Service Disciplines For Guaranteed Performance Service in Packet-Switching Networks

HUI ZHANG, MEMBER, IEEE

Abstract— **While today's computer networks supports only best-effort service, Future packet-switching integrated-services networks will have to support real-time communication services that allow clients to transport information with performance guarantees expressed in terms of delay, delay jitter, throughput and loss rate. An important issue in providing guaranteed performance service is the choice of the packet service discipline at switching nodes.**

In this paper, we survey several service disciplines that are proposed in the literature to provide per-connection end-to-end performance guarantees in packet-switching networks. We describe their mechanisms, their similarities and differences, and the performance guarantees they can provide. Various issues and tradeoffs in designing service disciplines for guaranteed performance service are discussed, and a general framework for studying and comparing these disciplines are presented.

I. Introduction

Communication systems have been revolutionized by technological advances in the last decade. The speed and capacity of various components in a communication system, such as transmission media, switches, memory, processors, have all followed technological curves that have grown either linearly or exponentially over the last ten years [18]. At the periphery of the network, driven by the same underlying technology — microelectronics, the capability of computers has been drastically increased while the cost has been significantly reduced. The advent of high speed networking has introduced opportunities for new applications such as video conferencing, scientific visualization and medical imaging. These applications have stringent performance requirements in terms of throughput, delay, delay jitter and loss rate. Current packet-switched networks (such as the Internet) offer only a best-effort service, where the performance of each session can degrade significantly when the network is overloaded. There is an urgent need to provide network services with performance guarantees and to develop algorithms supporting these services.

One of the most important issues in providing guaranteed performance services is the choice of the packet service discipline at the switch. In a packet-switching network, packets from different connections will interact with each other at each switch; without proper control, these interactions may adversely affect the network performance experienced by clients. The service disciplines at the switching nodes, which control the order in which packets are serviced, determine how packets from different connections interact with each other.

Hui Zhang is with the School of Computer Science, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213-3891

Although service disciplines and associated performance problems have been widely studied in the contexts of hard real-time systems and queueing systems, results from these studies are not directly applicable in the context of providing guaranteed performance service in packet-switching networks. Analyses of hard real-time systems usually assume a *single server* environment, *periodic* jobs, and the job delay bounded by its *period* [53]. However, the network traffic is *bursty*, and the delay constraint for each individual connection is *independent* of its bandwidth requirement. In addition, bounds on *end-to-end* performance need to be guaranteed in a *networking* environment, where traffic dynamics are far more complex than in a single server environment. Queueing analysis is often intractable for realistic traffic models. Also, classical queueing analyses usually study *average* performance for *aggregate* traffic [32], [57], while for guaranteed performance service *performance bounds* need to be derived on a *per-connection* basis [13], [38]. In addition to the challenge of providing end-to-end per-connection performance guarantees to heterogeneous and bursty traffic, service disciplines must be *simple* so that they can be implemented at very high speeds.

Recently, a number of new service disciplines that are aimed to provide per-connection performance guarantees have been proposed in the context of high-speed packet-switching networks [12], [16], [21], [22], [26], [56], [62], [67]. Also, new analysis techniques have been proposed to address the performance issues of these disciplines [1], [5], [8], [9], [34], [35], [37], [40], [42], [48], [49], [58], [60], [63], [64], [66], [68]. In this paper, we give an overview of the proposed service disciplines, and discuss the issues and tradeoffs in designing service disciplines in providing guaranteed performance service in packet-switching networks.

The rest of the paper is organized as follows. In Section II, we review general issues associated with providing performance guarantees in packet-switching networks and demonstrate the important role of service disciplines in the network control architecture. Sections III and IV discuss the two classes of service disciplines, work-conserving and non-work-conserving disciplines respectively. In each of the two sections, a brief description of each discipline is first given before a general framework is presented to show the similarities and differences among them. The end-to-end delay characteristics, buffer space requirement, and implementation issues of each discipline are then discussed

within the framework. In Section V, we summarize the paper by providing a taxonomy for classifying and comparing existing solutions.

II. Background

A. Network Model

We consider a network with arbitrary topology of links and switches[1]. Link are assumed to have bounded delay. Switches are assumed to be 'non-blocking', i.e., when packets arrive at an input link, they can be routed directly to the appropriate output links without switching conflicts. Packets destined for different output links do not interfere with each other, and queueing occurs only at the output ports of the switch [30]. With these assumptions, a connection in such a network can be modeled as traversing a number of queueing servers, with each server modeling the output link of a switch. The network supports variable-size packets.

B. Service Model

We consider the following guaranteed performance service model: before the communication starts, the client needs to specify its traffic characteristics and desired performance requirements. When the network accepts the client's request, it guarantees that the specified performance requirements will be met provided that the client obeys its traffic specification.

In this model, the guaranteed performance service defines a contractual relationship between the communication client and the network [13], [55], [15]: the network promises to fulfill its obligation (guaranteeing the performance for the client's traffic) only if the client honors its own part of the contact (not sending more data than declared). In addition, the network may reject the client's request due to lack of resources or administrative constraints. In its basic form, the contract is signed before data transfer during a connection establishment process and is kept effective throughout the life time of the connection [16]. To increase dynamicity and flexibility, the model can also be extended to allow contract to be modified in the middle of a connection [50].

Recently, a new service model called the predicted service was proposed [7]. There are two important differences between the predicted service and the guaranteed performance service discussed in this paper. First, while the admission control, which decides whether there are enough resources within the network to accept a new connection, is used to support both types of service, the criteria are quite different. In order to decide whether there are enough resources, one has to know the current network load. For predicted service, the current network load is based on measurement; for guaranteed service, it is based on prespecified characterization of existing connections. Since the measured network load may vary, the service commitment by predicted service is less reliable. Secondly, in the predicted service, the delay bound or playback point for a connection is provided by the network and may vary due to the network load fluctuation. It is assumed that applications using the predicted service can adapt to the changing of the playback point and tolerate infrequent service disruptions. In the guaranteed performance service model, delay bound is specified by the application and does not change during the life time of the connection without the explicit request by the client.

B.1 Performance Parameters in Guaranteed Service

The most important clauses in the service contract are the specifications of performance requirements and traffic characteristics. For the performance parameters, the single most important one is the end-to-end delay bound, which is essential for many applications that have stringent real-time requirements. While throughput guarantee is also important, it is provided automatically with the amount specified by the traffic characterization (Section II-B.2). Another important parameter is the end-to-end delay jitter bound. The delay jitter for a packet stream is defined to be the maximum difference between delays experienced by any two packets [13], [56]. For continuous media playback applications, the ideal case would be that the network introduces only *constant* delay, or *zero* delay-jitter. Having a bounded delay-jitter service from the network makes it possible for the destination to calculate the amount of buffer space needed to eliminate the jitter. The smaller the jitter bound, the less amount of buffer space is needed. Since it is more important to provide end-to-end delay and delay-jitter *bounds* than average low delay for guaranteed service class, packets arriving too earlier may not even be desirable in such a environment. In fact, the earlier a packet arrives before its delay bound or playback point, the longer it needs to occupy the buffer. This is one of the most important differences between the performance requirements of the guaranteed-performance service and the best-effort service provided by the traditional computer networks: performance bounds are more important for the guaranteed service while average performance indices are more important for the best-effort service.

A third important parameter is the loss probability. Packet loss can occur due to buffer overflown or delay bound violation. A statistical service [13], [37], [66] allows a non-zero loss probability while while a *deterministic* service guarantees zero loss. With a deterministic service, all packets will meet their performance requirements even in the worst case. With a statistical service, stochastic or probabilistic bounds are provided instead of worst case bounds. Statistical service allows the network to overbook resources beyond the worst-case requirements, thus may increase the overall network utilization by exploiting statistical multiplexing gain.

B.2 Traffic Models in Guaranteed Service

Although there is a general consensus within the research community on the (super) set of parameters to character-

[1] In the literature, the term "switch" is used in the context of ATM networks, while "gateway" or "router" is used more often in an internetworking environment. In this research, we will call switching elements as "switches".

ize performance requirements, there is no agreement on which traffic model or which set of traffic parameters should be adopted. In the traditional queueing theory literature, most models are based on stochastic processes. Among the more popular ones are the Poisson model for data [32], on-off model for voice sources [3] and more sophisticated Markovian models for video sources [43]. A good survey for the probabilistic models for voice and video sources is presented in [46]. In general, these models are either too simple to characterize the important properties of the source or too complex for tractable analysis.

Recently, several new models are proposed to *bound* the traffic rather than characterize the process exactly. Among them are: $(Xmin, Xave, I, Smax)$ [16], (σ, ρ) [8], (r, T) [20], [26], and the D-BIND model [35]. A traffic stream satisfies the $(Xmin, Xave, I, Smax)$ model if the inter-arrival time between any two packets in the stream is more than $Xmin$, the average packet inter-arrival time during any interval of length I is more than $Xave$, and the maximum packet size is less than $Smax$. Alternatively, a traffic stream satisfies the (σ, ρ) model if during any interval of length u, the number of bits in that interval is less than $\sigma + \rho u$. In the (σ, ρ) model, σ and ρ can be viewed as the maximum burst size and the long term bounding rate of the source respectively. Similarly, a traffic stream is said to satisfy (r, T) model if no more than $r \cdot T$ bits are transmitted on any interval of length T. Rather than using one bounding rate, the Deterministic Bounding Interval-Dependent (D-BIND) model uses a family of rate-interval pairs where the rate is a bounding rate over the corresponding interval length. The model captures the intuitive property that over longer interval lengths, a source may be bounded by a rate lower than its peak rate and closer to its long-term average rate.

In each of the above models, the exact traffic pattern for a connection is unknown, the only requirement is that the volume of the traffic be *bounded* in certain ways. Such bounding characterizations are both general and practical. They can characterize a wide variety of bursty sources. In addition, it is sufficient for resource management algorithms to allocate resources by knowing just the *bounds* on the traffic volume.

A bounding characterization can either be *deterministic* or *stochastic*. A bounding deterministic traffic characterization defines a deterministic traffic constraint function. A monotonic increasing function $b_j(\cdot)$ is called a deterministic traffic constraint function of connection j if during *any* interval of length u, the number of bits arriving on j during the interval is no greater than $b_j(u)$. More formally, let $A_j(t_1, t_2)$ be the total number of bits arrived on connection j in the interval of (t_1, t_2), $b_j(\cdot)$ is a traffic constraint function of connection j if $A_j(t, t+u) \leq b_j(u), \ \forall t, u > 0$. Notice that $b_j(\cdot)$ is a time invariant deterministic bound since it constrains the traffic stream over every interval of length u. For a given traffic stream, there are an infinite number of valid traffic constraint functions, out of which, a deterministic traffic model defines a parameterized family. All of the above traffic models have corresponding traffic constraint functions. For example, the traffic constraint function of (σ, ρ) model is $\sigma + \rho u$. The traffic constraint can also be stochastic. In [37], a family of stochastic random variables are used to characterize the source. Connection j is said to satisfy a characterization of $\{(\mathbf{R}_{t_1,j}, t_1), (\mathbf{R}_{t_2,j}, t_2), (\mathbf{R}_{t_3,j}, t_3)...\}$, where $\mathbf{R}_{t_i,j}$ are random variables and $t_1 < t_2 < \cdots$ are time intervals, if $\mathbf{R}_{t_i,j}$ is *stochastically larger* than the number of bits generated over any interval of length t_i by source j. This model is extended in [66] by explicitly considering the interval-dependent property of the source: over longer interval lengths, a source may be bounded by a rate lower than its peak-rate and closer to its long-term average. The resulted model is called Stochastic Bounding Interval Dependent or S-BIND model. Another related traffic model is the Exponentially Bounded Burstiness (E.B.B.) process proposed in [60], [59]. A source is said to be E.B.B. with parameters $(\rho, \mathcal{A}, \alpha)$ if $Pr\{A[s, s+t] \geq \rho t + \sigma\} \leq \mathcal{A} e^{-\alpha\sigma} \ \ \forall \sigma \geq 0$ and $s, t > 0$ where random variable $A[t_1, t_2]$ denotes the total number of bits generated by a source in the interval $[t_1, t_2]$.

In this paper, we assume that a communication client uses a deterministic bounding traffic model to specify its traffic if it requests a deterministic service and use a stochastic bounding traffic model to specify its traffic if it requests a statistical service.

C. Traffic Management Algorithms

In packet-switching networks, there is the possibility that the aggregate rate of the input traffic to the network (or a portion of the network) temporarily exceeds the capacity of the network, in which cases packets may experience long delays or get dropped by the network. This is called congestion. Although networks are expected to become even faster, the problem of congestion is not likely to go away [25]. Various congestion control or traffic management algorithms have been proposed in the literature. These solutions can be classified into two classes: reactive, or feedback control schemes [24], [51], and proactive, or resource reservation algorithms [16], [39], [67].

Reactive approaches detect and react dynamically to congestion inside the network by relying on the feedback information from the network, while proactive approaches eliminate the possibility of congestion by reserving network resources for each connection. From the point of view of control time scale, reactive approaches operate in a time scale of several *round-trip times* since the length of the interval between the time when the congestion is detected and the time when the congestion signal is passed back to the source is on the order of one round-trip time. Proactive approaches, on the other hand, operate on at least two timescales: connection level and packet level. At the connection level, when a new connection request comes in, a set of connection admission control conditions are tested at each switch. The new connection is accepted only if there are enough resources to satisfy the requirements of both the new connection and existing connections. At the packet level, the packet service discipline at each switch selects which packet to transmit next by discriminating

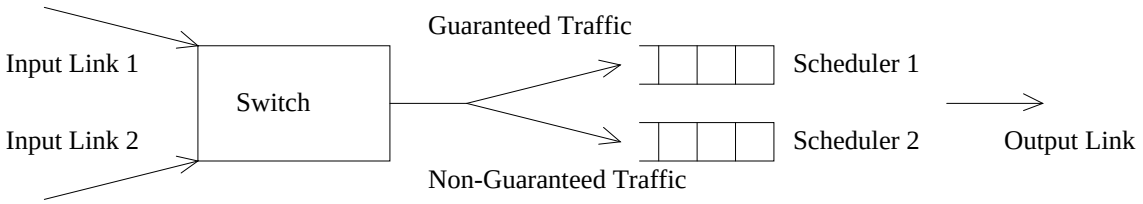


Fig. 1. Servicing both Guaranteed-Service and Non-Guaranteed Traffic

packets based on their performance requirements. Usually, different service disciplines need different admission control algorithms. A complete solution needs to specify both the service discipline and the associated connection admission control conditions.

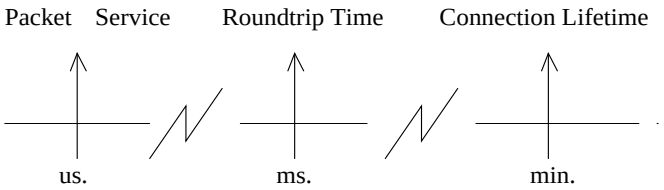


Fig. 2. Control Timescales For Traffic Management Algorithms

The three timescales used by traffic management algorithms are illustrated in Figure 2. While a reactive approach is suitable for supporting best-effort service, a proactive traffic management architecture is better for guaranteed performance service. The two approaches can co-exist in an integrated services network.

D. Service Disciplines

As can be seen from Figure 2, packet service disciplines operate at the smallest time scale, or with the highest frequency. Together with connection admission control algorithms, they provide the two most important components in a proactive traffic management architecture. While connection admission control algorithms *reserve* resources during connection establishment time, packet service disciplines *allocate* resources according to the reservation during data transfer. Three types of resources are being allocated by service disciplines [12]: bandwidth (*which* packets get *transmitted*), promptness (*when* do those packets get *transmitted*) and buffer space (*which* packets are *discarded*), which, in turn, affects three performance parameters: throughput, delay, and loss rate.

Even in reactive or feedback-based traffic management architecture, appropriate scheduling at packet switches will make end-to-end control more effective [12], [31]. In the rest of the paper, we consider architecture shown in Figure 1 for service disciplines in integrated services networks. There are separate queues and service policies for guaranteed service and other packets. Best-effort packets are transmitted only when no packets from the guaranteed service queue are available for transmission. It should be noticed that the two queues in Figure 1 are logical ones. Depending on the service discipline, each logical queue can corresponds to multiple physical queues. For example, if a Static Priority discipline with n priority levels is used for guaranteed traffic and a round robin discipline with m classes is used for non-guaranteed traffic, the number of physical queues is $n + m$ at each output port. In this paper, we will focus on the service disciplines for guaranteed traffic.

Although it is possible to build a guaranteed performance service on top of a vast class of service disciplines [14], we would like a service discipline to be efficient, protective, flexible, and simple.

Efficiency. To guarantee certain performance requirements, we need a connection admission control policy to limit the guaranteed service traffic load in the network, or limit the number of guaranteed service connections that can be accepted. A service discipline is more efficient than another one if it can meet the same *end-to-end* performance guarantees under a heavier load of guaranteed service traffic. An efficient service discipline will result in a higher utilization of the network.

Protection. Guaranteed service requires that the network protects well-behaving guaranteed service clients from three sources of variability: ill-behaving users, network load fluctuation and best-effort traffic. It has been observed in operational networks that ill-behaving users and malfunctioning equipments may send packets to a switch at a higher rate than declared. Also, network load fluctuations may cause a higher instantaneous arrival rate from a connection at some switch, even though the connection satisfies the traffic constraint at the entrance to the network. Another variability is the best-effort traffic. Although the guaranteed service traffic load is limited by connection admission control, best-effort packets are not constrained. It is essential that the service discipline should meet the performance requirements of packets from well-behaving guaranteed service clients even in the presence of ill-behaving users, network load fluctuation and unconstrained best-effort traffic.

Flexibility. The guaranteed performance service needs to support applications with diverse traffic characteristics and performance requirements. Scientific visualization and medical imaging will have very different characteristics from video. Even for video, conferencing applications, movie applications and HDTV require different qualities of service. Other factors, such as different coding algorithms and different resolutions, also contribute to the diversity of video requirements. Because of the vast diversity of traffic characteristics and performance requirements of existing applications, as well as the uncertainty about future appli-

cations, the service discipline should be flexible to allocate different delay, bandwidth and loss rate quantities to different guaranteed service connections.

Simplicity. The service discipline should be both conceptually simple to allow tractable analysis and mechanically simple to allow high speed implementation.

III. Work-Conserving Service Disciplines

A service discipline can be classified as either work-conserving or non-work-conserving. With a work-conserving discipline, a server is never idle when there is a packet to send. With 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 this paper, whether a service discipline is work-conserving affects the end-to-end delay analysis, buffer space requirements, and delay-jitter characteristics.

In this section, we will study the work-conserving disciplines: Delay Earliest-Due-Date (Delay-EDD) [16], [29], [69], Virtual Clock [67], Fair Queueing (FQ) [12] and its weighted version (WFQ) also called Packetized Generalized Processor Sharing (PGPS) [48], Self-Clocked Fair Queueing (SCFQ) [22], and Worst-case Fair Weighted Fair Queueing (WF^2Q) [2]. We first describe each of these disciplines, then present a framework to show the similarities and differences among them. Finally, we examine the end-to-end delay characteristics and buffer space requirements for each of them. Non-work-conserving disciplines will be discussed in Section IV.

A. Virtual Clock

The Virtual Clock [67] discipline aims to emulate the Time Division Multiplexing (TDM) system. 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. Packets are transmitted in the increasing order of virtual transmission times.

Figure 3 gives a simple example to illustrate how Virtual Clock works. In the example, there are three connections sharing the same output link. All three connections specify their traffic characteristics and reserve resources accordingly. Connections 1 has an average packet inter-arrival time of 2 time units, connection 2 and 3 have an average packet inter-arrival time of 5 time units. For simplicity, assume packets from all the connections have the same size, and the transmission of one packet takes one time unit. Hence, each of connections 2 and 3 reserve 20% of the link bandwidth, while connection 1 reserves 50% of the link bandwidth. The arrival pattern of the three connections are shown in the first three timelines. As can be seen, connections 2 and 3 send packets at higher rates than reserved, while connection 1 sends packet according to the specified traffic pattern. The fourth timelines show the service order of packets when the service discipline is FCFS. In this case, even though connection 1 reserves more resources, the misbehaviors of connections 2 and 3 affect its performance.

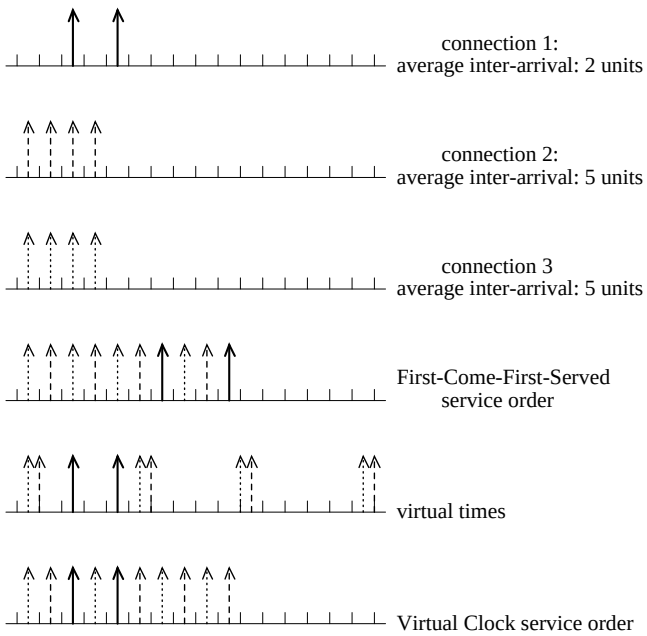


Fig. 3. Comparison of Virtual Clock and FCFS

The Virtual Clock algorithm assigns each packet a virtual transmission time based on the arrival pattern and the reservation of the connection to which the packet belongs. The fifth timeline shows the virtual transmission time assignment. The transmissions of packets are then ordered by the virtual transmission times. The service order of packets under the Virtual Clock discipline is shown in the sixth timeline. Notice that although connections 2 and 3 are sending packets at higher rates, the Virtual Clock algorithm ensures that each well-behaving connection, in this case connection 1, gets good performance.

B. WFQ and WF^2Q

WFQ and WF^2Q are two packet policies that try to approximate the same Fluid Fair Queueing (FFQ) or Generalized Processor Sharing (GPS) policy. FFQ is a general form of the head-of-line processor sharing service discipline (HOL-PS) [33]. With HOL-PS, there is a separate FIFO queue for each connection sharing the same link. During any time interval when there are exactly N non-empty queues, the server serves the N packets at the head of the queues simultaneously, each at a rate of one N^{th} of the link speed. While a HOL-PS server serves all non-empty queues at the same rate, FFQ allows different connections to have different service shares. A FFQ is characterized by N positive real numbers, $\phi_1, \phi_2, \cdots, \phi_N$, each corresponding to one queue. At any time τ, the service rate for a non-empty queue i is exactly $\frac{\phi_i}{\sum_{j \in B(\tau)}} C$ where $B(\tau)$ the set of non-empty queues and C is the link speed. Therefore, FFQ serves the non-empty queues in proportion to their service shares. FFQ is impractical as it assumes that the server can serve all connections with non-empty queues simultaneously and that the traffic is infinitely divisible. In a more realistic packet system, only one connection can receive service at a time and an entire packet must be served

before another packet can be served.

There are different ways of approximating FFQ service in a packet system. Among them, the most well-known one is the Weighted Fair Queueing discipline (WFQ) [12], also known as Packet Generalized Processor Sharing or PGPS [47]. In WFQ, when the server is ready to transmit the next packet at time τ, it picks, among all the packets queued in the system at τ, the first packet that would complete service in the corresponding FFQ system if no additional packets were to arrive after time τ.

While WFQ uses only finish times of packets in the FFQ system, WF^{2}Q uses both start times and finish times of packets in the FFQ system to achieve a more accurate emulation. In WF^{2}Q, when the next packet is chosen for service at time τ, rather than selecting it from among all the packets at the server as in WFQ, the server only considers the set of packets that *have started (and possibly finished) receiving service in the corresponding FFQ system at time* τ, and selects the packet among them that would complete service first in the corresponding FFQ system.

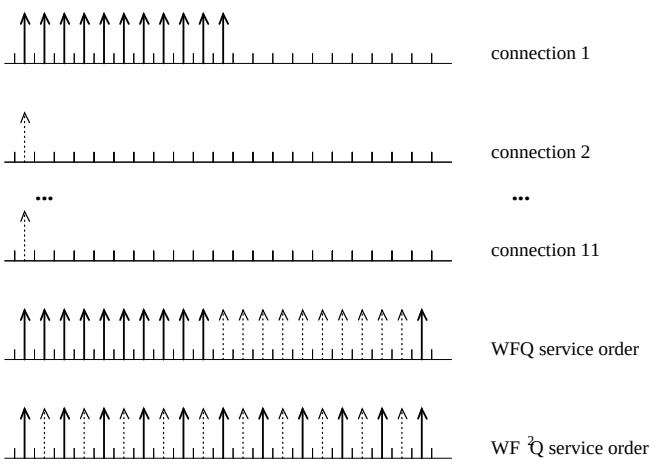


Fig. 4. WFQ and WF^{2}Q

The following example, shown in Figure 4, illustrates the difference between WFQ and WF^{2}Q. For simplicity, assume all packets have the same size of 1 and the link speed is 1. Also, let the guaranteed rate for connection 1 be 0.5, and the guaranteed rate for each of the other 10 connections be 0.05. In the example, connection 1 sends 11 back-to-back packets starting at time 0 while each of all the other 10 connections sends only one packet at time 0. If the server is FFQ, it will take 2 time units to service each of the first 10 packets on connection 1, one time unit to service the 11^{th} packet, and 20 time units to service the first packet from another connection. Denote the k^{th} packet on connection j to be p_j^k, then in the FFQ system, the starting and finishing times are $2(k-1)$ and $2k$ respectively for $p_1^k, k = 1 \ldots 10$, 20 and 21 respectively for p_1^{11}, and 0 and 20 respectively for $p_j^1, j = 2 \cdots 11$.

For the same arrival pattern, the service orders under the packet WFQ and WF^{2}Q systems are different. Under WFQ, since the first 10 packets on connection 1 ($p_1^k, k = 1 \ldots 10$) all have FFQ finish times smaller than packets on other connections [2], the server will service 10 packets on connection 1 back to back before service packets from other connections.

Under WF^{2}Q, At time 0, all packets at the head of each connection's queue, p_i^1, $i = 1, \cdots, 11$, have started service in the FFQ system. Among them, p_1^1 has the smallest finish time in FFQ, so it will be served first in WF^{2}Q. At time 1, there are still 11 packets at the head of the queues: p_1^2 and p_i^1, $i = 2, \cdots, 11$. Although p_1^2 has the smallest finish time, it will not start service in FFQ until time 2, therefore, it won't be eligible for transmission at time 1. The other 10 packets have all started service at time 0 at the FFQ system, thus are eligible. Since they all finish at the same time in the FFQ system, the tie-breaking rule of giving highest priority to the connection with the smallest number will yield p_2^1 as the next packet for service. At time 3, p_1^2 becomes eligible and has the smallest finish time among all packets, thus it will start service next. The rest of the sample path for the WF^{2}Q system is shown in Figure 4.

There are two noteworthy points. First, at any given time τ, the accumulated service provided for each connection (the total amount of bits transmitted) by either packet system *never* falls behind the fluid FFQ system by more than one packet size. Such a relationship with FFQ and the fact that end-to-end delay bounds can be provided in FFQ are the basis for establishing end-to-end delay bounds in WFQ and WF^{2}Q. Also, since in the worst case both WFQ and WF^{2}Q can fall behind FFQ by the same amount of service, they provide the same end-to-end delay bounds. However, as shown in the example, the service order under WFQ and WF^{2}Q can be quite different. Even though WFQ cannot fall much behind FFQ in terms of service, it can be quite far *ahead of* the FFQ system. In the example, by the time 10, 10 packets on connection 1 have been served in the WFQ system, while only 5 packets have been served in the FFQ system. The discrepancy between the service provided by WFQ and FFQ can be even larger when there are more connections in the system. In contrast, WF^{2}Q does not have such a problem. In the above example, by the time 10, WF^{2}Q will have served 5 packets, exactly the same as FFQ. In fact, it can be shown that the difference between the services provided by WF^{2}Q and FFQ is always less than one packet size. Therefore, WF^{2}Q is the most accurate packet discipline that approximates the fluid FFQ discipline.

Even though the difference between WFQ and WF^{2}Q does not affect the end-to-end delay bounds they provide, it is shown in [2] that such a difference may have important implications if they are used to provide best-effort services.

C. Self-Clocked Fair Queueing

Both WFQ and WF^{2}Q need to emulate a reference FFQ server. However, maintaining the reference FFQ server is computationally expensive. One simplier packet approx-

[2] The FFQ finish time of packet p_1^{10} is 20, the same as that of packets $p_j^1, j = 2 \cdots 11$. If we adopt the following tie-breaking policy in which the packet from the connection with the smallest connection number has a higher priority, packet p_1^{10} will be served before $p_j^1, j = 2 \cdots 11$.

imation algorithm of FFQ is Self-Clocked Fair Queueing (SCFQ) [22] also known informally as "Chuck's Approximation" [11]. The exact algorithm of SCFQ and the examples illustrating the difference between WFQ and SCFQ will be given in Section III-E.

D. Delay-Earliest-Due-Date

Delay-Earliest-Due-Date algorithm or Delay-EDD [16] is an extension to the classic Earliest-Due-Date-First (EDD or EDF) scheduling [41]. In the original EDD, each packet from a periodic traffic stream is assigned a deadline and the packets are sent in order of increasing deadlines. The deadline of a packet is the sum of its arrival time and the period of traffic stream. In Delay-EDD, the server negotiates a service contract with each source. The contract states that if a source obeys its promised traffic specification, such as 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 k^{th} packet from the client will get a deadline of $0.2k + 1$.

E. Framework for Work-Conserving Disciplines

Virtual Clock, Delay-EDD, WFQ, WF^{2}Q, and SCFQ all use a similar sorted priority queue mechanism. In such a mechanism, there is a state variable associated with each connection to monitor and enforce its traffic. Upon arrival of each packet from that connection, the variable is updated according to (a) the reservation made for the connection during the connection establishment time and, (b) the traffic arrival history of this connection and/or other connections during the data transfer. The packet is then stamped with the value of the state variable for the connection to which it belongs. The stamped value is used as a priority index. Packets are served in the order of increasing priority index values. This is shown in Figure 5. WF^{2}Q also needs additional mechanism to mark whether packets are eligible for transmission. As will be discussed in Section IV, this can be implemented with a calendar queue.

In Virtual Clock, the state variable is called auxiliary Virtual Clock ($auxVC$); in WFQ, WF^{2}Q, and SCFQ, it is called the Virtual Finish Time (F); in Delay-EDD, it is called Expected Deadline (ExD). In all three cases, $auxVC$, F and ExD are used as priority indices of packets. The computations of $auxVC$, F and ExD are based on the formula shown in Table I. In the table, the subscripts i, j, and k denotes server number, connection number, and packet number respectively. In Delay-EDD, $Xmin_j$ is the minimum packet inter-arrival time for connection j, $d_{i,j}$ is the local delay bound assigned to connection j at server i at connection establishment time. In Virtual Clock, $Vtick_j$ is the average packet inter-arrival time for connection j. In WFQ and WF^{2}Q, $V(t)$ is the system *virtual time* at time t, where the virtual time, defined below, is a measure of the system progress. L_j^k is the packet length measured in number of bits, and $a_{i,j}^k$ is arrival time of the k^{th} packet on connection j at switch i.

Virtual Clock	$auxVC_{i,j}^k \leftarrow max\{a_{i,j}^k, auxVC_{i,j}^k\} + Vtick_{i,j}$
WFQ & WF^{2}Q	$F_{i,j}^k \leftarrow max\{V_i(a_{i,j}^k), F_{i,j}^{k-1}\} + \frac{L_j^k}{\phi_{i,j}}$
SCFQ	$F_{i,j}^k \leftarrow max\{\hat{V}_i(a_{i,j}^k), F_{i,j}^{k-1}\} + \frac{L_j^k}{\phi_{i,j}}$
Delay-EDD	$ExD_{i,j}^k \leftarrow max\{a_{i,j}^k + d_{i,j}, ExD_{i,j}^k + Xmin_j\}$

TABLE I
COMPARISON OF WORK-CONSERVING DISCIPLINES

As shown in Table I, the priority index updating algorithms are very similar. However, there are also two important differences. The first is whether the calculation is based on just the rate parameter or both the delay and rate parameters. The second is whether the updating is based on system-load *independent* parameters or system-load *dependent* parameters.

Notice that in Delay-EDD, two parameters are used to update the priority index: $Xmin_j$, which is the minimum packet inter-arrival time for connection j, and $d_{i,j}$, which is the local delay bound for connection j at switch switch i. In other disciplines, only one rate parameter is used ($Vtick_j$ or $\phi_{i,j}$). Although delay bounds can be provided for all these disciplines, having only one rate parameter introduces the problem of coupling between the allocation of delay bound and bandwidth. For example, in Rate-Proportional Processor Sharing (RPPS), which is a special case of WFQ or PGPS and the ϕ's are allocated proportional to the bandwidth required by connections, if the traffic is constrained by (σ, ρ) [3] characterization, the end-to-end delay bound of the connection will be $\frac{\sigma+(n-1)L_{max}}{\rho} + \sum_{i=1}^{n} \frac{L_{max}}{C_i}$, where n is the number of hops traversed by the connection, and C_i is the link speed of the i^{th} server. Notice that the delay bound is inversely proportional to the allocated long term average rate. Thus, in order for a connection to get a low delay bound, a high bandwidth channel needs to be allocated. This will result in a waste of resources when the low delay connection also has low throughput. Delay-EDD, on the other hand, does not have such a restriction [16], [40].

The second difference between these disciplines is whether the updating of the state variables depends on system load. In Virtual Clock and Delay-EDD, the updating depends only on per connection parameters ($Vtick$ for Virtual Clock, $Xmin$ and d for Delay-EDD) but not on system load. In WFQ, WF^{2}Q, and SCFQ, the updating is based on a notion of virtual time. The evolution of virtual time measures the progress of the system and depends on system load. For WFQ and WF^{2}Q, the virtual time function $V(\cdot)$ during any busy period $[t_1, t_2]$ is defined as follows:

$$V(t_1) = 0 \quad (1)$$

[3] As mentioned in Section 1, σ is the maximum burst size, ρ is the long term average rate.

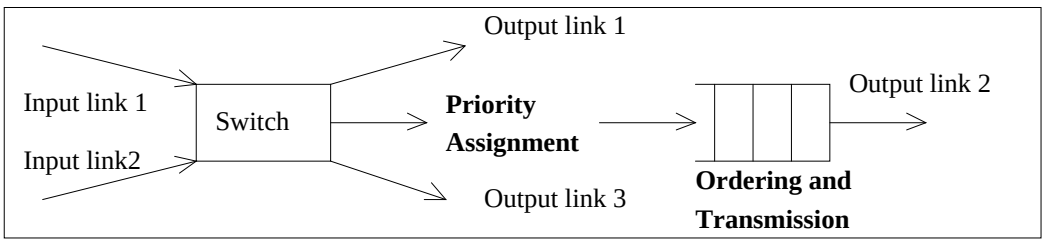


Fig. 5. Sorted priority mechanism

$$\frac{\partial V(\tau)}{\partial \tau} = \frac{1}{\sum_{j \in B_{FFQ}(\tau)} \phi_j} \quad \forall t_1 \le \tau \le t_2 \qquad (2)$$

where $B_{FFQ}(\tau)$ is the set of backlogged connections[4] at time τ under the reference FFQ system. In FFQ, if connection j is backlogged at time τ, it receives a service rate of $\frac{\partial V(\tau)}{\partial \tau}\phi_j C$, where C is the link speed. Therefore, V can be interpreted as increasing proportionally to the marginal rate at which backlogged connections receive service in FFQ.

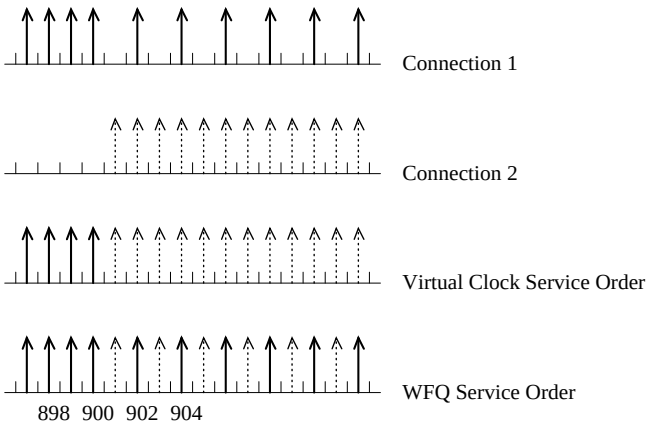


Fig. 6. WFQ and Virtual Clock

The following example, given in [47] and illustrated in Figure 6, shows the difference between WFQ and Virtual Clock. Suppose that there are two connections, both with a specified average rate of one packet every two seconds. All packets are fixed size and require exactly one second to service. Starting at time zero, 1000 packets from connection 1 arrive at a rate of 1 packet/second. No packets from connection 2 arrive at the interval [0,900). Starting at time 900, 450 connection 2 packets arrive at a rate of 1 packet/second. Since the first 900 packets from connection 1 are served in the interval [0,900), there are no packets in queue from either connection at time 900^-. If Virtual Clock algorithm is used, at time 900, the connection 1 $auxVC$ reads 1800 and the connection 2 clock reads 900 (as can be seen in Table I, the $auxVC$ value is at least the real-time value). When connection 2 packets arrive, they will be stamped 900, 902, $\cdots$, 1798, while the connection 1 packets that arrive after time 900 will be stamped 1800, 1802, $\cdots$, 1898. Thus, *all* of the connection 2 packets will be served before any of the connection 1 packets are served. On the other hand, if WFQ discipline is used, the number of active connections is 1 before time 900 and 2 after time 900. Since both connection 1 and connection reserve half of the link bandwidth, after time 900, the WFQ server will service packets from both connections interleavingly.

The different behaviors of Virtual Clock and PGPS are due to the fact that Virtual Clock is defined with reference to the *static* TDM system and the calculation of the virtual transmission time is *independent* of the behaviors of other connections. The delay of a packet depends on the entire arrival history of the connection, which is summarized in the state variable $auxVC$. Once a connection is mishaved (sending more packets than specified), it may be punished by Virtual Clock, *regardless whether such misbehavior affects the performance of other connections.* WFQ, on the other hand, is defined with reference to another *dynamic* queueing system FFQ. The virtual time of the system *depends* on how many other connections are active in the system.

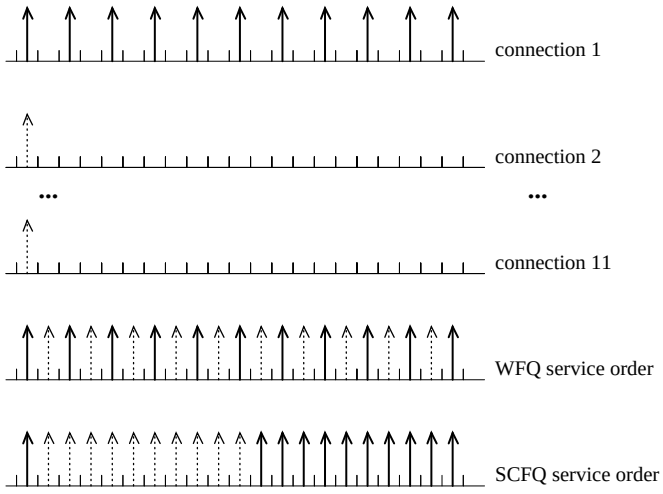


Fig. 7. SCFQ and WFQ

The dependency on virtual time also introduces extra complexities for WFQ and WF^{2}Q since the system needs to emulate FFQ and keep track of the number of active connections at any moment in FFQ. To reduce the complexity of computing virtual times, SCFQ introduces an approximation algorithm. The algorithm is based on the observation that the system's virtual time at any moment t may be estimated from the virtual service time of the packet currently being serviced. Formally, the approximate virtual

[4]A connection is said to be backlogged at time τ if it has packets queued at time τ.

time function $\hat{V}(t)$ is defined to be F^p where $s^p < t \leq f^p$, s^p and f^p denote the times packet p starts and finishes service in the SCFQ system.

While the calculation of virtual time is simplier in SCFQ, the inaccuracy incurred can make SCFQ perform much worse than WFQ. Consider the example illustrated in Figure 7. Again, assume all packets have the same size of 1, the link speed is 1, the guaranteed rate for connection 1 be 0.5, and the guarantee rate for each of the other 10 connections be 0.05. Under FFQ, the finish times will be $2k$ for packets $p_1^k, k = 1 \ldots 10$, 20 for packets $p_j^1, j = 2 \cdots 11$, and 21 for p_1^{11}. Transmitting packets in order of finish times under FFQ, WFQ will produce the service order as shown on the fourth timeline in Figure 7. If SCFQ is used, at time 0, same as in WFQ, it is p_1^1 that has the smallest virtual finish time, therefore, it receives service first. At time 1, all packets $p_i^1, i = 2, \ldots, 11$, have virtual finish time of $F_i^1 = 20$. With the tie-breaking rule, the first packet on connection 2, p_2^1, is served. Since SCFQ uses the finish time of the packet in service as the the current virtual time value, we have $\hat{V}(1) = \hat{V}(2) = F_2^1 = 20$. As a result when p_1^2 arrives at time 2, its virtual finish time is set to be: $F_1^2 = \max(F_1^1, \hat{V}(2)) + \frac{L}{r_1} = \max(2, 20) + 2 = 22$. Among all the packets ready to be served, p_1^2 has the largest finish number. Therefore, p_1^2 won't start service until all other 10 $p_i^1, i = 2, \ldots, 11$, packets finish services, i.e. it won't depart until time 12. In this example, even though connection 1 sends packet according to the specified average rate, its packets still get significantly delayed.

F. Traffic Characterization Inside the Network

As discussed in Section I, we are interested in providing end-to-end delay bounds on a per connection basis in a networking environment. One solution is to obtain worst-case local delay bounds at each switch independently and use the sum of these local delay bounds as the end-to-end delay bound [16]. Alternatively, smaller end-to-end delay bounds can be obtained by taking into account the dependencies in the sucessive switches that a connection traverses [47], [10], [68], [19], [17], [23]. For the first type of solution, in order to derive local delay bound, traffic needs to be characterized on a per connection basis at each switch inside the network. For the second type of solution, while the end-to-end delay bound may be derived for Virtual Clock, WFQ, SCFQ based only on the source traffic characterization [47], [10], [17], [23], as will be shown in Section III-G, the delay bound couples with bandwidth allocation. In [47], such a source allocation strategy is called rate-proportional allocation. It has been shown more general resource allocation strategies that decouples throughtput and delay bounds can result in higher utilization of the network. In general, for both types of solutions, the traffic needs to be characterized on a per connection basis at each switch inside the network.

The difficulty arises in a networking environment, where even if a connection's traffic can be characterized at the entrance to the network, traffic pattern may be distorted inside the network, thus make the source characterization not applicable at the servers traversed by the connection.

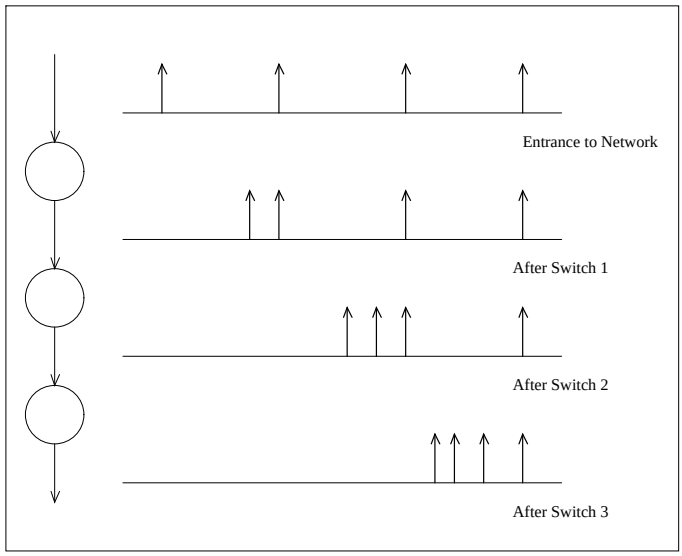


Fig. 8. Traffic pattern distortions due to load fluctuations

This is shown in the following example illustrated by Figure 8. In the example, four packets from one connection are sent with a certain inter-packet spacing from the source into a network where links have constant delay. At the first server, the first packet is delayed by a certain amount of time (less than the local delay bound) due to instantaneous cross traffic load, but the other three packets pass through instantly. Because the first packet was delayed longer than the second packet, the spacing between first and second packets becomes smaller when they arrive at the second server. At the second server, the first and the second packet are delayed some more time, but packets 3 and 4 pass through instantly. At the third server, the first three packets are delayed but packet 4 passes through with no delay. The figure shows traffic patterns at the entrance to each of the servers. Two things can be observed: (a) the traffic pattern of a connection can be distorted due to network load fluctuations, (b) the distortion may make the traffic burstier and cause instantaneously higher rates. In the worst case, the distortion can be accumulated, and downstream servers potentially face burstier traffic than upstream servers. Therefore, the source traffic characterization may not be applicable inside the network.

There are three solutions to address this problem of traffic pattern distortion:

1. controlling the traffic distortion within the network,
2. accounting for the distortion during scheduling,
3. characterizing the traffic distortion throughout the network.

To control traffic distortions within the network, some packets need to be held even when a server has the extra capacity. This requires non-work-conserving disciplines, which we will discuss in more detail in Section IV.

The second solution accounts for traffic distortions during scheduling. Instead of scheduling packets according to their actual arrival times, the server assigns each packet a logical arrival time based on its traffic characterization and previous arrival history, and schedules packets based on

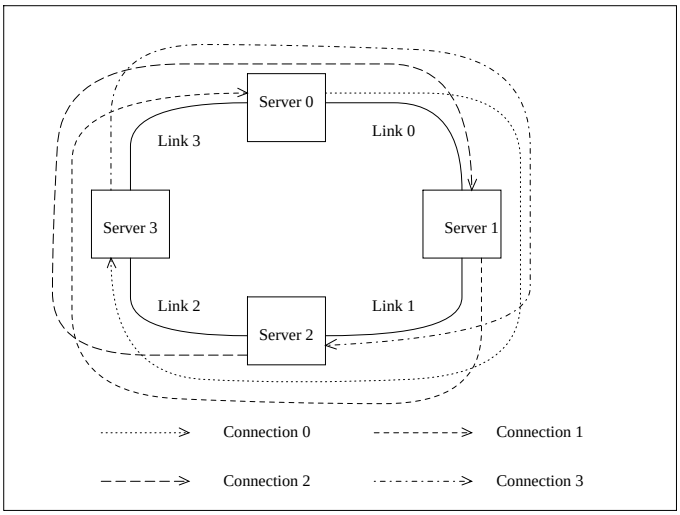


Fig. 9. Example of a Feedback Network

their logical arrival times. Delay-EDD and Virtual Clock use such an approach. For example, in Delay-EDD, the deadline of a packet is the sum of the local delay bound (d) and the expected arrival time (ExA) of the packet. The service discipline and the admission control policy ensure that the packet is guaranteed to leave before the deadline, or at most d time units after the expected arrival time of the packet. It is possible that a packet is delayed longer in a server than its local delay bound. However, this can only happen if the packet's expected arrival time is larger than its actual arrival time, which means that the packet is ahead of schedule in previous servers. It can be shown that the amount of the time the packet is queued at the server more than its delay bound is always less than the amount of time the packet is ahead of schedule at previous servers.

Accounting for traffic distorting during scheduling works only if the server has a concept of expected arrival time. A more general solution is to characterize the traffic inside the network. The problem can be formulated as the following: given the traffic characterization of all the connections at the entrance to the network and all the service disciplines at the switches, can the traffic be characterized on a per connection basis on all the links inside the network? Several solutions have been proposed to address this problem with different traffic models and service disciplines [1], [8], [37], [47]. They all employ a similar technique that consists of two steps. In the first step, a single node analysis technique is developed to characterize the output traffic of a server given the characterizations of all its input traffic. In the second step, starting from the characterizations of all the source traffic, an iterative process push the traffic characterizations from the links at the edge of the network to those inside the network. There are several limitations associated with such an approach.

First, characterizing the traffic inside the network is difficult and may not always be possible. In [9], [37], [49], it is shown that this is equivalent to solving a set of multivariable equations. In a feedback network, where traffic from different connections forms traffic loops, the resulting set of equations may be unsolvable. To illustrate this, consider the following example discussed in [9], [47].

In the 4-nodes network shown in Figure 9, there are four 3-hop connections and the aggregate traffic of the four connections forms a loop. In order to characterize the traffic on link 1, the characterization of the input traffic to server 1 has to be known. Assuming links only introduce fixed delay, the input traffic to server 1 is identical to the output traffic of server 0, or the traffic on link 0. There are three traffic streams on link 0, which are from connections 0, 2, and 3. While connection 0 traffic on link 0 is the same as its source traffic, connection 2 and connection 3 traffic on link 0 needs to be characterized. The characterizations of connection 2 and 3 traffic depend on their characterizations on link 3, which in turn depend on their characterizations on link 2. This dependency finally comes back to traffic characterizations on link 0. Because of this inter-dependency of traffic, characterizing all the traffic inside the network is equivalent to solving a set of multivariable equations, where each variable corresponds to one parameter in the traffic characterization of one connection on one link. The equations are solvable only under certain conditions. In this particular example, it is shown in [9] that if each server has a policy such that the traffic originating from the server has a lower priority than the through traffic, the condition for solving the equations is that the aggregate throughput of all connections must be less than 75% of the link bandwidth on each of the four links. This condition is not merely a technical restriction, the network *can* actually be unstable, i.e., have unbounded queue lengths, when the condition is violated [47]. How to derive the stability condition in a general networking environment is still a open problem. A distributed algorithm is even more difficult. One notable exception to such a restriction is the case when the service discipline used is a special class of PGPS called Rate Proportional Processor Sharing (RPPS) [47]. With RPPS, the stability condition can be derived. We will discuss the exact formula of the delay bound in Section III-G.

The second limitation of characterizing traffic inside the network is that it only applies to networks with *constant* delay links. Constant delay links have the desirable property that the traffic pattern at the receiving end of the link is the same as that at the transmitting end. This property is important for these solutions because central to the analysis is the technique of characterizing the output traffic from a server and using it as the characterization of the input traffic to the next-hop server. However, in an internetworking environment, where the link between two switches may be a subnetwork such as an ATM network or a FDDI network [14], load fluctuations within subnetworks may also introduce traffic pattern distortions. Though it is possible to bound delay over these subnetworks, the delays for different packets will be *variable*. Thus, these solutions need to be extended in order to be applicable in an internetworking environment.

	traffic constraint	end-to-end delay bound	end-to-end delay-jitter bound	buffer space at h^{th} switch
D-EDD	$b_j(\cdot)$	$\sum_{i=1}^{n} d_{i,j}$	$\sum_{i=1}^{n} d_{i,j}$	$b_j(\sum_{i=1}^{h} d_{i,j})$
FFQ	(σ_j, ρ_j)	$\frac{\sigma_j}{r_j}$	$\frac{\sigma_j}{r_j}$	σ_j
VC	(σ_j, ρ_j)	$\frac{\sigma_j+nL_{max}}{r_j} + \sum_{i=1}^{n} \frac{L_{max}}{C_i}$	$\frac{\sigma_j+nL_{max}}{r_j}$	$\sigma_j + hL_{max}$
WFQ & WF^{2}Q	(σ_j, ρ_j)	$\frac{\sigma_j+nL_{max}}{r_j} + \sum_{i=1}^{n} \frac{L_{max}}{C_i}$	$\frac{\sigma_j+nL_{max}}{r_j}$	$\sigma_j + hL_{max}$
SCFQ	(σ_j, ρ_j)	$\frac{\sigma_j+nL_{max}}{r_j} + \sum_{i=1}^{n} K_i \frac{L_{max}}{C_i}$	$\frac{\sigma_j+nL_{max}}{r_j} + \sum_{i=1}^{n} (K_i - 1) \frac{L_{max}}{C_i}$	$\sigma_j + hL_{max}$

TABLE II

END-TO-END DELAY BOUND, DELAY-JITTER BOUND, AND BUFFER SPACE REQUIREMENTS

Finally, in networks with work-conserving service disciplines, even in situations when traffic inside the network can be characterized, the characterization usually represents a burstier traffic inside the network than that at the entrance. This is independent of the traffic model being used. In [8], it is shown that if the traffic of connection j is characterized by (σ_j, ρ_j) at the entrance to the network, its characterization will be $(\sigma_j + \sum_{h=1}^{i-1} \rho_j d_{h,j}, \rho_j)$ at the entrance to the i^{th} server along the path, where $d_{h,j}$ is the local delay bound for the connection at the h^{th} server. Compared to the characterization of the source traffic, the maximum burst size increases by $\sum_{h=1}^{i-1} \rho_j d_{h,j}$. This increase of burst size grows monotonically along the path.

In [37], a family of stochastic random variables are used to characterize the source. Connection j is said to satisfy a characterization of $\{(\mathbf{R}_{t_1,j}, t_1), (\mathbf{R}_{t_2,j}, t_2), (\mathbf{R}_{t_3,j}, t_3)...\}$, where $\mathbf{R}_{t_i,j}$ are random variables and $t_1 < t_2 < \cdots$ are time intervals, if $\mathbf{R}_{t_i,j}$ is *stochastically larger* than the number of packets generated over any interval of length t_i by source j. If the traffic connection j is characterized by $\{(\mathbf{R}_{t_1,j}, t_1), (\mathbf{R}_{t_2,j}, t_2), ...\}$ at the entrance to the network, its characterization will be $\{(\mathbf{R}_{t_1+\sum_{h=1}^{i-1} b_{h,j}}, t_1), (\mathbf{R}_{t_2+\sum_{h=1}^{i-1} b_{h,j}}, t_2), ...\}$ at the h^{th} switch, where b_h is the length of the maximum busy period at switch h. The same random variable $\mathbf{R}_{t_m+\sum_{h=1}^{i-1} b_{h,j}}$ that bounds the maximum number of packets over an interval of length $t_m + \sum_{h=1}^{i-1} b_h$ at the entrance to the network, now bounds the maximum number of packets over a much *smaller* interval of length t_m at server i. In other words, the traffic characterization is burstier at server i than at the entrance.

G. End-to-End Delay Characteristics and Buffer Space Requirement

While the problem of deriving end-to-end delay bounds for a network of work-conserving servers has yet to be solved under general resource assignments, results have been obtained for Virtual Clock, WFQ, WF^{2}Q, SCFQ under the rate-proportional allocation strategy, and for Delay-EDD by considering each server in isolation. In both cases, the source traffic specifications are sufficient and traffic characterizations inside the network are not needed. In the former case, the end-to-end delay bound for a connection is a function of the guaranteed rate, which needs to be no less than the connection's average rate. In the latter case, the end-to-end delay bound is calculated as the sum of worst-case local delays at each switch. Since Delay-EDD scheduling is based on logical rather actual packet arrival times, local delay bounds at all switches can be calculated using the same source traffic characterization. To prevent packet loss, we assume buffer space is allocated on a per connection basis at each server during connection establishment time.

Table II presents the end-to-end characteristics and buffer space requirement of a connection when different work-conserving service disciplines are used. The table can be interpreted as the following. If a connection satisfies the traffic constraint as defined in the second column, and is allocated the amount of buffer space as listed in the fifth column, it can be guaranteed an end-to-end delay bound and delay-jitter bound as listed in the third and fourth column respectively, provided each server along the path uses the discipline in first column and appropriate admission control conditions are satisfied. In the table, C_i is link speed of the i^{th} switch on the path traversed by the connection, K_i is the number of connections sharing the link with the connection at the i^{th} switch, r_j is the guaranteed rate for the connection, and L_{max} is the largest packet size. Link delays are omitted from the expressions of end-to-end delays for simplicity.

Notice that the (σ, ρ) traffic model is used to characterize the traffic in all disciplines except Delay-EDD where a general traffic constraint function is used. The original Delay-EDD uses the $(Xmin, Xave, I, Smax)$ traffic model [16], [69]. However, the algorithm can easily be extended to accommodate connections using arbitrary deterministic traffic models that have associated traffic constraint functions. The corresponding admission control algorithm is described in [40]. A more general traffic model can characterize sources more accurately, thus resulting in a higher network utilization. A more detailed discussion on the relationship between achievable network utilization and accuracy of traffic characterization can be found in [34], [35].

There are several noteworthy points about the table. First, even though Virtual Clock, WFQ, and WF^{2}Q have a number of differences, they provide identical end-to-end delay bounds for connections that have leaky bucket constrained sources. In fact, if we compare the delay bound provided by them and that provided by the ideal fluid FFQ

discipline, we can see that they share the same main term $\frac{\sigma_j}{r_j}$, which can be interpreted as the time to send a burst of size σ_j in a fluid system with the guaranteed rate of r_j. For the three packet policies, there are additional terms to account for the fact that traffic is not infinitely divisible and the server needs to serve one packet at a time. Secondly, with the same guaranteed rate, the delay bound provided by SCFQ is larger than that provided by Virtual Clock, WFQ, and WF^{2}Q. This is due to the inaccuracy introduced by the approximation algorithm in calculating the virtual time. For all the four disciplines, since the server allocates service shares to connections proportional to their average rates, there is a coupling between the end-to-end delay bound and bandwidth provided to each connection. In particular, the end-to-end delay bound is inversely proportional to the allocated long term average rate. Thus, in order for a connection to get a low delay bound, a high bandwidth channel need to be allocated. This will result in a waste of resources when the low delay connection also has low throughput. WFQ and WF^{2}Q with general resource assignments do not have such a restriction [47]. However, due to the difficulties of characterizing traffic inside the network, the problem of deriving end-to-end delay bound for WFQ and WF^{2}Q under general resource assignments has yet to be solved. Delay-EDD does not have the problem of coupling between the allocations of delay bound and bandwidth either. However, the end-to-end delay bound listed in the table was derived without taking into account the delay dependency among successive switches, and is rather loose. As a final point to be noted, the end-to-end delay-jitter bounds for all disciplines are loose. In fact, the end-to-end delay-jitter bound is equal to the maximum end-to-end queueing delay. This can be easily understood by the following observation. Recall that delay-jitter bound is the maximum difference between delays of any two packets. In a network with work-conserving disciplines, a packet can experience little queueing delay when the network is lightly loaded while another packet can experience a much longer queueing delay when the network is heavily loaded. Thus the maximum difference between delays experienced by these two packets is the maximum end-to-end queueing delay.

H. Implementation Issues

As described in Section III-E, all the proposed work-conserving disciplines use the mechanism of a sorted priority queue. The insertion operation for a sorted priority queue has an intrinsic complexity of $O(\log N)$ [36], where N is the number of packets in the queue. In a network that is designed to support many connections with bursty traffic, the switch usually has buffer space for a large number of packets. For example, the queue module of each link of the Xunet switch contains memory to store 512K ATM cells [28]. Potentially, the queue length can be long. It may not be feasible to implement an operation that has an $O(\log N)$ complexity at very high speed. Since all disciplines ensure that packets on the same connection are serviced in the order of their arrivals, a clever implementation can arrange packets on a per-connection basis and sort only the first packet of each connection. Recently, it was reported that a sequencer chip clocked at 33 Mhz has been implemented to support sorting of up to 256 packets [6]. Thus, up to 256 connections or classes of connections can be supported with such an implementation. It is unclear whether such an implementation can scale to higher speed or more connections.

A sorted priority queue mechanism also requires computation of the priority index on a per packet basis. For service disciplines that use real time to compute the priority index, such as Virtual Clock and Delay-EDD, the computation is simple and straightforward. For service disciplines that use virtual times in another reference queueing system, such as WFQ and WF^{2}Q, the computation is more complex. In particular, both WFQ and WF^{2}Q need to keep track the set of connections that have packets queued at any time instant. This is very difficult to implement at high speed. SCFQ simplifies the computation by using an an approximate algorithm that does not need to keep track of the set of active connections.

IV. Non-Work-Conserving Disciplines

In Section III-F, we showed that in order to derive end-to-end delay bounds and buffer space requirements in a networking environment, traffic needs to be characterized inside the network on a per connection basis. With work-conserving disciplines, the traffic pattern is distorted inside the network due to network load fluctuation, and there are a number of difficulties and limitations in deriving the traffic characterization after the distortion.

Another approach to deal with the problem of traffic pattern distortions is to control the distortions at each switch using *non-work-conserving disciplines*. With a non-work-conserving discipline, the server may be idle even when there are packets waiting to be sent. Non-work-conserving disciplines were seldom studied in the past. This is mainly due to two reasons. First, in most of previous performance analyses, the major performance indices are the *average* delay of all packets and the *average* throughput of the server. With a non-work-conserving discipline, a packet may be held in the server even when the server is idle. This may increase the average delay of packets and decrease the average throughput of the server. Secondly, most previous queueing analyses assumed a single server environment. The potential advantages of non-work-conserving disciplines in a networking environment were therefore not realized. In guaranteed performance service, the more important performance index is the end-to-end delay *bound* rather than the average delay. In addition, delay needs to be bounded in a *networking* environment rather than just in a single node. Therefore, the above reasons for not using non-work-conserving disciplines do not hold any more.

Several non-work-conserving disciplines have been proposed in the context of high speed integrated services networks. Among them are: Jitter Earliest-Due-Date (Jitter-EDD) [56], Stop-and-Go [21], Hierarchical Round Robin (HRR) [26], and Rate-Controlled Static Priority

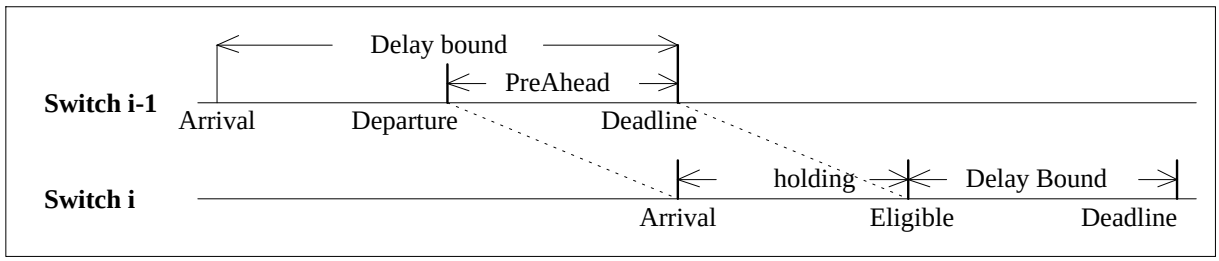


Fig. 10. Packet Service in Jitter-EDD

(RCSP) [62]. In this section, we first describe each of the algorithms in turn, then present a unified framework called rate-controlled service disciplines and show that all of them can be represented in such a framework. Finally, we discuss the end-to-end delay characteristics and buffer space requirements for these disciplines within the framework of rate-controlled service disciplines.

A. Jitter-Earliest-Due-Date

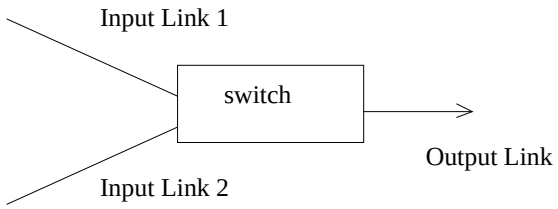
The Jitter-EDD discipline [56] extends Delay-EDD to provide delay-jitter bounds (that is, a bound on the maximum delay difference between two packets). After a packet has been served at each server, a field in its header is stamped with the difference between its deadline and the actual finishing time. A regulator at the entrance of the next server holds the packet for this period before it is made eligible to be scheduled.

Jitter-EDD is illustrated in Figure 10, which shows the progress of a packet through two adjacent servers. In the first server, the packet got served $PreAhead$ seconds before its deadline. So, in the next server, it is made eligible to be sent only after $PreAhead$ seconds. Since there is a constant delay between the eligibility times of the packet at two adjacent servers, the packet stream can be provided a delay jitter bound. Assuming there is no regulator at the destination host, the end-to-end delay jitter bound is the same as the local delay bound at the last server.

B. Stop-and-Go

As shown in Figure 11, Stop-and-Go uses a framing strategy [20]. In such a strategy, the time axis is divided into frames, which are periods of some constant length T. 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 $0 \leq \theta < T$. 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, the output link may be left idle even when there are packets in the switch to be transmitted, thus, Stop-and-Go is a non-work-conserving policy.

Stop-and-go ensures that packets on the same frame at the source stay in the same frame throughout the network. If the traffic is characterized at the source by (r, T), i.e., no more than $r \cdot T$ bits are transmitted during any frame of size T, it satisfies the same characterization throughout the network. By maintaining traffic characteristics throughout the network, end-to-end delay bounds can be guaranteed in a network of arbitrary topology as long as each local server can ensure local delay bounds for traffic characterized by (r, T) specification.



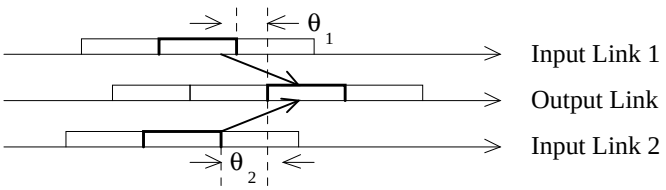


Fig. 11. Synchronization between input and output links in Stop-and-Go

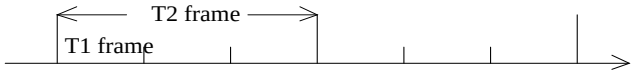


Fig. 12. Two levels of framing with $T_2 = 3T_1$

The framing strategy introduces the problem of coupling between delay bound and bandwidth allocation granularity. The delay of any packet at a single switch is bounded by two frame times. To reduce the delay, a smaller T is desired. However, since T is also used to specify traffic, it is tied to bandwidth allocation granularity. Assuming a fixed packet size P, the minimum granularity of bandwidth allocation is $\frac{P}{T}$. To have more flexibility in allocating bandwidth, or a smaller bandwidth allocation granularity, a larger T is preferred. It is clear that low delay bound and fine granularity of bandwidth allocation cannot be achieved simultaneously in a framing strategy like Stop-and-Go.

To get around this coupling problem, a generalized version of Stop-and-Go with multiple frame sizes is proposed. In the generalized Stop-and-Go, the time axis is divided

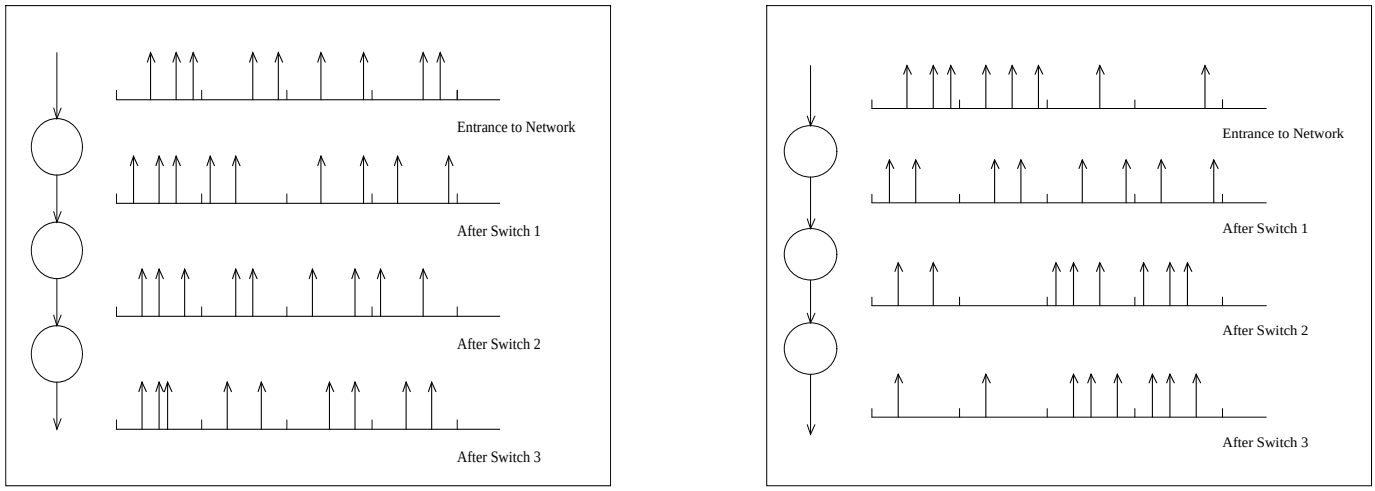


Fig. 13. Difference between Stop-and-Go and HRR

into a hierarchical framing structure as shown in Figure 12. For a n level framing with frame sizes $T_1, \cdots, T_n$, and $T_{m+1} = K_m T_m$ for $m = 1, \cdots, n-1$, packets on a level p connection need to observe the Stop-and-Go rule with frame size T_p, i.e., level p packets which arrived at an output link during a T_p frame will not become eligible for transmission until the start of next T_p frame. Also, for two packets with different frame sizes, the packet with a smaller frame size has a non-preemptive priority over the packet with a larger frame size. With multi-frame Stop-and-Go, 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 bandwidth allocation granularity still exists within each frame. In [52], a scheme is proposed to add a separate shaping mechanism at the network entry point for networks with framing based disciplines. With traffic shaping at the entrance to the network, it is possible to multiplex several connections on a single slot of a frame, therefore avoid the problem of coupling between frame size and bandwidth allocation granularity.

C. Hierarchical Round Robin

HRR is similar to Stop-and-Go in that it also uses a mult-level framing strategy. A slot in one level can either be allocated to a connection or to a lower level frame. The server cycles through the frame and services packets according to the assignment of slots. If the server cycles through a slot assigned to a connection, one packet from that connection is transmitted; if it cycles through a slot assigned to a lower level frame, it will service one slot from the lower level frame in the same fashion. HRR is non-work-conserving in the sense that if it cycles through a slot with no packets waiting, it will leave the server idle for that slot time rather than sending packets assigned to other slots.

Similar to Stop-and-Go, HRR also maintains traffic smoothness inside the network due to its non-work-conserving nature. However, there are also important differences between HRR and Stop-and-Go. The example shown in Figure 13 illustrates their difference. In the example, it is assumed that 3 packet transmission times are allocated to a connection in each frame. In Stop-and-Go, packets that are transmitted in the same frame at the entrance to the network will be transmitted in the same frame on all the links traversed by the connection. The difference between delays experienced by any two packets from the source to any server is bounded by T, where T is the frame size. In HRR, packets that are transmitted in the same frame at the entrance to the network do not necessarily stay in the same frame inside the network; however, the property that *no more than three packets from the connection are transmitted during one frame time* holds throughout the network.

Since HRR uses the framing strategy, it also has the problem of coupling between delay and bandwidth allocation granularity.

D. Rate-Controlled Static Priority

While the Earliest-Due-Date algorithm can provide flexible delay bounds and bandwidth allocation, it is based on a sorted priority mechanism, which is difficult to implement. Stop-and-Go and HRR use a framing strategy instead of the sorted priority to achieve simplicity, however, such a strategy introduces coupling between delay bound and bandwidth allocation granularity. The goal of RCSP is to achieve flexibility in the allocation of delay and bandwidth as well as simplicity of implementation.

As shown in Figure 14, a RCSP server has two components: a rate-controller and a static priority scheduler. Conceptually, a rate controller consists of a set of regulators corresponding to each of the connections traversing the server; each regulator is responsible for shaping the input traffic of the corresponding connection into the desired traffic pattern. Upon arrival of each packet, an eligibility time is calculated and assigned to the packet by the regu-

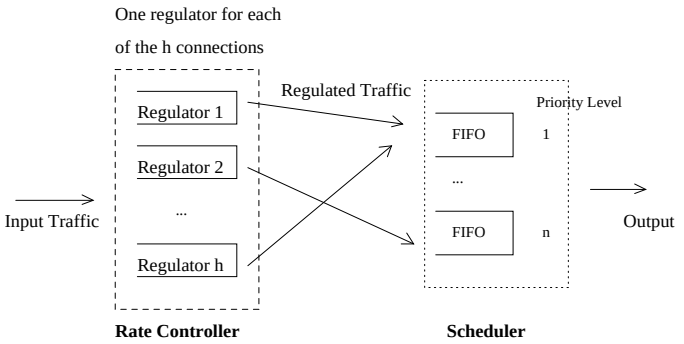


Fig. 14. Rate-Controlled Static Priority

lator. The packet is held in the regulator till its eligibility time before being handed to the scheduler for transmission. Different ways of calculating the eligibility time of a packet result in different types of regulators. As will be discussed in [61] and Section IV-F, many regulators can be used for RCSP. We will consider two examples in this section. The $(Xmin, Xave, I)$ RJ regulator ensures that the output of the regulator satisfy the $(Xmin, Xave, I)$ traffic model, while the DJ_r regulator ensures that the output traffic of the regulator is exactly the same as the the output traffic of the regulator at the previous server. Thus, if the traffic satisfies the $(Xmin, Xave, I)$ characterization at network entrance, both types of regulators will ensure that the output of the regulator, which is the input to the scheduler, will satisfy the same traffic characterization.

For a $(Xmin, Xave, I)$ RJ regulator, where $Xmin \leq Xave < I$ holds, the eligibility time of the k^{th} packet on connection j at the i^{th} server along its path, e_i^k, is defined with reference to the eligibility times of packets arriving earlier at the server on the same connection:

$$e_{i,j}^k = -I, \quad k < 0 \tag{3}$$

$$e_{i,j}^1 = a_{i,j}^1 \tag{4}$$

$$e_{i,j}^k = max(e_{i,j}^{k-1} + Xmin, e_{i,j}^{k-\lfloor \frac{I}{Xave} \rfloor + 1} + I,\ a_{i,j}^k), \quad k > 1 \tag{5}$$

where $a_{i,j}^k$ is the time the k^{th} packet on connection j arrived at the i^{th} server. (3) is defined for convenience so that (5) holds for any $k > 1$.

From this definition, we can see that $e_{i,j}^k \geq a_{i,j}^k$ always holds, i.e., a packet is never eligible before its arrival. Also, if we consider the sequence of packet eligibility times at i^{th} server , $\{e_{i,j}^k\}_{k=1,2,...}$, it always satisfies the (X_{min}, X_{ave}, I) traffic characterization.

The eligibility time of a packet for a DJ_r regulator is defined with reference to the eligibility time of the same packet at the immediately upstream server. The definition assumes that the queueing delays of packets on the connection, and the link delay from the upstream server to the current server, are bounded. Let $d_{i-1,j}$ be the local delay bound for the connection in the scheduler at the $(i-1)^{th}$ server, and π_i be the maximum link delay from the $(i-1)^{th}$ server to the i^{th} server. The DJ_r regulator is defined as: is defined as:

$$e_{0,j}^k = a_{0,j}^k \tag{6}$$

$$e_{i,j}^k = e_{i-1,j}^k + d_{i-1,j} + \pi_i, \quad i > 0 \tag{7}$$

It is easy to show that the following holds:

$$e_{i,j}^{k+1} - e_{i,j}^k = a_{0,j}^{k+1} - a_{0,j}^k \quad \forall k, i \geq 0 \tag{8}$$

i.e., the traffic pattern on a connection at the output of the regulator of every server traversed by the connection is exactly the same as the traffic pattern of the connection at the *entrance* to the network.

The scheduler in a server RCSP uses a non-preemptive Static Priority policy: it always selects the packet at the head of highest priority queue that is not empty. The SP scheduler has a number of priority levels with each priority level corresponding to a delay bound. Each connection is assigned to a priority level during connection establishment time. Multiple connections can be assigned to the same priority level, and all packets on the connections associated with a priority level are appended to the end of the queue for that priority level.

E. A Framework For Non-Work-Conserving Disciplines

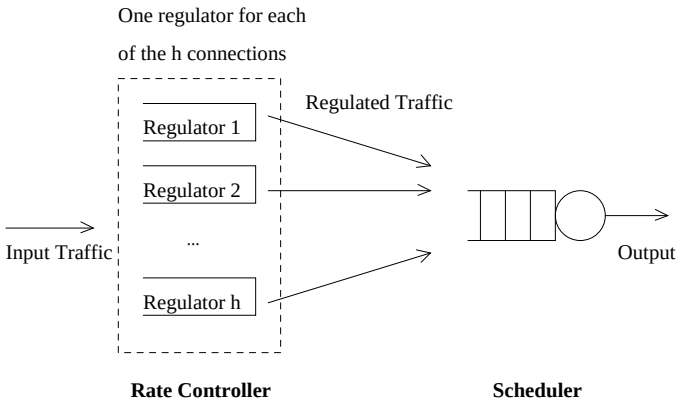


Fig. 15. Rate-Controlled Service Disciplines

In previous sections, we described four non-work-conserving disciplines. In this section, we show that all of them can be expressed by a general class of disciplines called rate-controlled service disciplines [64]. As shown in Figure 15, a rate-controlled server can be considered as a generalization of RCSP: it also has two components, a rate-controller and a scheduler. The rate controller, which consists of a number of regulators, is responsible for shaping traffic. The scheduler is responsible for multiplexing eligible packets coming from different regulators. While RCSP uses two types of regulators and the Static Priority scheduler, many other regulators and schedulers can be used. By having different combinations of regulators and schedulers, we have a general class of disciplines. Among the four disciplines discussed in this section, RCSP and Jitter-EDD are rate-controlled servers, Stop-and-Go and HRR can be implemented by rate-controlled servers by selecting appropriate regulators and schedulers.

Jitter-EDD can be viewed as a combination of a Earliest-Due-Date scheduler and DJ_e regulators, which are defined as follows:

$$e_{i,j}^k = a_{i,j}^k + Ahead_{i-1,j}^k \tag{9}$$

where $Ahead_{i-1,j}^k$ is the amount of time the packet is ahead of schedule at the $(i-1)^{th}$ server along the path.

A Stop-and-Go server with n frame sizes ($T_1 < T_2 < ... < T_n$) can be implemented by a rate-controlled server with an n-level static priority scheduler and DJ_s regulators:

$$e_{i,j}^k = a_{i,j}^k + Ahead_{i-1,j}^k + \theta_{i,j} \tag{10}$$

where $Ahead_{i-1,j}^k$ is the amount of time the packet is ahead of schedule in switch $i-1$, and $\theta_{i,j}$ is the synchronization time between the framing structures on the input and output links. Each pair of input and output links in a switch may have a different value of θ. Figure 11 illustrates this synchronization time. In the static priority scheduler, the delay bound associated with level m is T_m, $1 \le m \le n$.

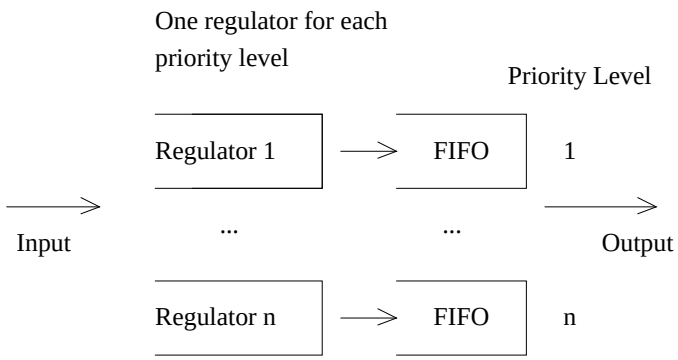


Fig. 16. Implement Stop-and-Go Using a Rate-Controlled Server

Although the above implementation of Stop-and-Go is very similar to RCSP, there are also important differences, as can be seen by comparing Figure 14 and Figure 16. In an RCSP server, there is a regulator for each connection, and the regulated traffic on each connection can be assigned to *any* priority level in the scheduler. In a Stop-and-Go server, regulators are associated with priority levels in the scheduler. In fact, there is a one-to-one correspondence between the regulator and the priority level. The traffic on a connection has to be specified with respect to the frame size, which is the same as the connection's local delay bound. This not only introduces the coupling between the allocations of bandwidth and delay bounds, but also implies that admission control algorithm has to be based on a busy period argument, which tends to produce looser bounds when compared to more elaborate analysis [8], [63].

Because of the framing, there are dependencies among the local delay bounds at each priority level in a Stop-and-Go server. In particular, $T_{m+1} = K_m T_m$ must hold, with $1 \le m < n$, and K_m being an integer. Furthermore, the delay bound allocations for each connection in different servers are coupled with one another. In [21], a connection has to have the same frame size in all the servers. In [65], a looser requirement is presented: the frame times of a connection along the path should be non-decreasing. None of these restrictions apply to RCSP. The impact of flexibility of allocating delay bounds inside the network on network utilization was studied in [45].

Discipline	$e_{i,j}^k$ defined in regulator	Scheduler
RCSP/DJ_r	$a_0^k + Ahead_{i-1,j}^k + (\pi_i - \pi_i^k)$	SP
	$e_{i-1,j}^k + d_{i-1,j} + \pi_i^k$	
Jitter-EDD	$a_{i,j}^k + Ahead_{i-1,j}^k$	EDD
	$e_{i-1,j}^k + d_{i-1,j} + \pi_i^k$	
Stop-and-Go	$a_{i,j}^k + Ahead_{i-1,j}^k + \theta$	SP
	$e_{i-1,j}^k + T^m + \pi_i^k$	
RCSP/RJ_r	$\max(e_{i,j}^{k-1} + Xmin_j,$ $e_{i,j}^{k-\lfloor \frac{I}{Xave_j} \rfloor + 1} + I, a_{i,j}^k)$	SP
HRR	$\max(a_{i,j}^k, e_{i,j}^{k-a_{i,j}} + T_j^m)$	SP

TABLE III
NON-WORK-CONSERVING DISCIPLINES

A Hierarchical Round Robin server with n frame sizes ($T_1 < T_2 < ... < T_n$) can be implemented by a rate-controlled server with an n-level static priority scheduler and RJ_h regulators defined by:

$$e_{i,j}^k = max(a_{i,j}^k + \tau, e_{i,j}^{k-q_{i,j}^m} + T_m) \tag{11}$$

where $a_{i,j}^k + \tau$ is the beginning time of the next frame and $q_{i,j}^m$ is the maximum number of packets that can be served on the connection within each frame of size T_m. In the static priority scheduler, the delay bound associated with level m is T_m, $1 \le m \le n$. If a connection traverses a level-m RJ_h regulator, it has to be assigned to the priority level m in the scheduler. This introduces the coupling between delay and bandwidth allocation. In contrast, in an RCSP server, a connection can be assigned to any priority level regardless of its rate parameters.

Table III summarizes the regulators and schedulers for the four disciplines. Notice that there are two equivalent definitions of eligibility times for each of the DJ_r, DJ_e and DJ_s regulators.

F. Delay-jitter-control and Rate-jitter-control Regulators

As shown in Table III, the regulators for RCSP/DJ_r, Jitter-EDD, and Stop-and-Go are very similar. For each of the three regulators, the eligibility time of a packet at a switch is defined with respect to the eligibility time of the *same* packet at the *previous* switch. Also, the regulators for RCSP/RJ_r and HRR are similar in that the eligibility time of a packet at a switch is defined with respect to *earlier arriving* packets at the *same* switch. In [61], two general classes of regulators called delay-jitter controlling regulators and rate-jitter controlling regulators are defined. Regulators for RCSP/DJ_r, Jitter-EDD, and Stop-and-Go fall into the former class, whereas regulators for RCSP/RJ_r and HRR are in the later class.

For a delay-jitter controlling regulator, the eligibility time of a packet is defined with reference to the eligibility time of the same packet at the immediately upstream

	traffic constraint	end-to-end delay bound	end-to-end delay-jitter bound	buffer space at h^{th} switch
Stop-and-Go	(r_j, T_j)	$nT_j + \sum_{i=1}^{n} \theta_i$	T_j	$r_j(2T_j + \theta_i)$
HRR	(r_j, T_j)	$2nT_j$	$2nT_j$	$2r_jT_j$
Rate-controlled Servers with $b^*(\cdot)$ RJ regulators	$b_j(\cdot)$	$D(b_j, b^*) + \sum_{i=1}^{n} d_{i,j}$	$D(b_j, b^*) + \sum_{i=1}^{n} d_{i,j}$	$\sigma_j + b^*(d_{1,j})$ for 1^{st} switch
				$b^*(d_{i-1,j} + d_{i,j})$ for j^{th} switch $j > 1$
Rate-controlled Servers with $b^*(\cdot)$ RJ regulator for 1st switch and DJ regulators for other switches	$b_j(\cdot)$	$D(b_j, b^*) + \sum_{i=1}^{n} d_{i,j}$	$D(b_j, b^*) + d_{n,j}$	$\sigma_j + b^*(d_{1,j})$ for 1^{st} switch
				$b^*(d_{i-1,j} + d_{i,j})$ for j^{th} switch $j > 1$

TABLE IV

END-TO-END DELAY, DELAY JITTER AND BUFFER SPACE REQUIREMENT FOR NON-WORK-CONSERVING DISCIPLINES

server. The following definition assumes that the queueing delays of packets on the connection at the immediately upstream server and the link delay from the upstream server to the current server are bounded.

$$e^k_{1,j} = a^k_{1,j} \quad (12)$$

$$e^k_{i,j} = e^k_{i-1,j} + d_{i-1,j} + \pi_{i,j} + \theta_{i,j}, \quad i > 1 \quad (13)$$

where $a^k_{1,j}$ is the arrival time of the k^{th} packet at the entrance to the network, and $\theta_{i,j}$ is a constant delay.

While delay-jitter (DJ) regulators maintain all the traffic characteristics by completely reconstructing traffic pattern at output of each regulator, rate-jitter (RJ) regulators only maintain certain characteristics of the traffic. Depending on which traffic models are used by the resource allocation algorithm, different RJ regulators can be defined. As discussed in Section II-B.2 and in [61], each deterministic traffic model, such as $(Xmin, Xave, I, Smax)$ [16], (r, T) [21], (σ, ρ) [8], and D-BIND [35], defines a deterministic traffic constraint function $b(\cdot)$. A monotonic increasing function $b_j(\cdot)$ is called a deterministic traffic constraint function of connection j if during *any* interval of length u, the number of bits arriving on j during the interval is no greater than $b_j(u)$. For each traffic model with a corresponding deterministic traffic constraint function $b(\cdot)$, we can construct a rate-jitter controlling regulator with the following definition of $e^k_{i,j}$:

$$e^k_{i,j} = min\{v : v \geq max(e^{k-1}_{i,j}, a^k_{i,j}), \; E_{i,j}(u, v) \leq b_j(v - u) \;\; \forall u \leq v\} \quad (14)$$

where $E_{i,j}(.,.)$, defined below, is the number of bits on connection j that become eligible in interval (u, v) at the i^{th} server:

$$E_{i,j}(u, v) = \sum_k (L^k_j | u \leq e^k_{i,j} < v) \quad (15)$$

and L^k_j is the length of the k^{th} packet on connection j.

Equation (14) is very general and defines a class of rate-jitter controlling policies. Any deterministic traffic model that can be defined with a traffic constraint function has a corresponding rate-jitter controlling regulator. The regulator for HRR is a rate-jitter controlling regulator using the (r, T) traffic model, and the regulator for RCSP/RJ_r is the one using the $(Xmin, Xave, I)$ model. In addition, the implementation of rate-jitter controlling regulators can be very simple. For example, the regulator for the (σ, ρ) traffic model can be implemented by the popular leaky bucket mechanism [54].

G. End-to-End Delay Characteristics and Buffer Space Requirements

The end-to-end delay characteristics and buffer space requirement for non-work-conserving disciplines are shown in Table III. In the table, $D(b_j, b^*)$ is the worst-case delay introduced by a RJ regulator with the constraint function $b^*(\cdot)$ for a traffic stream characterized by the constraint function $b_j(\cdot)$.

As shown in the table, the two frame-based disciplines Stop-and-Go and HRR have similar end-to-end delay bounds and buffer space requirements. The only major difference between them is that Stop-and-Go provides a tighter jitter bound than HRR. This is because Stop-and-Go uses delay-jitter control while HRR uses rate-jitter control.

While the end-to-end delay bounds for Stop-and-Go and HRR are derived by considering each server in isolation, tighter end-to-end delay bounds can be derived for rate-controlled service disciplines by taking into consideration the delay dependencies in successive switches traversed by a connection [19]. The key observation is that, $b^*(\cdot)$, the traffic constraint function used in the regulators, does not have to be the same as $b_j(\cdot)$, the traffic constraint function used to specify the source. By appropriately setting parameters for regulators and local delay bounds at schedulers, rate-controlled service disciplines can provide end-to-end delay bounds at least as tight at those provided by FFQ-based work-conserving service disciplines. To compare with FFQ-based disciplines, assume that the traffic on connection j is characterized by the (σ_j, ρ_j) model. That is,

$$b_j(u) = \sigma_j + \rho_j u \quad (16)$$

We consider two cases. In the first case, only RJ regulators are used. The traffic constraint function for the regulators and the local delay bound for each scheduler are defined as follows:

$$b^*(u) = L_{max} + \rho_j u \quad (17)$$

$$d_{i,j} = \frac{L_{max}}{\rho_j} + \frac{L_{max}}{C_i} \quad (18)$$

In the second case, the first switch still uses the RJ regulator defined above, but all subsequent switches use DJ

regulators with $\theta_{i,j} = 0$. Same local delay bounds are assigned to each switch.

It can be shown that the following holds:

$$D(b_j, b^*) = \frac{\sigma_j}{\rho_j} \tag{19}$$

According to Table IV, an end-to-end delay bound of $\frac{\sigma_j + nL_{max}}{\rho_j} + \sum_{i=1}^{n} \frac{L_{max}}{C_i}$ can be provided to the connection in both cases. Compared to Table II, the above delay bound is identical to that provided by WFQ, WF^{2}Q, and Virtual Clock servers. The about assignments are just examples to illustrate the flexibility of rate-controlled service disciplines. More elaborate assignments of regulators and local delay bounds can achieve higher network utilization [19]. With rate-controlled service disciplines, since the traffic can be characterized throughout the network, end-to-end delay bounds can be derived for general resource assignments. WFQ, WF^{2}Q, and Virtual Clock do not have such a property. In fact, it has been shown in [19] that by properly setting parameters for regulators and local delay bounds for schedulers, rate-controlled service disciplines can always out-perform FFQ-based disciplines in terms of the number of connections that can be accepted.

Compared to FFQ-based disciplines, rate-controlled service disciplines have the additional advantage of requiring less buffer space inside the network to prevent packet loss. Based on (17), (18), (19), and Table IV, it can be easily shown that the total amount of buffer space required for connection j in a network of rate-controlled servers is:

$$\sigma_j + (2n-1)L_{max} + (2\sum_{i=1}^{n-1} \frac{L_{max}}{C_i} + \frac{L_{max}}{C_n})\rho_j \tag{20}$$

which is less than

$$\sigma_j + (4n-2)L_{max} \tag{21}$$

Alternatively, based on Table II, the total amount of buffer space required for connection j in a network of WFQ servers is

$$n\sigma_j + \frac{n(n-1)}{2} L_{max} \tag{22}$$

Since σ_j, which is the maximum burst size, is usually much larger than a packet size, the terms with σ_j dominate (21) and (22). While the amount of the buffer space required for a connection increases linearly with the number of hops when WFQ is used, the amount of buffer space is almost independent of the number of hops when rate-controlled service disciplines are used.

H. Bounding Delay in a Single Scheduler

In the previous section, we showed that end-to-end delay bounds can be provided in a network of non-work-conserving servers only when the local delay bound can be provided at the scheduler in each server. Many schedulers such as FCFS, SP, and EDD can be used. Various analysis techniques have been developed to bound the delay in a single scheduler when the input traffic to the scheduler is constrained. In a rate-controlled server, the input traffic to the scheduler is always constrained due to the use of regulators. Therefore, these analysis techniques can be directly applied.

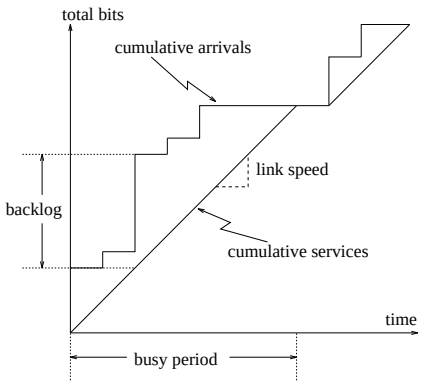


Fig. 17. Concepts: Delay, Backlog and Busy Period

Figure 17 illustrates the basic concept used in the analysis developed by Cruz [8]. The horizontal axis is time and the vertical axis is bits. The upper curve represents the total number of bits that have arrived in the scheduler by time t and the lower curve represents the total number of bits transmitted by time t. The difference between the two curves is the number of bits currently in the queue, or the *backlog* function. When the backlog function returns to zero (the two curves meet) there are no bits in the queue and thus a busy period has ended. The key to this analysis is that if the upper curve is a deterministic bounding curve, then the maximum delay can be expressed as a function of the two curves. For example, the following two observations hold: the maximum busy period provides an upper bound on delay for any work-conserving server; the maximum backlog divided by the link speed provides an upper bound on delay for a FCFS server. Delay bounds for other policies can also be expressed [1], [8], [40], [48].

Table V shows delay bound tests for FCFS, SP, and EDD schedulers as derived in [40]. Notice that while a FCFS scheduler only provides one delay bound and an SP scheduler provides a fixed number of delay bounds, an EDD scheduler can provide a continuous spectrum of delay bounds. In an integrated services networks where applications have diverse traffic characteristics and performance requirements, the flexibility of allocating delay bounds affects the utilization that can be achieved by guaranteed service traffic. In [34], it is shown that SP and EDD schedulers can out-perform FCFS scheduler significantly in terms of link utilization when connections have different delay bounds. However, there is little difference in achievable link utilization between SP and EDD schedulers. Since an SP scheduler has only a fixed number of FCFS queues, it is much easier to implement than an EDD scheduler which requires a sorted queue mechanism. Thus, an SP scheduler strikes a good balance between simplicity of implementation and flexibility in allocating delay bounds [62].

Delay Bound Test	Condition
FCFS	$d \geq \sum_{j \in \mathcal{N}} b_j(t) - t + \max_{k \in \mathcal{N}} s_k$ for all $t \geq 0$.
SP	$(\exists \tau \leq d_p)\ t + \tau \geq \sum_{j \in C_p} b_j(t) + \sum_{q=1}^{p-1} \sum_{j \in C_q} b_j(t+\tau) + \max_{r>p} s_r$ for all $p,\ t \geq 0$.
EDF	$\begin{cases} t \geq \sum_{j \in \mathcal{N}} b_j(t - d_j) & \text{for all } t \geq 0. \\ t \geq \sum_{j \in \mathcal{N}} b_j(t - d_j) + \max_{d_k > t} s_k & \text{for all } d_1 \leq t < d_{\lvert\mathcal{N}\rvert}. \end{cases}$

TABLE V
DELAY BOUND TESTS FOR FCFS, SP, AND EDF PACKET SCHEDULERS.

I. Implementation Issues

Among the four non-work-conserving disciplines discussed in this paper, HRR, Stop-and-Go, and RCSP all use a non-preemptive Static Priority scheduler. Only Delay-EDD use an EDD scheduler which requires a sorted priority queue mechanism. The complexity of implementing sorted priority queue has been discussed in III-H. Among HRR, Stop-and-Go, and RCSP, the former two disciplines implement the rate-controller and the scheduler using one framing mechanism while RCSP needs to implement both using separate mechanisms.

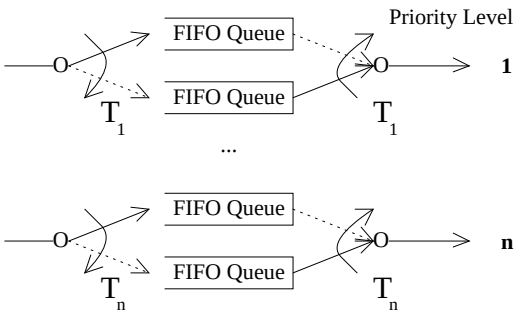


Fig. 18. Implementation of Stop-and-Go

To implement Stop-and-Go, mechanisms are needed at both the link level and at the queue management level. At the link level, a framing structure is needed, and there is a synchronization requirement such that the framing structure is the same at both the sending and the receiving ends of the link. At the queue management level, two FIFO queues are needed for each priority level, one storing the eligible packets ready to be transmitted, the other storing the packets that won't be eligible until the end of the current frame time. Mechanisms are needed to swap the two FIFO queues at the start of each frame time. Also, the set of FIFO queues with eligible packets need to be serviced according to a non-preemptive static priority policy. This is shown Figure 18.

HRR does not need the framing structure at the link layer. However, it requires buffering on a per connection basis and a set of timers to perform rate-control. An implementation of a prototype HRR server with 16 priority levels has been reported [27].

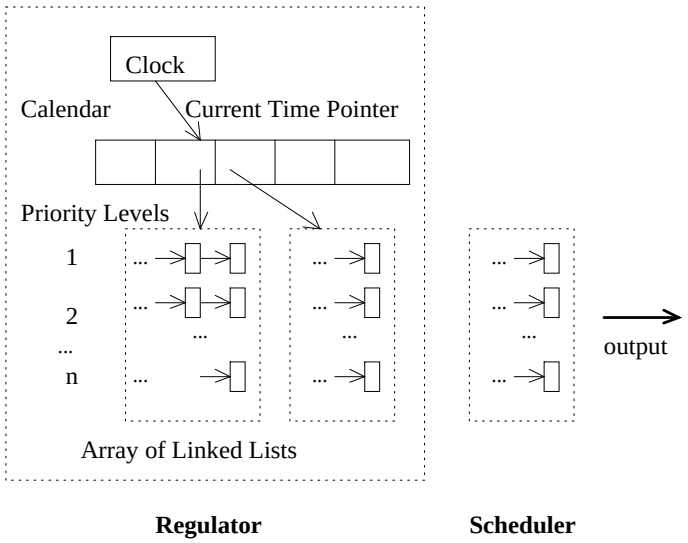


Fig. 19. Implementation of RCSP

RCSP seems to be more complex than Stop-and-Go and HRR since it requires traffic regulation on a per connection basis. However, the conceptual decomposition of the rate controller into a set of regulators in RCSP does not imply that there must be multiple physical regulators in an implementation; a common mechanism can be shared by all logical regulators. Figure 19 shows an example implementation of RCSP based on a modified version of a calendar queue [4]. A calendar queue consists of a clock and a calendar, which is a pointer array indexed by time. Each entry in the calendar points to an array of linked lists indexed by priority levels. The clock ticks at fixed time intervals. Upon every tick of the clock, the linked lists in the array indexed by the current time are appended at the end of the scheduler's linked lists. Packets from the linked list of one priority level in the rate-controller are appended to the linked list of the same priority level in the scheduler. The

scheduler just selects the first packet at the highest priority queue that is non-empty. As can be seen, the data structures used in the proposed implementation are simple: arrays and linked lists. The operations are all constant-time ones: array indexing, insertion at the tail of a linked list, deletion from the head of a linked list. Another implementation of RCSP that is based on a two-dimensional shifters is proposed in [44].

We would like to point out that a calendar queue is a simpler mechanism that a sorted priority queue. In a calendar queue, only packets pointed by the current time pointer are dequeued at every clock tick. In a sorted priority queue, the next packet needs to be dequeued each time the server finishes service of the current packet. If the sorted queue is implemented by a calendar queue, the dequeueing operation potentially needs to go through *all* the entries in the calendar.

J. Work-Conserving Rate-Controlled Service Disciplines

In previous sections, we showed that non-work-conserving rate-controlled service disciplines exhibit several interesting properties that make them desirable for supporting guaranteed performance service. These properties include:

1. End-to-end delay analysis can be decomposed into local delay analysis at each switch, and tight end-to-end delay bounds can be derived with such simple analysis for general resource assignments.
2. Heterogeneous servers with different schedulers and regulators can be used at different switches.
3. By separating the rate-control mechanism and the scheduler, the allocation of delay bounds and bandwidth can be decoupled without using the sorted priority queue mechanism.
4. Due to the traffic regulation inside the network, less buffer space is needed at each switch to prevent packet los.
5. The traffic at the exit of the network satisfies certain desirable properties, for example, bounded rate or delay jitter.

However, non-work-conserving disciplines also have several disadvantages. First, with non-work-conserving disciplines, a client is *always* punished when it sends more than specified. Even though this is acceptable under the guaranteed service model, it puts an extra burden on the client to always characterize its traffic correctly. For applications that use live sources such as video conferencing, it is difficult to come up with an accurate traffic characterization *before* the data transmission. If connections are always punished whenever it sends more than specified regardless whether there are spare resources available at that time, they may have to specify the characterization based on an over-estimation of the traffic, which results in a waste of resources. Secondly, while non-work-conserving disciplines optimize for guaranteed performance service, they may negatively affect the performance of other packets. For example, with a non-work-conserving discipline, the server will be idle if there are only guaranteed service packets queued at the server and none of them are eligible for transmission. If some best-effort service packets arrive at the server right after these guaranteed service packets become eligible, the best-effort packets will have to wait before the guaranteed service packets finish service. However, if a work-conserving policy were used, the guaranteed service packets would have been served before the arrival of the best-effort service packets, therefore, the best-effort service packets would not have to wait after they arrive.

A non-work-conserving rate-controlled server can be easily modified to be work-conserving [62], [10], [19]. In a work-conserving rate-controlled server, there is one more queue in the scheduler, called the *stand-by queue* [62]. It works as follows:

- All the packets in the rate-controller are also queued in the stand-by queue. Packets are inserted or deleted from the rate-controller and the stand-by queue simultaneously.
- The scheduler will service the next packet in the stand-by queue only if there are no non-guaranteed packets and eligible guaranteed packets in the scheduler.

The stand-by queue allows the non-eligible packets to *stand by* at the scheduler, so that they can be transmitted when there is spare capacity at the output link.

In [19], it has been shown that the resulted work-conserving rate-controlled server can provide the same end-to-end delay bound as its non-work-conserving counterpart. Among the five properties listed at the beginning of the section, the first three, and perhaps the more important ones among all, still hold for rate-controlled servers with stand-by queues.

As a last note in the section, we would like to point out that even without the stand-by queue, a rate-controlled discipline does not necessarily have to be non-work-conserving. In [2], it has been shown that the Worst-case Fair Weighted Fair Queueing (WF^2Q) is equivalent to a rate-controlled server with a WFQ scheduler and regulators defined by:

$$e_i^k = b_{i,FFQ}^k \tag{23}$$

where $b_{i,FFQ}^k$ is the time the packet starts service in the corresponding FFQ system.

In addition, it has been shown that WF^2Q is work-conserving. Notice that the regulator defined above is neither a rate-jitter controlling regulator, which is defined by a traffic constraint function, nor a delay-jitter controlling regulator, which is defined by the local delay bound at the previous server. Instead, it is defined with reference to a FFQ system, therefore, the eligibility times of packets are *dependent* on the system load.

V. Summary

In this paper, we have examined a number of packet service disciplines that have been proposed to support guaranteed performance service connections in packet-switching integrated services networks. As shown in Figure 20, these disciplines can be classified along two dimensions: (1) how the service discipline allocates, explicitly or implicitly, dif-

Interaction of Multiple Servers		*Delay/Bandwidth Allocation*				
			Server has one mechanism		**Server has two mechanisms: Rate-controller and Scheduler**	
			Sorted Priority Queue	**Multi-level Framing**	**Scheduler using Sorted Queue**	**Scheduler using Static Priority**
	Control Distortion (non work conserving)	**Delay-jitter Control**		Stop-and-Go	Jitter-EDD	RCSP
		Rate-jitter Control		HRR		
	Accommodate Distortion (work conserving)	**Index Update Based-on Per Connection Parameter**	Delay-EDD Virtual Clock		Rate-controlled servers With Stand-by Queues	
		Index Update Based-on Reference Queueing Model	WFQ SCFQ		WF^2Q	

Fig. 20. Taxonomy of Service Disciplines

ferent delay bounds and bandwidths to different connections in a single server; (2) how the service discipline handles traffic distortions in a networking environment.

The first issue relates to the design of a single server. The objective of the allocation of delay bound and bandwidth is that, with a certain discipline, a connection can be guaranteed to receive a certain throughput, and each packet on that connection can be guaranteed to have a bounded delay. In addition to the scheduler, which is responsible for multiplexing packets from different connections and choosing the next packet to transmit, a server can also have a rate-controller. To provide different quality of services to different connections, a server needs to discriminate packets based on their performance requirements. Either a dynamic sorted priority queue or a static priority queue can be used for this purpose. In the case when the server consists of a static priority scheduler and no rate-controller, additional mechanisms are needed to ensure that packets at higher priority levels do not starve packets at lower priority levels. Toward this end, Stop-and-Go and HRR adopt non-work-conserving multi-level framing strategies. When compared to the more general rate-controlled service disciplines, mult-level framing suffers from a number of disadvantages.

The second issue concerns the interaction between different servers along the path traversed by the connection. Since the traffic pattern of each connection can be distorted inside the network due to load fluctuations, the server either needs to accommodate the distortion by buffering or control the distortion by regulating the traffic inside the network. Controlling traffic pattern distortion requires non-work-conserving disciplines, which can be implemented by either using a multi-level framing strategy or decoupling the server into a rate-controller and a scheduler. There are two classes of algorithms to control traffic pattern distortion: delay-jitter control, which maintains the same traffic characteristics at each switch as that at the previous switch, and rate-jitter control, which shapes the traffic according to a pre-specified traffic constraint function. All work-conserving disciplines use the sorted priority queue mechanism. This is not coincidental. Only a sorted priority queue has the flexibility to perform both functions of delay bound/bandwidth allocation and adjusting for traffic pattern distortions.

To provide guaranteed performance service, end-to-end delay bounds need to be provided in a networking environment on a per connection basis. Various analysis techniques have been developed. One solution is to analyze the worst-case local delay at each switch independently and bound the end-to-end delay of a connection by using the sum of the local delay bounds at all switches traversed by the connection. Alternatively, it has been observed that smaller end-to-end delay bounds can be obtained by taking into account the delay dependencies among successive switches traversed by the connection. In general, for both types of solutions, the traffic needs to be characterized on a per connection basis at each switch inside the network. For most of the proposed work-conserving disciplines, due to the difficulty of characterizing traffic inside the network, tight end-to-end delay bounds can be derived only for a restricted class of resource assignment strategies called rate-proportional assignments. With rate-proportional assignment, the allocation of delay bounds and bandwidth are coupled. For rate-controlled disciplines, since traffic is regulated inside the network, tight end-to-end delay bounds can be derived for general resource assignments. It has been shown in [19] that by properly setting parameters for regulators and local delay bounds for schedulers, rate-

controlled disciplines can always out-perform WFQ type of disciplines in terms of the number of connections that can be accepted.

Among the proposed algorithms, rate-controlled service disciplines [64], [19], which separate the server into a rate controller and a scheduler, exhibit the following distinct advantages: 1) simplified stability analysis, which allows tight end-to-end delay bounds to be derived for general resource assignments; 2) decoupling delay bound and bandwidth allocation without using the sorted priority queue; and 3) allowing heterogeneous servers with different schedulers and regulators to be used at different switches. While rate-controlled service disciplines are in general non-work-conserving, which has the additional advantage of requiring less buffer space within the network to prevent packet loss, they can be easily modified to be work-conserving by introducing a stand-by queue.

Although we have provided important insights into the issues and tradeoffs of designing service disciplines for integrated services networks, there are several important problems that remain unresolved and need to be addressed in future research. For example, it has been shown that tight end-to-end delay bounds can be derived under general resource assignments for rate-controlled service disciplines but can only be derived under rate-proportional resource assignments for most work-conserving disciplines other than those modified from rate-controlled servers. Future work should develop more advanced techniques to bound end-to-end delay under general resource assignments for FFQ-based work-conserving disciplines Also, how important is it to have general resource assignments? How much higher network utilization can be achieved with general resource assignments compared with rate-proportional resource assignments, and under what traffic mix conditions and network environments? We leave these questions for future research.

As a final note, we would like to point out that the focus the paper is on service disciplines for *guaranteed performance service*. Other services such as the predicted service and various types of best-effort services have different requirements, and there will be different tradeoffs in designing service disciplines for these services. For example, for the same resource assignment, WFQ and WF^2Q always provide identical end-to-end delay bounds for all connections. However, as discussed in [2] and Section III-B, the services that they provide or best-effort traffic can be quite different. Issues in designing service disciplines for network services other than the guaranteed performance service are beyond the scope of the paper.

VI. Acknowledgement

The author would like to thank Ed Knightly, Jon C.R. Bennett and anonymous referees for providing insightful comments that greatly improved the presentation of the paper.

References

[1] 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.

[2] J.C.R. Bennett and H. Zhang. WF^2Q: Worst-case fair weighted fair queueing. In *Proceedings of IEEE INFOCOM'96*, pages 120–128, San Francisco, CA, March 1996.

[3] P. Brady. A techniques for investigating on-off patterns in speech. *Bell System Technical Journal*, 44:1–22, January 1965.

[4] R. Brown. Calendar queues: A fast O(1) priority queue implementation for the simulation event set problem. *Communications of the ACM*, 31(10):1220–1227, October 1988.

[5] C. Chang. Stability, queue length, and delay of deterministic and stochastic queueing networks. *IEEE Transactions on Automatic Control*, 39(5):913–931, May 1994.

[6] H. Chao. Architecture design for regulating and scheduling user's traffic in ATM networks. In *Proceedings of ACM SIGCOMM'92*, pages 77–87, Baltimore, Maryland, August 1992.

[7] D. Clark, S. Shenker, and L. Zhang. Supporting real-time applications in an integrated services packet network: Architecture and mechanism. In *Proceedings of ACM SIGCOMM'92*, pages 14–26, Baltimore, Maryland, August 1992.

[8] R. Cruz. A calculus for network delay, part I : Network elements in isolation. *IEEE Transaction of Information Theory*, 37(1):114–121, 1991.

[9] R. Cruz. A calculus for network delay, part II : Network analysis. *IEEE Transaction of Information Theory*, 37(1):121–141, 1991.

[10] R. Cruz. Service burstiness and dynamic burstiness measures: A framework. *Journal of High Speed Networks*, 1(2):105–127, 1992.

[11] J. Davin and A. Heybey. A simulation study of fair queueing and policy enforcement. *Computer Communication Review*, 20(5):23–29, October 1990.

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

[13] D. Ferrari. Client requirements for real-time communication services. *IEEE Communications Magazine*, 28(11):65–72, November 1990.

[14] D. Ferrari. Real-time communication in an internetwork. *Journal of High Speed Networks*, 1(1):79–103, 1992.

[15] D. Ferrari, A. Banerjea, and H. Zhang. Network support for multimedia: a discussion of the Tenet approach. *Computer Networks and ISDN Systems*, 26(10):1167–1180, July 1994.

[16] D. Ferrari and D. 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.

[17] N. Figueira and J. Pasquale. An upper bound on delay for the virtualclock service discipline. *IEEE/ACM Transactions on Networking*, 3(4), April 1995.

[18] A. Fraser. Designing a public data network. *IEEE Communications Magazine*, 30(10):31–35, October 1991.

[19] L. Georgiadis, R. Guérin, and V. Peris. Efficient network QoS provisioning based on per node traffic shaping. In *IEEE INFOCOM'96*, San Francisco, CA, March 1996.

[20] S. 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.

[21] S. Golestani. A stop-and-go queueing framework for congestion management. In *Proceedings of ACM SIGCOMM'90*, pages 8–18, Philadelphia, PA, September 1990.

[22] S. Golestani. A self-clocked fair queueing scheme for broadband applications. In *Proceedings of IEEE INFOCOM'94*, pages 636–646, Toronto, CA, June 1994.

[23] P. Goyal, S. Lam, and H. Vin. Determining end-to-end delay bounds in heterogeneous networks. In *Proceedings of the 5th International Workshop on Network and Operating System Support For Digital Audio and Video*, pages 287–298, Durham, New Hampshire, April 1995.

[24] V. Jacobson. Congestion avoidance and control. In *Proceedings of ACM SIGCOMM'88*, pages 314–329, August 1988.

[25] R. Jain. Congestion control in computer networks: Issues and trends. *IEEE Network Magazine*, pages 24–30, May 1990.

[26] C. Kalmanek, H. Kanakia, and S. Keshav. Rate controlled servers for very high-speed networks. In *IEEE Global Telecommunications Conference*, pages 300.3.1 – 300.3.9, San Diego, California, December 1990.

[27] C. Kalmanek, S. Morgan, and R. C. Restrick. A high performance queueing engine for ATM networks. In *Proceedings of 14th International Switching Symposium*, Yokahama, Japan, October 1992.

[28] C. Kalmannek and R. Restrick. Xunet 2 queue module, October 1989. AT&T Bell Laboratories internal technical report.

[29] D. Kandlur, K. Shin, and D. Ferrari. Real-time communication in multi-hop networks. In *Proceedings of 11th International Conference on Distributed Computer Systems*, May 1991.

[30] M. Karol, M. Hluchyj, and S. Mogan. Input versus output queueing on a space-division packet switch. *IEEE Transactions on Communications*, 35(12):1347–1356, December 1987.

[31] S. Keshav. A control-theoretic approach to flow control. In *Proceedings of ACM SIGCOMM'91*, pages 3–15, Zurich, Switzerland, September 1991.

[32] L. Kleinrock. *Queueing Systems*. John Wiley and Sons, 1975.

[33] L. Kleinrock. *Queueing Systems, Volume 2: Computer Applications*. Wiley, 1976.

[34] E. Knightly, D. Wrege, J. Liebeherr, and H. Zhang. Fundamental limits and tradeoffs for providing deterministic guarantees to VBR video traffic. In *Proceedings of ACM Sigmetrics'95*, pages 275–286, Ottawa, CA, May 1995.

[35] E. Knightly and H. Zhang. Traffic characterization and switch utilization using deterministic bounding interval dependent traffic models. In *Proceedings of IEEE INFOCOM'95*, Boston, MA, April 1995.

[36] D. Knuth. *The Art of Computer Programming. Volume 3: Sorting and searching*. Addison-Wesley, 1975.

[37] J. Kurose. On computing per-session performance bounds in high-speed multi-hop computer networks. In *ACM Sigmetrics'92*, 1992.

[38] J. Kurose. Open issues and challenges in providing quality of service guarantees in high-speed networks. *ACM Computer Communication Review*, 23(1):6–15, January 1993.

[39] A. Lazar and C. Pacifici. Control of resources in broadband networks with quality of service guarantees. *IEEE Communication Magazine*, pages 66–73, October 1991.

[40] J. Liebeherr, D. Wrege, and D.Ferrari. Exact admission control for networks with bounded delay services. Technical Report CS-94-29, University of Virginia, Department of Computer Science, July 1994.

[41] C. Liu and J. Layland. Scheduling algorithms for multiprogramming in a hard real-time environment. *Journal of ACM*, 20(1):46–61, January 1973.

[42] S. Low. *Traffic Control in ATM Networks*. PhD dissertation, University of California at Berkeley, May 1992.

[43] B. Maglaris, D. Anastassiou, P. Sen, G. Karlsson, and J. Robbins. Performance models of statistical multiplexing in packet video communications. *IEEE Transaction on Communication*, 36(7):834–844, July 1988.

[44] M. Maresca, June 1993. Personal communication.

[45] R. Nagarajan, J. Kurose, and D. Towsley. Local allocation of end-to-end quality-of-service in high-speed networks. In *IFIP TC6 Task Group/WG6.4 International Workshop on Performance of Communication Systems*, pages 99–118, Martinique, January 1993.

[46] I. Nikolaidis and I. Akyildiz. Source characterization and statistical multiplexing in atm networks. Technical Report GIT-CC-92/24, College of Computing, Georgia Institute of Technology, Atlanta, GA 30332-0280, 1992.

[47] A. Parekh. *A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks*. PhD dissertation, Massachusetts Institute of Technology, February 1992.

[48] A. Parekh and R. Gallager. A generalized processor sharing approach to flow control - the single node case. In *Proceedings of the INFOCOM'92*, 1992.

[49] A. Parekh and R. Gallager. A generalized processor sharing approach to flow control in integrated services networks: The multiple node case. In *Proceedings of the INFOCOM'93*, pages 521–530, San Francisco, CA, March 1993.

[50] C. Parris, H. Zhang, and D. Ferrari. Dynamic management of guaranteed performance multimedia connections. *Multimedia Systems Journal*, 1:267–283, 1994.

[51] K. Ramakrishnan, D. Chiu, and R. Jain. Congestion avoidance in computer networks with a connectionless network layer. In *Proceedings of ACM SIGCOMM'88*, pages 303–313, Stanford, CA, August 1988.

[52] D. Saha, S. Mukherjee, and S.Tripathi. Multi-rate traffic shaping and end-to-end performance guarantees in ATM networks. In *Proceedings of 1994 International Conference on Network Protocols (ICNP'94)*, Boston, MA, October 94.

[53] J. Stankovic and K. Ramamritham. *Hard Real-Time Systems*. IEEE Computer Society Press, 1988.

[54] J. Turner. New directions in communications(or which way to the information age?). *IEEE Communication Magazine*, 24(10), October 1986.

[55] D. Verma. *Guaranteed Performance Communication in High Speed Networks*. PhD dissertation, University of California at Berkeley, November 1991.

[56] D. Verma, H. Zhang, and D. Ferrari. Guaranteeing delay jitter bounds in packet switching networks. In *Proceedings of Tricomm'91*, pages 35–46, Chapel Hill, North Carolina, April 1991.

[57] R. Wolff. *Stochastic Modeling and the Theory of Queues*. Prentice Hall, 1989.

[58] G. Xie and S. Lam. Delay guarantee of virtual clock server. *IEEE/ACM Transactions on Networking*, 3(4):683–689, December 1995.

[59] O. Yaron and M. Sidi. Calculating performance bounds in communication networks. In *Proceedings of IEEE INFOCOM'93*, pages 539–546, San Francisco, CA, April 1993.

[60] O. Yaron and M. Sidi. Performance and stability of communication networks via robust exponential bounds. *IEEE/ACM Transactions on Networking*, 1(3):372–385, June 1993.

[61] H. Zhang. Providing end-to-end performance guarantees using non-work-conserving disciplines. To appear in *Computer Communications: Special Issue on System Support for Multimedia Computing*.

[62] H. Zhang and D. Ferrari. Rate-controlled static priority queueing. In *Proceedings of IEEE INFOCOM'93*, pages 227–236, San Francisco, California, April 1993.

[63] 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.

[64] H. Zhang and D. Ferrari. Rate-controlled service disciplines. *Journal of High Speed Networks*, 3(4):389–412, 1994.

[65] H. Zhang and S. Keshav. Comparison of rate-based service disciplines. In *Proceedings of ACM SIGCOMM'91*, pages 113–122, Zurich, Switzerland, September 1991.

[66] H. Zhang and E. Knightly. Providing end-to-end statistical performance guarantees with interval dependent stochastic models. In *ACM Sigmetrics'94*, pages 211–220, Nashville, TN, May 1994.

[67] L. Zhang. Virtual clock: A new traffic control algorithm for packet switching networks. In *Proceedings of ACM SIGCOMM'90*, pages 19–29, Philadelphia, PA, September 1990.

[68] Z. Zhang, D. Towsley, and J. Kurose. Statistical analysis of generalized processor sharing scheduling discipline. In *Proceedings of ACM SIGCOMM'94*, London, UK, August 1994.

[69] Q. Zheng and K. Shin. On the ability of establishing real-time channels in point-to-point packet-switching networks. *IEEE Transactions on Communications*, pages 1096–1105, March 1994.