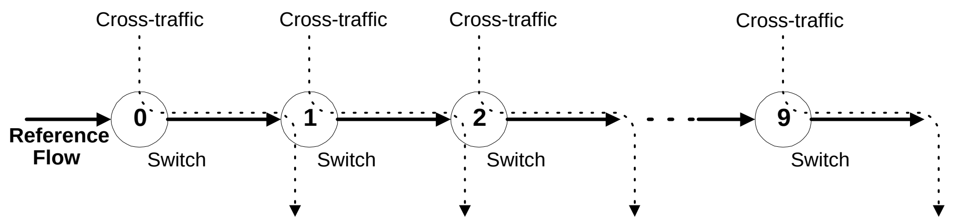


Figure 6: Simulated network configuration.

as shown in Figure 6. In the simulations, we study the end-to-end delays and buffer requirements for a reference packet flow which traverses all switches in the simulated configuration. Switch 0 is the first and Switch 9 is the last switch on the path of the reference flow. At each switch, there is cross-traffic which interferes with the traffic from the reference flow. As indicated in Figure 6, cross-traffic generated at a particular switch interferes with the reference flow only at the scheduler at this switch. The characteristics of cross-traffic are identical at each switch, and cross-traffic at different switches is not correlated.

All packets are assumed to have a constant size of 53 bytes. The links between switches have a data rate of 51.85 Mbps. The packet scheduling method at the switches are Delay-EDD, RC-EDF, and EEDF. For all experiments, we have ensured that the schedulability conditions from Theorem 3 are satisfied. Thus, at each switch, all packets satisfy their delay bound.

The traffic profile of the reference flow complies to the $(\sigma, \rho)$-model with

$$\sigma_r = 100 \text{ Kbps} \qquad \rho_r = 10 \text{ Mbps}$$

The parameters are chosen to resemble that of a high-volume CBR video source, such as Motion-JPEG. The delay bounds are set to 65 msec per switch, resulting in an end-to-end delay bound of 650 msec. We assume that the reference flow is greedy, that is, it will initially send a burst of size $\sigma_r$ and then transmit at rate $\rho_r$.

The cross-traffic has a $(\sigma, \rho)$ traffic profile and parameters:

$$\sigma_c = 3 \text{ Mbps}, \qquad \rho_c = 30 \text{ Mbps}.$$

Cross-traffic arrives in an on-off pattern. Bursts have size $\sigma_c$ and bursts have exponentially distributed interarrival times. The delay bound of cross-traffic is set to 60 msec at each switch. Note that the average load from both reference flow and cross-traffic is about 80% of the link capacity.

The length of simulation experiments is set to 50 sec. We run the simulation three times. In each run we use a different random seed to generate the cross-traffic. We only take measurements of the reference flow. We measure the maximum and average end-to-end packet delays over the three runs. The maximum delay is the maximum end-to-end delay of the packet among all the packets generated. The average delay is measured by taking the average of the end-to-end delays of all the packets generated. Furthermore, we measure, for each switch, the maximum backlog of traffic from the reference flow. The backlog is the aggregated backlog in the rate controller and the scheduler. Recall that the maximum backlog directly translates into buffer requirements.

## End-to-End Delays

In Table 1 we summarize the results for the end-to-end delays. The average end-to-end delays of the non-workconserving RC-EDF and the workconserving Delay-EDD differ by almost an order of magnitude. Even though all schedulers guarantee an end-to-end delay bound of 650 msec, we see that the actually measured maximum delay is smaller. The values for EEDF are between those of Delay-EDD and RC-EDF, dependent on the selection of $\varepsilon^*$.

| | | *Value of $\varepsilon^*$ (msec) in EEDF* | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | RC-EDF | 30 | 60 | 90 | 120 | 150 | 180 | 210 | 240 | 270 | Delay-EDD |
| *Average Delay (msec)* | 407.3 | 312.9 | 248.3 | 206.3 | 173.5 | 141.1 | 113.8 | 89.5 | 68.6 | 56.1 | 47.0 |
| *Max. Delay (msec)* | 533.3 | 452.3 | 402.3 | 355.6 | 341.4 | 336.6 | 336.6 | 336.6 | 336.6 | 336.6 | 336.6 |

Table 1: Average and maximum observed end-to-end delays of reference flow.

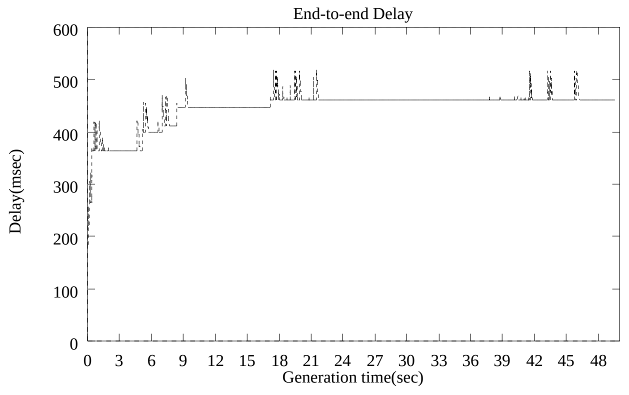
In Figure 7, we plot the end-to-end delays packets from the reference flow over the duration of one of the runs of the experiment. The non-workconserving RC-EDF has the highest end-to-end delays. For EEDF, as we increase $\varepsilon^*$, we see that the end-to-end delays of packets decrease. An observation to be made is that the delay jitter, grows fast when we increase $\varepsilon^*$.

## Buffer Requirements

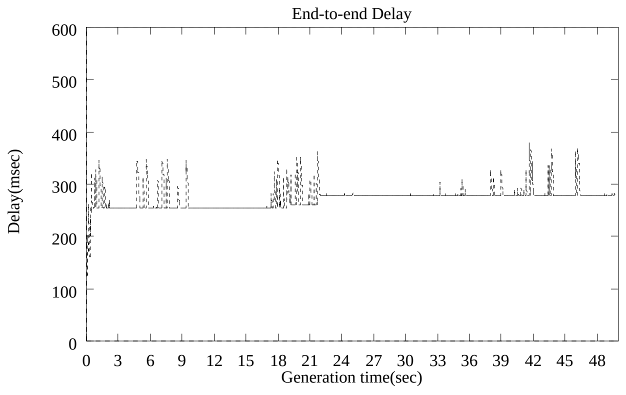
In Table 2 we show the maximum observed backlog over the three runs of the experiment. The data demonstrates that, in a work-conserving Delay-EDD scheduler, the backlog increases significantly with the number of hops traversed. The buffer requirements at the last switch are three times larger than those of the first switch.

| | | *Value of earliness ε(msec) in EEDF* | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Switch | RC-EDF | 30 | 60 | 90 | 120 | 150 | 180 | 210 | 240 | 270 | Delay-EDD |
| *0* | 0.750 | 0.750 | 0.750 | 0.750 | 0.750 | 0.750 | 0.750 | 0.750 | 0.750 | 0.750 | 0.750 |
| *1* | 1.097 | 1.072 | 1.072 | 1.072 | 1.072 | 1.072 | 1.072 | 1.072 | 1.072 | 1.072 | 1.072 |
| *2* | 1.171 | 1.247 | 1.471 | 1.471 | 1.471 | 1.471 | 1.471 | 1.471 | 1.471 | 1.471 | 1.471 |
| *3* | 1.147 | 1.246 | 1.570 | 1.745 | 1.745 | 1.745 | 1.745 | 1.745 | 1.745 | 1.745 | 1.745 |
| *4* | 1.172 | 1.321 | 1.496 | 1.696 | 1.920 | 1.945 | 1.995 | 1.995 | 1.995 | 1.995 | 1.995 |
| *5* | 1.197 | 1.297 | 1.421 | 1.671 | 1.770 | 1.845 | 1.845 | 1.845 | 1.845 | 1.845 | 1.845 |
| *6* | 1.197 | 1.347 | 1.446 | 1.646 | 1.770 | 2.069 | 2.194 | 2.194 | 2.194 | 2.194 | 2.194 |
| *7* | 1.172 | 1.396 | 1.671 | 1.945 | 2.146 | 2.444 | 2.618 | 2.618 | 2.618 | 2.618 | 2.618 |
| *8* | 1.172 | 1.396 | 1.770 | 2.020 | 2.144 | 2.394 | 2.568 | 2.743 | 2.893 | 2.869 | 2.869 |
| *9* | 1.172 | 1.371 | 1.621 | 1.895 | 2.194 | 2.319 | 2.319 | 2.319 | 2.319 | 2.593 | 2.593 |

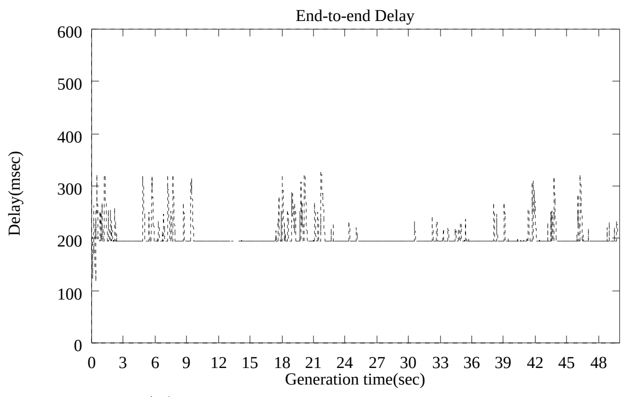
Table 2: Observed maximum backlog of traffic from reference flow at each switch (All values are in $Mb$).



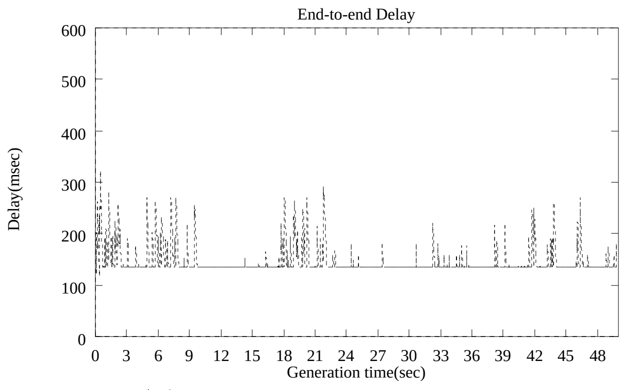
(a) RC-EDF



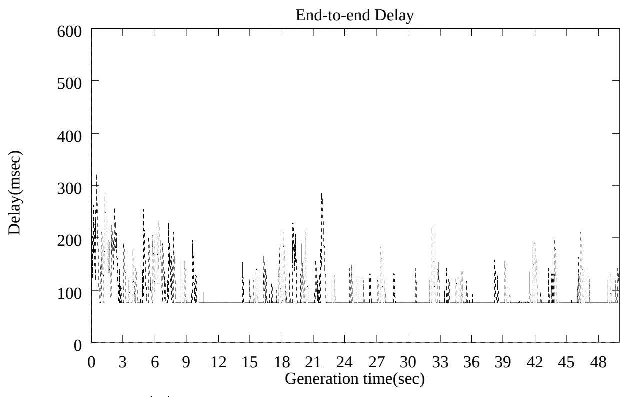
(b) EEDF with $\varepsilon = 60$ ms



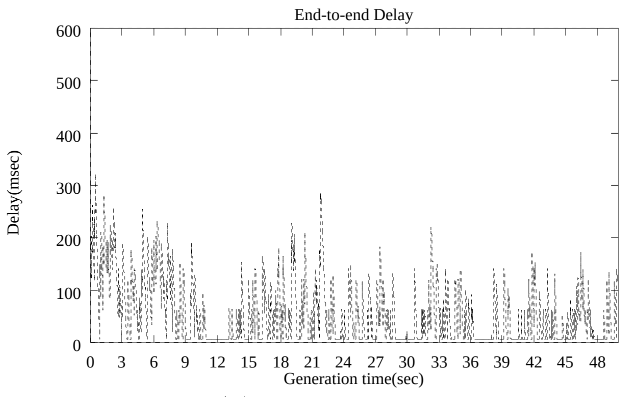
(c) EEDF with $\varepsilon = 120$ ms



(d) EEDF with $\varepsilon = 180$ ms



(e) EEDF with $\varepsilon = 240$ ms



(g) Delay-EDD

Figure 7: End-to-End Delays of packets from reference flow.

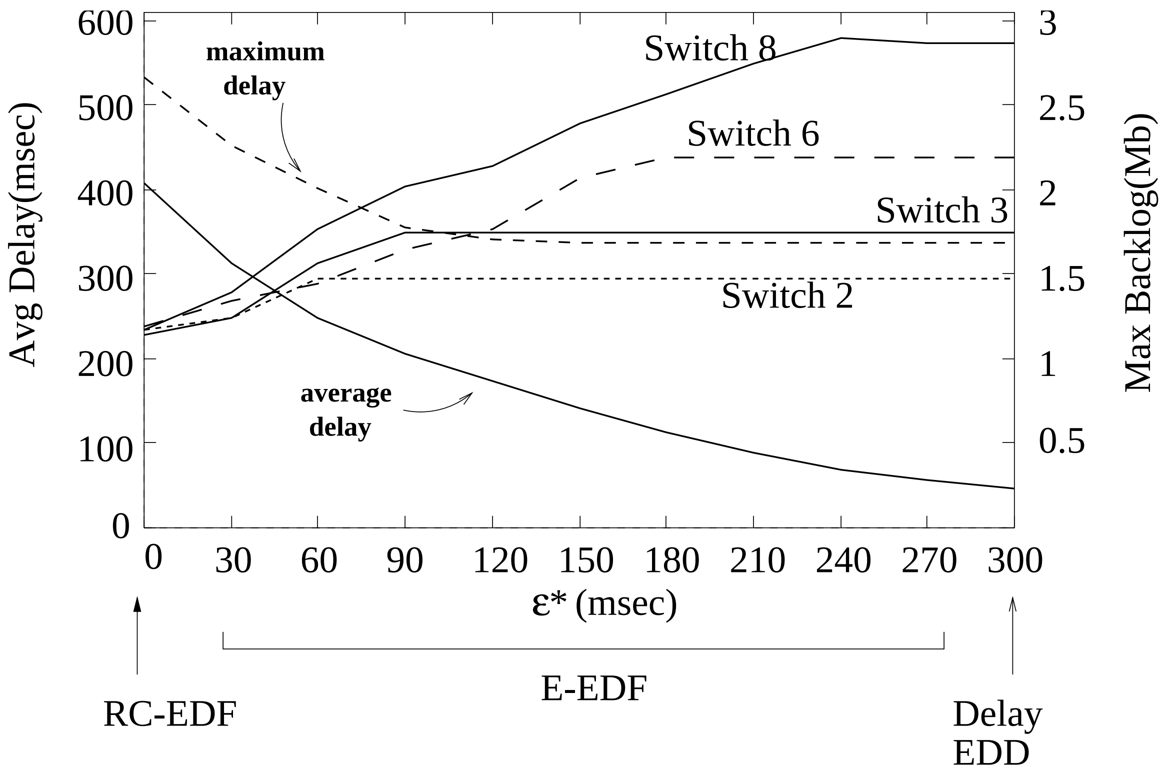


Figure 8: Summary of trade-off of delays and buffer requirements.

## Summary of the Tradeoff

Figure 8 summarizes the simulation results from Tables 1–3. The figure illustrates how the end-to-end delays and the maximum buffer requirements change as the earliness parameter $\varepsilon^*$ is varied. For $\varepsilon^* = 0$, we have a RC-EDF scheduler. For $\varepsilon^* = 300$, no packet from the reference flow is delayed at any rate controller on its path. Hence, the scheduler is identical to a workconserving Delay-EDD scheduler. For clarity of the presentation, the maximum backlogs are shown only for a subset of switches, namely, Switches 2, 3, 6 and 8.

The first observation is that varying $\varepsilon^*$ gives noticeably different results. The delays of RC-EDF ($\varepsilon^* = 0$) are much higher those of Delay-EDD ($\varepsilon^* = 300$) . The difference is more pronounced for the average delays than for the maximum delays. Figure 8 also shows how the buffer requirements increase as $\varepsilon^*$ is increased. Note that buffer requirements for schedulers with $\varepsilon^* > 0$ increase with the number of hops traversed.

Figure 8 supports our claim that EEDF is a compromise between the two extremes Delay-EDD and RC-EDF. The figure shows, that for a selection of $\varepsilon = 120$ ms, EEDF has half the average delay of RC-EDF, and, at the same time, significantly lower buffer requirements than Delay-EDD.

### Counterexample

We want to emphasize that the difference between RC-EDF and Delay-EDD, need not be as pronounced as in the previous example. (On the other hand, the simulation results do not depict a worst-case scenario feasible with the given parameters.) If switches are only lightly loaded, or if switches never built up a significant backlog, the difference between workconserving and non-workconserving disciplines may be negligible. In such cases, RC-EDF, Delay-EDD, and EEDF all give similar results.

We have run a simulation with the same parameters as in the previous example, with the exception that we changed the cross-traffic from a bursty mode to a Poisson source with average rate $\rho_c$. With this change, as shown in Table 3, the average and maximum end-to-end delays, respectively, for work-conserving and non-workconserving disciplines are very similar (and quite small). Table 4 shows that, the maximum backlog of the reference flow is small, too, for all disciplines. Obviously, in such a situation, EEDF offers no benefits.

| | RC-EDF | Delay-EDD |
|---|---|---|
| *Average Delay (msec)* | 5.69 | 3.99 |
| *Maximum Delay (msec)* | 14.85 | 14.85 |

Table 3: Average and maximum end-to-end delay.

| | *Switch* | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Scheduling Discipline | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| *RC-EDF* | 194.6 | 69.7 | 75.0 | 75.5 | 75.5 | 84.7 | 104.5 | 99.7 | 84.7 | 79.9 |
| *Delay-EDD* | 194.6 | 69.7 | 60.0 | 69.7 | 70.2 | 84.7 | 99.7 | 89.5 | 99.7 | 90.0 |

Table 4: Backlog at each switch (All values are in $Kb$).

## 6 Conclusions

We have argued that there is a rich design space between workconserving and non-workconserving (rate-controlled) packet schedulers. A disadvantage of non-workconserving packet schedulers over workconserving schedulers is that they lead to higher average end-to-end delays. On the other hand, workconserving schedulers result in higher buffer requirements. We demonstrated that the trade-off presented by the two classes of schedulers can be exploited using a tunable rate control mechanism. We presented a version of Earliest-Deadline-First (EDF), called Earliness-based EDF (EEDF), and we demonstrated that EEDF schedulers can balance buffer requirements and average end-to-end delays.

## Acknowledgments

We would like to thank Hui Zhang from CMU for making available his packet simulator.

## References

[1] C.M. Aras, J. F. Kurose, D. S. Reeves, and H. Schulzrinne. Real-Time Communication in Packet-Switched Networks. *Proceedings of the IEEE*, 82(1):122–139, January 1994.

[2] A. Banerjea, D. Ferrari, B. A. Mah, M. Moran, D. C. Verma, and H. Zhang. The Tenet Realttime Protocol Suite: Design, Implementation, and Experiences. *IEEE/ACM Transactions on Networking*, 4(1):1–10, February 1996.

[3] A. Banerjea and S. Keshav. Queueing delays in rate controlled networks. In *Proceedings of IEEE INFOCOM'93*, pages 547–556, San Francisco, CA, April 1993.

[4] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource ReSerVation Protocol (RSVP). IETF Internet-Draft, ftp://ds.internic.net/internet-drafts/, November 1997.

[5] F. Chiussi and V. Sivaraman. Achieving High Utilization in Guaranteed Services Networks using Early-Deadline-First Scheduling. In *Proc. 6th IEEE/IFIP International Workshop on Quality of Service (IWQoS '98)*, May 1998.

[6] D. D. Clark, S. Shenker, and L. Zhang. Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanisms. In *Proc. ACM Sigcomm*, pages 14–26, August 1992.

[7] R. Cruz. Quality of service guarantees in virtual circuit switched networks. *IEEE Journal on Selected Areas in Communications*, 13(6):1048–1056, August 1995.

[8] R. L. Cruz. A Calculus for Network Delay, Part I: Network Elements in Isolation. *IEEE Transactions on Information Theory*, 37(1):114–131, January 1991.

[9] R. L. Cruz. SCED+: Efficient Management of Quality of Service Guarantees. In *Proc. IEEE Infocom '98 Conference*, March/April 1998.

[10] D. Ferrari and D. C. Verma. A Scheme for Real-Time Channel Establishment in Wide-Area Networks. *IEEE Journal on Selected Areas in Communications*, 8(3):368–379, April 1990.

[11] V. Firoiu, J. Kurose, and D. Towsley. Efficient Admission Control for EDF Schedulers. In *Proc. IEEE Infocom '97 Conference*, pages 310–317, April 1997.

[12] L. Georgiadis, R. Guerin, and A. Parekh. Optimal Multiplexing on a Single Link: Delay and Buffer Requirements. In *Proc. IEEE Infocom '94*, pages 524–532, June 1994.

[13] L. Georgiadis, R. Guérin, V. Peris, and K. Sivarajan. Efficient network QoS provisioning based on per node traffic shaping. *IEEE/ACM Transactions on Networking*, 4(4):482–501, August 1996.

[14] S. J. Golestani. A Framing Strategy for Congestion Management. *IEEE Journal on Selected Areas In Communications*, 9(7):1064–1077, September 1991.

[15] S. Keshav. *An Engineering Approach to Computer Networking.* Addison Wesley, 1997.

[16] J. Liebeherr, D. E. Wrege, and Domenico Ferrari. Exact Admission Control in Networks with Bounded Delay Services. *IEEE/ACM Transactions on Networking*, 4(6):885–901, December 1996.

[17] D. Verma, H. Zhang, and D. Ferrari. Guaranteeing Delay Jitter Bounds in Packet Switching Networks. In *Proc. Tricomm '91*, Chapel Hill, North Carolina, April 1991.

[18] H. Zhang. *Service Disciplines for Packet-Switching Integrated-Services Networks.* PhD thesis, University of California - Berkeley, November 1993.

[19] H. Zhang. Providing End-to-End Performance Guarantees Using Non-Work-Conserving Disciplines. *Computer Communications*, 18(10):769–781, October 1995.

[20] H. Zhang. Service Disciplines for Guaranteed Performance Service in Packet-Switching Networks. *Proceedings of the IEEE*, 83(10):1374–1399, October 1995.

[21] H. Zhang and D. Ferrari. Rate-Controlled Static-Priority Queueing. In *Proc. IEEE Infocom '93*, pages 227–236, April 1993.

[22] H. Zhang and D. Ferrari. Improving Utilization for Deterministic Service in Multimedia Communication. In *1994 International Conference on Multimedia Computing and Systems*, pages 295–304, Boston, MA, May 1994.

[23] H. Zhang and S. Keshav. Comparison of Rate-Based Service Disciplines. In *Proc. ACM Sigcomm*, pages 113–121, Zurich, Switzerland, September 1991.

[24] Q. Z. Zheng and K. G. Shin. On the Ability of Establishing Real-Time Channels in Point-to-Point Packet Switching Networks. *IEEE Transactions on Communications*, 42(3):1096–1105, March 1994.

## Proof of Theorem 2

The definition of earliness given in Eqn. (3) specifies that the earliness of an arrival from flow $i$ at time $t$, $\varepsilon_i(t)$, is the greatest lower bound, i.e., the *infimum*, of the set of $x$'s satisfying:

$$A_i^{in}[0,t] \leq \inf_{0\leq\tau\leq t}\{A_i^{in}[0,t-\tau] + A_i^*(\tau + x)\} \tag{6}$$

In our proof of Theorem 2, we verify that $\varepsilon_i(t) = \sup_{0\leq\tau\leq t}\{A_i^{*,-1}(A_i^{in}(t-\tau,t]) - \tau\}$ is indeed the greatest lower bound which satisfies Eqn. (6). For this, we need to perform two steps:

> **Step 1:** We show that $x = \sup_{0\leq\tau\leq t}\{A_i^{*,-1}(A_i^{in}(t-\tau,t]) - \tau\}$ satisfies Eqn. (6)
>
> **Step 2:** We show that, for any $x$ satisfying Eqn. (6), we have $x \geq \sup_{0\leq\tau\leq t}\{A_i^{*,-1}(A_i^{in}(t-\tau,t]) - \tau\}$

**Step 1:** Assuming that $A_i^{*,-1}$, inverse function of $A_i^*$, exists we can express the arrivals for flow $i$ in the time interval $[0,t]$ as:

$$A_i^{in}[0,t] = A_i^{in}[0,t-\tau] + A_i^*(A_i^{*,-1}(A_i^{in}[0,t] - A_i^{in}[0,t-\tau])) \qquad \{\forall\tau \mid 0\leq\tau\leq t\} \tag{7}$$

Since $A_i^*$ is a non-decreasing function, we can rewrite Eqn. (7) as

$$\begin{aligned} A_i^{in}[0,t] \;\leq\; & A_i^{in}[0,t-\tau] \\ & + A_i^*(\tau + \sup_{0\leq\tau\leq t}\{A_i^{*,-1}(A_i^{in}[0,t] - A_i^{in}[0,t-\tau]) - \tau\}) \quad \{\forall\tau \mid 0\leq\tau\leq t\} \end{aligned} \tag{8}$$

By substituting $x$ in place of $\sup_{0\leq\tau\leq t}\{A_i^{*,-1}(A_i^{in}(t-\tau,t]) - \tau\}$ above, this becomes equivalent to Eqn. (6).

**Step 2:** For any $x$ which satisfies Eqn. (6), by the definition of *infimum* we can rewrite Eqn. (6) as:

$$A_i^{in}[0,t] \quad \leq \quad A_i^{in}[0,t-\tau] + A_i^{*}(\tau + x) \qquad \{\forall\tau \,|\, 0 \leq \tau \leq t\} \tag{9}$$

Assuming that $A_i^{*,-1}$ exists, we obtain via simple manipulations:

$$x \quad \geq \quad A_i^{*,-1}(A_i^{in}[0,t] - A_i^{in}[0,t-\tau]) - \tau \qquad \{\forall\tau \,|\, 0 \leq \tau \leq t\} \tag{10}$$

Using the definition of *supremum*, this equation is equivalent to:

$$x \quad \geq \quad \sup_{0\leq\tau\leq t}\{A_i^{*,-1}(A_i^{in}(t-\tau,t] - \tau\} \tag{11}$$

□