In Proceedings of ACM SIGCOMM'91

Comparison of Rate-Based Service Disciplines*

Hui Zhang and *Srinivasan Keshav*
Computer Science Division
University of California at Berkeley
Berkeley, CA 94720

Abstract

This paper compares six new queue service disciplines that can be implemented at the output queues of switches in a connection-oriented packet switched data network. These are Virtual Clock, Fair Queueing, Delay-Earliest-Due-Date, Jitter-Earliest-Due-Date, Stop-and-Go and Hierarchical Round Robin. We describe their mechanisms, their similarities and differences, and some implementation strategies. In particular, we show why each discipline can or cannot provide bandwidth, delay and delay jitter guarantees. This leads to some interesting conclusions about the relative strengths and weaknesses of each approach.

1 Introduction

High speed networking introduces opportunities for new applications that have stringent performance requirements in terms of throughput, delay, delay jitter [1] and loss rate [3]. Conventional packet switching data networks with window-based flow control and first-come-first-served service discipline cannot provide services with strict performance guarantees. Thus, new rate-based flow control and rate-based service disciplines have been proposed in the context of a connection-oriented network architecture with explicit resource allocation and admission control policies [16].

A rate-based service discipline is one that provides a client with a minimum service rate independent of the traffic characteristics of other clients (though it may serve a client at a rate faster than this minimum). Such a discipline, operating at each switch[2] in the network, manages the following resources: bandwidth, service priority and buffer space. In conjunction with appropriate admission policies, such disciplines allow clients to get performance guarantees in terms of throughput, delay, delay jitter and loss rate.

Several rate-based scheduling disciplines have been proposed recently: Delay-EDD [4, 6], Virtual Clock [16], Fair Queueing[3] [2], the Stop-and-Go protocol [7, 8], the Hierarchical Round Robin Discipline (HRR) [11], and Jitter-EDD [14]. We feel that it is instructive to compare and contrast these disciplines since they are closely related, yet they have some important differences.

In particular, we wish to answer the following questions:

- What are the similarities and differences between the underlying mechanisms?
- What kinds of performance guarantees can be provided?
- What are the buffer space requirements?
- What are the associated admission control policies?
- What are the implementation issues?

We do not intend to select any particular discipline as the 'best'; rather, our intention is to explore the differences between the disciplines. The choice of which discipline to implement depends upon a number of factors, such as the performance requirements, the workload characteristics, network configuration and implementation complexity. Since these are site-specific variables, we do not feel that it is appropriate to declare any discipline the 'winner'.

A comparison between Virtual Clock and Leaky Bucket [13] is presented in [15]. However we feel that comparing Leaky Bucket and Virtual Clock is inappropriate since Leaky Bucket is not a service discipline: it just controls the rate of one channel and does not specify the service order of packets from different channels.

Another similar study [17] compares Virtual Clock, EDD, Stop-and-Go and HRR. It uses TDM as the baseline for comparison and takes a slightly different perspective from ours.

*This research was supported by the National Science Foundation and the Defense Advanced Research Projects Agency (DARPA) under Cooperative Agreement NCR-8919038 with the Corporation for National Research Initiatives, by AT&T Bell Laboratories, Hitachi, Ltd., the University of California under a MICRO grant, and the International Computer Science Institute. The views and conclusions contained in this document are those of the authors, and should not be interpreted as representing official policies, either expressed or implied, of the U.S. Government or any of the sponsoring organizations.

[1] Delay jitter is defined to be the maximum difference between end-to-end delays experienced by any two packets [14].

[2] In the literature, the term 'switch' is used in the context of ATM networks, while 'gateway' or 'router' is used in the Internet environment. In this paper, we will uniformly call all switching elements as 'switches'.

[3] Fair Queueing was first proposed in datagram networks. However, when used in a connection-oriented network architecture with resource allocation and admission control, as considered in this paper, it can also provide bandwidth guarantees as mentioned in [2]

Our study is more complete in the sense that we also compare Fair Queueing, and discuss buffer space requirements, admission control policies and implementation issues.

The guarantees discussed in this paper are deterministic guarantees as defined in [3]: that is, all packets should meet a guarantee even in the worst case. Static allocation of buffers is assumed to prevent packet loss even in the worst case. While statistical guarantees can also be provided by controlled overbooking resources, as proposed in [4, 9], we do not discuss those guarantees in this paper.

2 Background

Guaranteed service requires that the network *protect* clients from two sources of variability: misbehaved users and network load fluctuations. Misbehaved users can send packets at a higher rate than the bandwidth allocated to them and network load fluctuations may cause a higher instantaneous arrival rate from a channel[4] at some switch, even though the channel satisfies the bandwidth allocation constraint at the entrance to the network. Traffic distortion due to network fluctuations suggests that protection should be implemented within the network using a rate-based service discipline. Such a discipline achieves protection by ensuring that each channel gets a guaranteed rate of service regardless of the behavior of other channels.

What should happen when a channel sends packets at a higher rate than the bandwidth allocated to it, but this higher rate will not affect the performance of other channels? Rate-based service disciplines can be classified into two categories depending on the policy they adopt:

- Rate allocating service disciplines will serve packets at the higher rate as long as it will not affect the performance guarantees made to other channels. Delay-EDD, Virtual Clock and Fair Queueing are in this category.

- Rate controlled service disciplines will not serve packets at a higher rate under any circumstances. Stop-and-Go, HRR, Jitter-EDD are in this category.

A rate-based service discipline can also be classified as either work-conserving or non-work-conserving. In a work-conserving discipline, a server is never idle when there is a packet to send. In a non-work-conserving discipline, each packet is assigned, either explicitly or implicitly, an *eligibility time*. Even when the server is idle, if no packets are eligible, none will be transmitted. As will be shown later in the paper, whether a service discipline is work-conserving or non-work-conserving affects both the buffer space requirements and the delay jitter characteristics.

Of the six disciplines considered in the paper, Delay-EDD, Virtual Clock and Fair Queueing are work-conserving, while Stop-and-Go, HRR and Jitter-EDD are non-work-conserving. Notice that all three rate-controlled service disciplines are non-work-conserving. This is not just a coincidence: only non-work-conserving disciplines can place an upper bound on the service rate of a channel.

[4]There are different terms in the literature for the same or similar objects. A channel is also called a connection, a circuit, a conversation, a flow, etc..

We now present intuitive descriptions of the six service disciplines. Technical details are presented in section 3.

2.1 Fair Queueing

The aim of Fair Queueing [2] is simple: if N channels share an output trunk, then each should get 1/N of the bandwidth, with the provision that if any channel uses less than its share, the slack is equally distributed among the rest. This can be achieved by doing a bit-by-bit round robin (BR) service among the channels. This is impractical, and so Fair Queueing tries to emulate BR. Each packet is given a finish number, which is the round number at which the packet would have received service, had the server been doing BR. By servicing packets in order of the finish numbers, it can be shown that Fair Queueing emulates BR [10]. Channels can be given different fractions of the bandwidth by giving them weights; a weight corresponds to the number of bits of service the channel receives per round of BR service.

2.2 Virtual Clock

The Virtual Clock [16] discipline aims to emulate the Time Division Multiplexing (TDM) service discipline in the same way as Fair Queueing emulates BR. Each packet is allocated a virtual transmission time, which is the time at which the packet would have been transmitted were the server actually doing TDM. A simplified example: if a client is to get a service rate of 5 packets/second, incoming packets from that client are stamped with virtual service times 0.2 seconds apart. By sending packets in virtual time order, Virtual Clock can be shown to emulate TDM.

2.3 Delay Earliest-Due-Date

In classic earliest-due-date (EDD) scheduling, each packet is assigned a deadline, and the packets are sent in order on increasing deadlines. The Delay-EDD [4] service discipline is an extension where the server negotiates a service contract with each source. The contract states that if a source obeys a peak and average sending rate, then the server will provide a delay bound. The key lies in the assignment of deadlines to packets. The server sets a packet's deadline to the time at which it should be sent had it been received according to the contract. This is just the expected arrival time added to the delay bound at the server. For example, if a client assures that it will send packets every 0.2 seconds, and the delay bound at a server is 1 second, then the kth packet from the client will get a deadline of $0.2k+1$. By reserving bandwidth at the peak rate, Delay-EDD can assure each channel a hard delay bound.

2.4 Jitter Earliest-Due-Date

The Jitter-EDD discipline [14] extends Delay-EDD to provide delay-jitter bounds (that is, a bound on the minimum as well on the maximum delay). After a packet has been served at each server, it is stamped with difference between its deadline and actual finishing time. A regulator at the entrance of the next switch holds the packet for this period before it is made eligible to be scheduled. This provides the required minimum and maximum delay guarantees

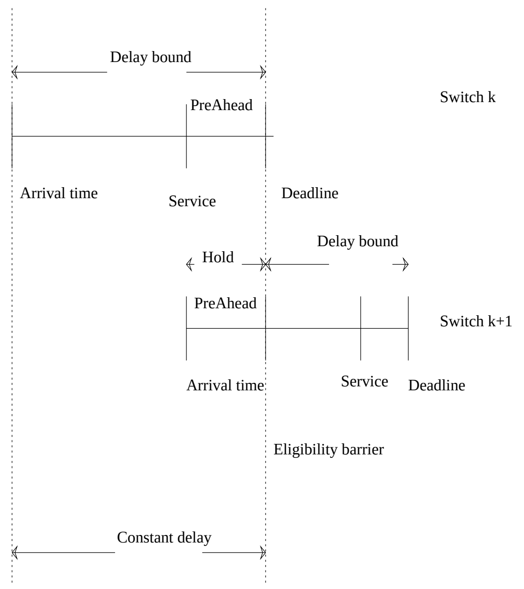

Figure 1: Packet Service in Jitter-EDD

Jitter-EDD is illustrated in Figure 1, which shows the progress of a packet through two adjacent switches. In the first switch, the packet got served PreAhead seconds before its deadline. So, in the next switch, it is made eligible to be sent only after PreAhed seconds. Since a packet obtains a constant delay at each switch, it can be provided a jitter bound.

2.5 Stop-and-Go

The Stop-and-Go service discipline [8] aims to preserve the 'smoothness' property of traffic as it traverses the network. Time is divided into frames. In each frame time, only packets that arrived at the server in the previous frame time are sent. It can be shown that with this scheme, a packet receives both a minimum and a maximum delay as it goes from a source to a destination. Since the delay and delay-jitter bounds are linked to the length of the frame time, Stop-and-Go proposes multiple frame sizes.

2.6 Hierarchical Round Robin

The Hierarchical Round Robin (HRR) [11] server has several service levels, where each level provides round-robin service to a fixed number of slots. A channel is allocated some number of service slots at a selected level, and the server cycles through the slots at each level. The time a server takes to service all the slots at a level is called the frame time at that level. The key to HRR lies in its ability to give each level a constant share of the bandwidth. 'Higher' levels get more bandwidth than 'lower' levels, so the frame time at a higher level is smaller than the frame time at a lower level. Since a server always completes one round through its slots once every frame time, it can provide a maximum delay bound to the channels allocated to that level.

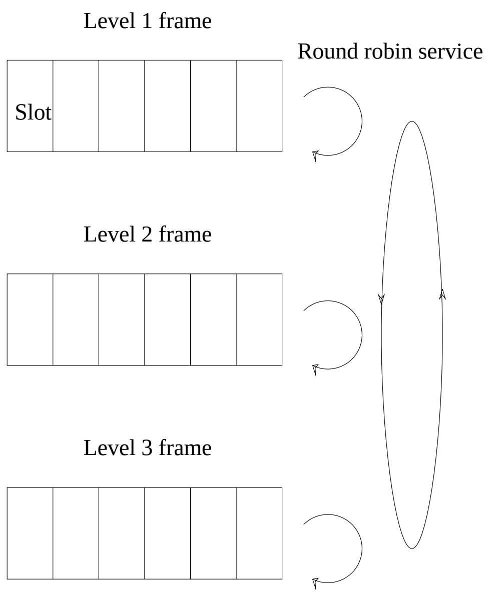

Figure 2: Hierarchical round robin frames

2.7 Traffic Specification

Rate-based service disciplines need to allocate resources per client. This requires clients to specify their traffic type, so that sufficient resources can be reserved by each switch. The traffic specifications in Virtual Clock, HRR and Stop-and-Go are essentially the same: a transmission rate (AR) averaged over an interval (AI). Fair Queueing was described in a datagram network context and no traffic specification was proposed. In this paper, we assume that the same (AR, AI) specification applies to the Fair Queueing as well.

Delay-EDD and Jitter-EDD have a three parameter traffic specification: $Xmin$, which is the minimal packet inter-arrival time, $Xave$, which is an average packet inter-arrival time, and I, which is the interval over which $Xave$ is computed. The two parameters $Xmin$ and $Xave$ are used to characterize bursty traffic and to offer statistical guarantees [3] (as opposed to deterministic guarantees).

In order to compare the three parameter EDD traffic models with the two parameter model for the other disciplines, we need to map the three parameter model to the two parameter model. How should this be done? In EDD disciplines a server uses $Xave$ to do bandwidth allocation, and $Xmin$ to ensure that delay bounds can be met. Thus, for the purpose of comparing the bandwidth allocations of the six disciplines, we can assume that $Xmin$ is equal to $Xave$. Then, the following correspondence holds:

$$AR \Longleftrightarrow \frac{1}{Xmin}$$

Note that this does not mean that clients send data in a smooth continuous stream; it merely implies that a source never sends data faster than the negotiated average rate.

Virtual Clock	Fair Queueing	Delay-EDD
$auxVC_i^k \leftarrow max(AT, auxVC_i^k)$	$F_i^k \leftarrow max(R^k, F_i^k)$	$ExD_i^k \leftarrow ExD_i^k + Xmin_i$
$auxVC_i \leftarrow auxVC_i + Vtick_i$	$F_i^k \leftarrow F_i^k + \frac{P_i}{n_i}$	$ExD_i^k \leftarrow max(AT + d_i^k, ExD_i^k)$
Stamp packet with $auxVC_i^k$	Stamp packet with F_i^k	Stamp packet with ExD_i^k

Table 1: Comparision of Virtual Clock, Fair Queueing and Delay-EDD

3 Rate Based Service Disciplines

3.1 Virtual Clock, Fair Queueing and Delay-EDD

In each of Virtual Clock, Fair Queueing and Delay-EDD, at each switch, there is a state variable associated with each channel to monitor and enforce the rate for that channel. In Virtual Clock, the variable is called auxiliary Virtual Clock [5] ($auxVC$); in Fair Queueing, it is called Finish Number (F); in Delay-EDD, it is called Expected Deadline (ExD)[6].

In all three cases, $auxVC$, F and ExD are used as priority indices of packets; packets are served in the order of increasing priority index. The computations of $auxVC$, F and ExD are shown in Table 1. The subscript i is the channel number, the superscript k is the switch number and d_i^k is the local delay bound [7] assigned to the channel at channel establishment time. In Virtual Clock and delay-EDD, AT is the packet arrival time; in Fair Queueing, R^k is the number of rounds that have been completed for a hypothetical bit-by-bit round robin server at switch k, n_i is the weighting factor, and P_i is the packet length measured in number of bits.

Virtual Clock and Fair Queueing are completely equivalent, and is clear from:

$$AT \Longleftrightarrow R^k$$

$$auxVC_i^k \Longleftrightarrow F_i^k$$

$$Vtick_i \Longleftrightarrow \frac{P_i}{n_i}$$

Hence, we do not discuss Fair Queueing any further.

If we combine the two state equations for both Virtual Clock and Delay-EDD, we have

- In Virtual Clock,

$$auxVC_i^k \leftarrow max(AT + Vtick_i, auxVC_i^k + Vtick_i)$$

- In Delay-EDD,

$$ExD \leftarrow max(AT + d_i^k, ExD_i^k + Xmin_i)$$

We have the following mapping:

$$auxVC_i^k \Longleftrightarrow ExD$$

$$Vtick_i \Longleftrightarrow Xmin_i \ or \ d_i^k$$

Recall that in Delay-EDD, $Xmin_i$ is the minimum packet inter-arrival time and d_i^k is the local delay bound; in Virtual Clock, $Vtick_i$ is the average packet inter-arrival time. As can be seen, there are two differences here:

1. Delay-EDD imposes the restriction of minimum spacing between packets, while Virtual Clock does not.
2. Delay-EDD decouples the delay and bandwidth requirements by using both $Xmin_i$ and d_i^k, while Virtual Clock just has one counterpart $Vtick_i$.

The two differences are the reason why Delay-EDD, in conjunction with the establishment and admission control scheme described in [4], can provide both delay and bandwidth guarantees, while Virtual Clock can provide only bandwidth guarantees.

Virtual Clock does have a mechanism to handle priorities – the AT in the equation can be replaced by $AT - Priority$. In this case, packets from channels with higher $Priority$ will get lower average delay. However, without imposing minimum spacing between packets and a priority allocation scheme similar to the delay bound allocation scheme in [4], Virtual Clock cannot provide deterministic delay bounds.

3.2 Delay-EDD and Jitter-EDD

The comparison between Delay-EDD and Jitter-EDD is shown in Table 2. Notice that the state equations for Delay-EDD are written in a form different from that in Table 1. This is only for the sake of exposition. It can be shown that these two representations of the state equations are equivalent. The variable ExA_i^k is the expected arrival time of the next packet. It is related to ExD_i^k of Table 1 by:

$$ExD_i^k \Longleftrightarrow ExA_i^k + Xmin_i$$

The variable $Ahead$ is the amount of time the packet arrives ahead of schedule at the current switch.

As can be seen in Table 2, there are two differences between Delay-EDD and Jitter-EDD:

- There is one more term in calculation of $Ahead$ in Jitter-EDD: $PreAhead$. $Prehead$ is a value carried in from the previous switch; it tells the switch *how much the packet is ahead of schedule with respect to the previous switch*, i.e., the difference between the deadline and the actual finishing time of the packet in the previous switch.
- Jitter-EDD holds the packet for $Ahead$ units of time, and then calculates its deadline and hands it to the scheduler, while Delay-EDD extends the deadline by

[5] There is another variable $VirtualClock$ associated with each connection that is periodically checked to measure the difference between the actual rate and the allocated rate. $VirtualClock$ is irrelevant in this discussion.

[6] Notation used in the paper is summarized in the Appendix.

[7] The delay bound is the maximum allowed difference between the arrival time and departure time of a packet from the kth switch for the ith channel [4].

Delay-EDD	Jitter-EDD
$ExA_i^k \leftarrow max(ExA_i^k + Xmin_i, AT)$	$ExA_i^k \leftarrow max(ExA_i^k + Xmin_i, AT)$
$Ahead \leftarrow ExA_i^k - AT$	$Ahead \leftarrow max(ExA_i^k - AT, PreAhead)$
	Hold the packet for $Ahead$ time
$Deadline \leftarrow AT + Ahead + d_i^k$	$Deadline \leftarrow AT + Ahead + d_i^k$
Stamp packet with $Deadline$	Stamp packet with $Deadline$

Table 2: Comparision between Delay-EDD and Jitter-EDD

Ahead time units and hands the packet immediately to the scheduler.

It is this holding time that makes Jitter-EDD a non-work-conserving discipline. It has been shown in [14] that Jitter-EDD has the following properties:

1. The traffic pattern of a channel is preserved at each switch in spite of network load fluctuation; if the traffic from a channel obeys the $(Xmin, I)$ specification at the entrance of the network, then this traffic will also obey $(Xmin, I)$ at each switch.
2. Let switch 0 be the source, d_i^h be the local delay bound assigned to channel i at switch h and t_i^h be the service time of the packet at switch h. If a packet from channel i enters the network at time $EnterTime$, the earliest time it can arrive at switch k is $(EnterTime + \sum_{h=0}^{k-2} d_i^h + t_i^{k-1})$[8]; the earliest time and the latest time the packet can leave the switch are $(EnterTime + \sum_{h=0}^{k-1} d_i^h + t_i^k)$ and $(EnterTime + \sum_{h=0}^{k} d_i^h)$, respectively.
3. From property 2, it immediately follows that the maximum residence time of a packet on channel i at switch k is $d_i^{k-1} + d_i^k - t_i^{k-1}$.

Property 2 provides jitter bounds for channels[14]; properties 1 and 3 give buffer bounds to prevent packet loss, which will be discussed in more detail in Section 3.4. As will be shown in the next section, the other two non-work-conserving service disciplines, Stop-and-Go and HRR, have similar properties.

However, it should be noticed that the three work-conserving disciplines: Virtual Clock, Fair Queueing and Delay-EDD, do not have such properties. The following statements, that are in some sense parallel to properties 1 through 3 above, apply to Virtual Clock, Fair Queueing and Delay-EDD.

1. For all three service disciplines, even if the traffic pattern of a channel obeys an average rate bound at the entrance of the network, a switch may face a higher instantaneous input rate over that channel due to network load fluctuations.
2. For Delay-EDD, if a packet from channel i enters the network at time $EnterTime$, the earliest time it can arrive at switch k is $(EnterTime+\sum_{h=0}^{k-1} t_i^h)$; the earliest time and the latest time it can leave the switch are $(EnterTime+\sum_{h=0}^{k} t_i^h)$ and $(EnterTime+\sum_{h=0}^{k} d_i^h)$, respectively. Since Virtual Clock and Fair Queuing do not provide a worst-case delay bound, they do not have a similar property.
3. For Delay-EDD, the maximum residence time of a packet on channel i in switch k is $(\sum_{h=0}^{k} d_i^h - \sum_{h=0}^{k-1} t_i^h)$.

[8]For simplicity, the propagation delay is not included in this calculation

3.3 Stop-and-Go and HRR

Both Stop-and-Go and HRR use a multi-level framing strategy. In this section, one level of framing is examined before discussing multi-level framing

For Stop-and-Go and HRR, the time axis is divided into periods of some constant length T, each called a frame[9]. Bandwidth is allocated to each channel as a certain fraction of the frame time.

Stop-and-Go defines *departing* and *arriving* frames for each link. At each switch, the arriving frame of each incoming link is mapped to the departing frame of the output link by introducing a constant delay θ, where $0 \leq \theta < T$. All the packets from one arriving frame of an incoming link and going to output link l are delayed by θ and put into the corresponding departing frame of l. According to the stop-and-go discipline, the transmission of a packet that has arrived on any link l during a frame f should always be postponed until the beginning of the next frame. Since packets arriving during a frame f of the output link are not eligible for transmission until the next frame, Stop-and-Go is a non-work-conserving service discipline. Within each frame, the service order of packets is arbitrary[10]

One level HRR is equivalent to a non-work-conserving round robin (or TDM) service discipline. Each channel is assigned a fraction of the total available bandwidth, and receives that bandwidth in each frame, if it has sufficient packets available for service. The server ensures that the assigned bandwidth is also the maximum service rate for that channel in each frame. This means that, in a frame, after providing a channel's bandwidth allocation, even if the server is available and more packets from that channel are queued for transmission, the packets will not be served until the next frame. Since these extra packets are not eligible for transmission until the next frame, HRR is a non-work-conserving service discipline. Within each frame, the service

[9]The notation FT is used in HRR, while T is used in Stop-and-Go; we adopt T for both in this paper.

[10]FIFO is used as the service discipline within each frame in [7, 8]; however, as pointed out by the author, this is just for convenience and is not part of Stop-and-Go.

order of packets is arbitrary.

Statements similar to those we made above for Jitter-EDD are as follows for Stop-and-Go:

1. If the traffic over a channel obeys, at the entrance of the network, the average rate constraint, where the average interval is T, the traffic will obey the same constraint throughout the network[11].

2. If a packet from channel i enters the network at time $EnterTime$, the earliest time it can arrive at switch k is $EnterTime + \sum_{h=0}^{k-1} \theta_h + (k-2)T$; the earliest time and the latest time it can leave switch k are $(EnterTime + \sum_{h=0}^{k} \theta_h + (k-1)T)$ and $(EnterTime + \sum_{h=0}^{k} \theta_h + kT)$, respectively, where switch 0 is the source, and θ_h is the synchronization time between the input link and the outlink at switch h, $0 \leq \theta_h < T$.

3. From property 2, the maximum packet residence time in switch k is: $2T + \theta_k$.

Again, property 2 gives bounds on jitter, and properties 1 and 3 give bounds on buffer space requirements.

For HRR we have:

1. If the traffic from a channel obeys, at the entrance of the network, the average rate constraint, the traffic will obey the same constraint at each switch. If the traffic rate is higher at the entrance, the first switch will smooth out the traffic, and the traffic of the channel will obey the average rate constraint at downstream switches.

2. If a packet over channel i enters the network at time $EnterTime$, the earliest time it can arrive at switch k is $EnterTime + \sum_{h=0}^{k-1} t_i^h$; the earliest time and the latest time it can leave the switch are $(EnterTime + \sum_{h=0}^{k} t_i^h)$ and $(EnterTime + (k+!)T)$, respectively. Here, switch 0 is the source, t_i^h is the service time of a channel i packet at switch h.

3. The maximum packet residence time in a switch is $2T$.

Notice that property 2 here does not give a tight bound on the minimum delay that a packet can incur as in Jitter-EDD and Stop-and-Go. However, HRR does provide an upper bound on delay. Thus HRR provides delay bounds but not tight delay jitter bounds[12]. Also, in Jitter-EDD and Stop-and-Go, property 3 is derived from property 2, while in HRR property 3 holds by itself due to the non-work-conserving service discipline.

If the packet size is assumed to be constant, we have the following equation for both HRR and Stop-and-Go:

$$ServiceQuantum = \frac{PacketSize}{T}$$

[11]This is called the (r, T)-smoothness property in [7, 8], where r is the average rate and T is the frame time.

[12]Delay jitter is defined as the maximum difference between delays experienced by any two packets in a channel. Jitter is defined differently in [11]. Providing a delay jitter bound is sufficient to provide the jitter bound as defined in [11], but the other direction does not hold.

	Stop-and-Go	**HRR**
Delay Bound	$H*T < Q < 2*H*T$	$2*H*T$
Jitter Bound	$2*T$	–

Table 3: Comparision between Stop-and-Go and HRR

where the service quantum is the minimum granularity of bandwidth allocation.

The delay bounds that can be provided at each switch by Stop-and-Go and HRR are $T + \theta$ $(0 \leq \theta < T)$ and $2T$, respectively. From these two equations, it is clear that the service quantum and frame time cannot decrease simultaneously, i.e., low delay bound and fine granularity of bandwidth allocation cannot be achieved simultaneously.

To get around this coupling between delay and service quantum, both HRR and Stop-and-Go propose the use of multiple frames with different frame times. In this case, it is possible to provide low delay bounds to some channels by putting them in frames with a smaller frame time, and to allocate bandwidth with fine granularity to other channels by putting them in levels with a larger frame time. However, the coupling between delay and service quantum still exists within each frame.

In Multi-Frame-Time Stop-and-Go and Multi-Frame-Time HRR, it is possible that a channel is assigned to frames with different sizes in different switches. The three properties discussed above for Stop-and-Go and HRR will hold if the channel is assigned to frames with same frame time at each switch. Under the same assumption, Table 3 summarizes the end-to-end delay characteristics of the channel for each of the two service disciplines. H and T are the number of hops of the channel and the frame time, respectively. The exact value of Q is $\sum_{h=0}^{H} \theta_h + HT$. When a channel is allocated different frame times at different switches, similar worst case delay bounds hold for both Stop-and-Go and HRR. However, the delay expressions are rather complicated, and so we will not discuss them here.

The most important difference between Stop-and-Go and HRR is that Stop-and-Go synchronizes the arriving frames of the input links and the departing frames of the output link at each switch. There are two implications:

1. By this synchronization, tight delay jitter bounds can be provided by Stop-and-Go.

2. The synchronization also means that in Multi-Frame-Time Stop-and-Go, the frame times of a channel along the path should be non-decreasing. HRR does not have this restriction. This gives HRR more flexibility in assigning channels with different frame times at different switches.

Another difference between Stop-and-Go and HRR is their response to ill-behaved gateways that allow a channel to send data at more than AR. A HRR server queues data per channel, so if a channel sends more data than it ought to, it can only hurt itself. On the other hand, a Stop-and-Go server has no way to prevent itself from being flooded, and misbehaved users (that evade flow regulation at the input to

Delay-EDD	$\frac{\sum_{h=0}^{k} d_i^h}{Xmin_i}$
Jitter-EDD	$\frac{d_i^{k-1}+d_i^k}{Xmin_i}$
Stop-and-Go	$(2T_i + \theta_k) \times AR_i$
HRR	$2T_i \times AR_i$

Table 4: Buffer space requirements

the network) could cause other users to drop packets. This could be a significant problem in some networks.

3.4 Buffer Requirement

In this paper, we assume that the buffer space is statically allocated on a per-channel basis to prevent packet loss even in the worst case.

The buffer space required for a channel can be calculated by multiplying the maximum packet arrival rate by the longest residence time of a packet in a switch. This corresponds to the first and third properties discussed above for different service disciplines. Since the delay characteristics are unclear for Virtual Clock and Fair Queueing, only the other four disciplines are discussed here. The amount of buffer space required for channel i at node k are given in Table 4. As mentioned before, the following correspondence holds:

$$AR_i \Longleftrightarrow \frac{1}{Xmin_i}$$

If local delay is assumed to be the same for each switch in Delay-EDD and Jitter-EDD, the buffer space requirements for them are $\frac{(k+1)d_i}{Xmin_i}$ and $\frac{2d_i}{Xmin_i}$ respectively.

It can be seen that the buffer space requirements for the three non-work-conserving disciplines are almost constant for each node traversed by the channel, while the buffer space requirement for the work-conserving Delay-EDD increases linearly for each node along the path.

3.5 Admission Control Policies

In order to offer performance guarantees in terms of delay and bandwidth, resources need to be reserved at each node for each channel. Besides buffer space as described in the previous section, two other resources, bandwidth and schedulability (discussed below) need to be managed. For the six service disciplines, if deterministic guarantees [3] are to be offered, the overall bandwidth allocated to the channels on any link should not exceed the bandwidth of that link.

To provide delay guarantees, schedulability should also be considered. As pointed out in [4], scheduling saturation can occur even if bandwidth is not overbooked. For example, two packets with delay bounds of 4 time units and service time of 3 time units may arrive at a node at the same time. In this situation, it is impossible to meet the delay bounds for both packets.

It can be shown that scheduling saturation can be avoided if for any packet arrival pattern, a output sequence (a schedule) can be constructed such that, even in the worst case, all the packets between any two regeneration points meet their deadlines. In Stop-and-Go and HRR, the delay bound of all the channels is equal to the frame time. The starting of a frame is a regeneration point. If bandwidth is not overbooked, all the packets assigned to one frame are serviced in that frame. Thus, in Stop-and-Go and HRR scheduling saturation is automatically avoided by the framing strategy.

In the two EDD schemes, each channel can have an arbitrary delay bound. There does not exist a natural regeneration point as the starting of a frame in Stop-and-Go and HRR, so a schedulability test has to be performed at the channel establishment time. The test constructs the worst case packet arrival pattern from all channels and examines if all the packets can meet their deadlines in such a case.

Though, framing in Stop-and-Go and HRR automatically handles the problem of scheduling saturation, it introduces the problem of coupling between delay and bandwidth allocation granularity. The EDD schemes avoid this coupling and allow a more flexible assignment of delay, but require an explicit schedulability test. This can be illustrated in the following example. Assume there are four channels sharing one output link with each channel assigned one fourth of the bandwidth. The service time of a packet for each channel is 1 time unit. In Stop-and-Go and HRR, the smallest possible frame time is 4, i.e., the smallest delay bound for all four channels is 4. In the EDD schemes, it can be shown that the four channels can have delay bounds of 2, 3, 4, 4 respectively, i.e., two channels can have lower delays than in the case of Stop-and-Go and HRR.

4 Implementation

Thus far, we have compared the service disciplines without regard to the implementation cost. This cost is hard to quantify, since it depends on whether the implementation is done in hardware or software, and the choice of data structure and algorithms. Thus, we will only sketch the steps necessary to implement the algorithms, and present some implementation issues.

4.1 Fair Queueing, Virtual Clock and Delay-EDD

The three algorithms have been shown to be quite similar, and their implementation is identical. In each case, when a packet arrives, per-channel state has to be retrieved and updated. Then, the packet is stamped with a priority index and placed in a priority queue. When the output trunk becomes free, the server removes the packet at the head of the priority queue and sends it.

The efficient implementation of a priority queue for Fair Queueing in software has been investigated in [12], and the results indicate that a simple linked list is a good priority queue implementation. This result should hold for the other two disciplines as well.

An efficient hardware scheme to implement a priority queue is a dictionary machine [13]. Here, a large number of processors (one per packet in the buffer) are placed at the leaves of a data distribution tree. Each processor stores a priority index and has a bidirectional data path to its two

[13]Suggested by S. Rajagopal at UC Berkeley

neighbors. A priority index is introduced at the root of the tree and asynchronously propagates to the leaves. Each processor at a leaf node compares the incoming value with the one that it has stored, and accordingly shifts its stored value left or right. By coordinating these shifts, it is possible to accommodate the new index into one of the processors at the leaves such that the stored values at the leaves form an ordered list. In this way, a single hardware step can update the priority queue. Deleting values from the tree is accomplished by a single right shift.

4.2 Jitter-EDD

Jitter-EDD can be implemented in the same way as Delay-EDD with the addition of a regulator. A regulator has to delay a packet an additional amount of time in order to reconstruct the input traffic stream. It can do so by constructing lists of packets for each delay value and placing them in a calendar queue [1]. This is described in [5].

4.3 Stop-and-Go

Assuming the service order within each frame time is FIFO, Stop-and-Go can be implemented by a number of FIFO queues, a service controller, a transmission queue and a transmission server [8]. Each queue corresponds to a frame time and stores packets for channels that are allocated at that frame time. At the beginning of each frame time, the service controller is interrupted and it places the packets present in the corresponding queue (which have been collected in the previous frame time) at the *head* of the transmission queue. The server services packets from the head of the transmission queue. In this way, the service of the packets from a frame of larger frame time may be interrupted, on a non-preemptive basis, by the starting of a new frame of smaller frame time.

The periodic interruption of the service controller requires a set of timers. No particular design has been suggested for the timers in the published literature. The major cost associated with the implementation is in the design of the timers and we conjecture that it can be implemented by a scheme resembling HRR. Another implementation issue is the synchronization of incoming and outgoing frames. This requires the synchronization of the output frames at all the output links and introducing a fixed delay at each input link.

4.4 Hierarchical Round Robin

HRR is specifically designed to be implemented in hardware. Three counters associated with each service level, and these are loaded at the beginning of each frame time. Transfer of control between levels is based on simple combinational logic operating on the values of the counters. The choice of the next packet to send involves only one step in the hardware, and can be done at very high speeds. Per-channel queueing of the data is done using custom VLSI (this has been built by C.R. Kalmanek and R.C. Restrick at AT&T Bell Laboratories, Murray Hill), and a prototype implementation of HRR has been designed to run at 1.3 Gbps.

5 Conclusions

In this paper, six rate-based service disciplines have been examined and compared in the context of connection-oriented packet switching networks. The emphasis has been on examining their mechanisms and the specific properties that can provide delay, delay jitter and bandwidth guarantees. Table 5 summarizes the results.

We showed that the Virtual Clock and Fair Queueing disciplines are identical in their behavior, and that if the traffic is regular, they are both identical to Delay-EDD. While Fair Queueing and Virtual Clock cannot provide worst case delay bounds, Delay-EDD is able to do so because a) it restricts the peak arrival rate to be $1/Xmin$, b) reserves bandwidth at this peak rate, and c) the admission control tests assure that neither bandwidth saturation nor scheduler saturation can occur.

We have shown that using a regulator at the input to a Delay-EDD server allows us to provide delay jitter bounds. The key idea in using a regulator is that it restores the traffic characterization to what it was when the traffic entered the network, which ensures that the network traffic load is *regular* at all the servers. This makes the network easier to control. The synchronization of frames in Stop-and-Go achieves precisely the same purpose, which is why both the disciplines are able to provide delay jitter bounds. Since HRR does not restore traffic to its original form, delay jitter control is not possible, but bandwidth and delay guarantees can be made.

Both Stop-and-Go and HRR have the notion of frames. This simplifies the allocation of bandwidths, delays and jitters. However, a coupling is introduced between the frame size, the delay and the bandwidth allocation granularity. In both cases, multiple frame sizes are used to alleviate the coupling, but the solution is not complete. It would be interesting to see if it is possible to come up with a discipline that allows complete decoupling of these three elements.

While work-conserving service disciplines are dominant in conventional networks, non-work-conserving service disciplines exhibit features that are suitable for offering guaranteed network performance. We feel that analysis and implementation of non-work-conserving disciplines is a promising area for future research.

This study limits its scope to smooth traffic specification, deterministic performance guarantees and static buffer allocation scheme. Further work should study the behavior of different service disciplines when burstiness is introduced and statistical guarantees are offered.

6 Acknowledgements

We would like to thank Prof. D. Ferrari and the members of the Tenet group (at UC Berkeley and International Computer Science Institute) for their stimulating discussions and thoughtful criticism.

References

[1] Randy Brown. Calendar queues: A fast O(1) priority queue implementation for the simulation event set prob-

	VCL	**FQ**	**D-EDD**	**SG**	**HRR**	**J-EDD**
Work Conserving	Yes	Yes	Yes	No	No	No
Throughput Guarantee	Yes	Yes	Yes	Yes	Yes	Yes
Delay Guarantee	No	No	Yes	Yes	Yes	Yes
Jitter Guarantee	No	No	No	Yes	No	Yes
Constant Buffer	No	No	No	Yes	Yes	Yes
Protection	Yes	Yes	Yes	No	Yes	Yes

Table 5: Summary of Results

lem. *Communications of the ACM*, 31(10):1220–1227, October 1988.

[2] Alan Demers, Srinivasan Keshav, and Scott Shenker. Analysis and simulation of a fair queueing algorithm. In *Journal of Internetworking Research and Experience*, pages 3–26, October 1990. Also in Proc. ACM SIGCOMM'89, pp 3-12.

[3] Domenico Ferrari. Client requirements for real-time communication services. *IEEE Communications Magazine*, 28(11), November 1990. also RFC 1193.

[4] Domenico Ferrari and Dinesh Verma. A scheme for real-time channel establishment in wide-area networks. Technical Report TR-89-036, International Computer Science Institute, Berkeley, California, May 1989.

[5] Domenico Ferrari and Dinesh Verma. Quality of service and admission control in ATM networks. Technical Report TR-90-064, International Computer Science Institute, Berkeley, California, December 1990.

[6] Domenico Ferrari and Dinesh 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.

[7] S. Jamaloddin Golestani. Congestion-free transmission of real-time traffic in packet networks. In *Proceedings of IEEE INFOCOM'90*, pages 527–542, San Francisco, California, June 1990. IEEE Computer and Communication Societies.

[8] S. Jamaloddin Golestani. A stop-and-go queueing framework for congestion management. In *SIGCOMM'90 Symposium, Communications Architecture & Protocols*, pages 8–18, Philadelphia Pennsylvania, September 1990. ACM SIGCOMM.

[9] S. Jamaloddin Golestani. Duration-limited statistical multiplexing of delay-sensitive traffic in packet networks, April 1991.

[10] Albert Greenberg and Neil Madras. How fair is fair queueing? *Proc. Performance '90*, 1990.

[11] Charles R. Kalmanek, Hemant Kanakia, and Srinivasan Keshav. Rate controlled servers for very high-speed networks. In *IEEE Global Telecomminications Conference*, San Diego, California, December 1990.

[12] Srinivasan Keshav. On the efficient implementation of fair queueing, 1991. To appear in Journal of Internetworking Research and Experience.

[13] M. Sidi, W.-Z. Liu, I Cidon, and I Gopal. Congestion control through input rate regulation. In *IEEE Global Telecomminications Conference*, December 1989.

[14] Dinesh Verma, Hui Zhang, and Domenico Ferrari. Guaranteeing delay jitter bounds in packet switching networks. In *Proceedings of Tricomm'91*, Chapel Hill, North Carolina, April 1991.

[15] Abel Weinrib and L.T. Wu. Virtual clocks and leaky buckets: Flow control protocols for high speed networks. In *Second IFIP International Workshop on Protocols For High-Speed Networks*, Palo Alto, California, November 1990.

[16] Lixia Zhang. *A New Architecture for Packet Switched Network Protocols.* PhD dissertation, Massachusetts Institute of Technology, July 1989.

[17] Lixia Zhang. A comparison of traffic control algorithms for high-speed networks. In *The 2nd Annual Workshop on Very High Speed Networks, Proceedings Supplement*, Greenbelt, Maryland, March 1991.

A Notation

A.1 Virtual Clock

AR:	average rate.
AI:	average interval over which AR is computed.
$auxVC_i^k$:	auxiliary Virtual Clock of channel i at switch k.
$Vtick^i$:	average packet inter-arrival time for channel i.
AT:	arrival time of the packet

A.2 Fair Queueing

P_i:	packet length measured in number of bits.
R^k:	number of rounds that have been completed for a hypothetical bit-by-bit round robin server at switch k.
F_i^k:	the finish number of channel i at switch k.
n_i:	the relative weight of channel i.

A.3 Delay-EDD and Jitter-EDD

$Xmin_i$:	minimum packet inter-arrival time of channel i.
$Xave_i$:	average packet inter-arrival time of channel i.
t_i^k:	maximum service time of a channel i packet at switch k.
I:	interval over which $Xave$ is computed.
ExD_i^k:	expected deadline of next packet of channel i at switch k.
ExA_i^k:	expected arrival time of next packet of channel i at switch k.
d_i^k:	the local delay bound assigned to channel i at switch k.
$Ahead$:	the amount of time the packet arrives ahead of schedule at current switch.
$PreAhead$:	the amount of time the packet is transmitted ahead of schedule previous switch.

A.4 Stop-and-Go and HRR

T:	the frame time.
θ_k:	the amount of time between the end of the arriving frame of an incoming link and the beginning of the departing frame of an outgoing link.