Generalized Guaranteed Rate Scheduling Algorithms: A Framework

Pawan Goyal and Harrick M. Vin

Abstract—

In this paper, we define a class of generalized Guaranteed Rate (GR) scheduling algorithms that includes algorithms which allocate variable rate to packets of a flow. We define work conserving generalized Virtual Clock, Packet-by-Packet Generalized Processor Sharing and Self Clocked Fair Queuing scheduling algorithms that can allocate variable rate to the packets of a flow. We also define scheduling algorithms suitable for servers where packet fragmentation may occur. We demonstrate that if a class of rate controllers is employed for a flow in conjunction with any scheduling algorithm in GR, then the resulting non-work conserving algorithm also belongs to GR. This leads to the definition of several non-work conserving algorithms.

We then present a method for deriving the delay guarantee of a network of servers when: (1) different rates are allocated to packets of a flow at different servers along the path and the bottleneck server for each packet may be different, and (2) packet fragmentation and/or reassembly may occur. This delay guarantee enables a network to provide various service guarantees to flows conforming to any specification. We illustrate this by utilizing delay guarantee to derive delay bounds for flows conforming to Leaky Bucket, Exponentially Bounded Burstiness, and Flow Specification. Our method for determining these bounds is valid in internetworks and leads to tighter results.

1 Introduction

1.1 Motivation

Due to the inherent characteristics of audio and video, many multimedia applications (e.g., audio and video conferencing, multimedia information retrieval, etc.) require the network to provide a wide range of Quality of Service (QoS) guarantees (with respect to bandwidth, packet delay, delay jitter and loss). To enable a network to provide such guarantees, sources specify their traffic characteristics. The network, on the other hand, provides QoS guarantees by reserving and scheduling network resources in accordance with the specifications. The traffic specification and the QoS guarantees constitute a 'contract' between the network and a source: the network guarantees that, as long as the source conforms to its traffic specification, its QoS requirements will be met. Mechanisms for providing these guarantees must address:

- *Heterogeneity in source traffic characteristics*: The traffic characteristics of multimedia sources differ significantly. For example, whereas many audio applications require constant bit rate, resource requirement of applications transmitting Variable Bit Rate (VBR) compressed video sequences varies significantly over time.
- *Heterogeneity in the network characteristics*: Current networks are, and future networks will remain, heterogeneous along several dimensions. For example, in a large network consisting of several autonomous domains, switches may employ different scheduling algorithms. Furthermore, due to the variation in the size of data transmission unit in internetwork environments (e.g., an internetwork consisting of ATM, FDDI, ethernet, and token ring), packet fragmentation and/or reassembly may also occur in the network.

In such heterogeneous environments, the techniques for providing QoS guarantees should be flexible enough to accommodate: (1) a wide range of traffic specifications, (2) variable rate allocations for a source, (3) a variety of scheduling algorithms at the switches, and (4) internetworking environments (in which fragmentation and reassembly may occur). A framework for meeting these requirements is the subject matter of this paper.

1.2 Relation to Previous Work

The sequence of packets transmitted by a source is referred to as a *flow* [26]. Each packet within a flow is serviced by a sequence of servers (or switching elements) along the path from the source to the destination in the network. To provide guaranteed QoS to flows, several rate-based scheduling algorithms have been proposed in the literature [9, 18, 25, 26, 27] (see [23] for an in-depth exposition and of the algorithms and their characteristics). Most of these algorithms can be classified as being either work conserving or non-work conserving and either allocating only rate or separating rate and delay allocation [23]. A fundamental characteristic of most of these scheduling algorithms is that they do not permit the rates allocated to flows to vary over time. The ability to vary rate allocation of a flow, however, is highly desirable to efficiently transmit VBR video streams. To address this requirement, a non-work conserving scheduling algorithm (referred to as Burst Scheduling), that combines the Virtual Clock scheduling algorithm with the concept of an *active flow*, was proposed in [15]. Work conserving algorithms that achieve the same objective, however, have not received much attention.

In addition to a variety of scheduling algorithms, to provide QoS guarantees, a wide range of traffic specifications

The authors are with the Distributed Multimedia Computing Laboratory, Department of Computer Sciences, The University of Texas at Austin, Taylor Hall 2.124, Austin, TX 78712. Their e-mail addresses are {pawang,vin}@cs.utexas.edu, and their URL is http://www.cs.utexas.edu/users/dmcl.

have also been proposed in the literature [15, 18, 22, 25]. However, most of the techniques for providing QoS guarantees have only investigated specific combinations of traffic specification and scheduling algorithms [3, 4, 15, 18, 22](see [23] for description of the analysis methods). This limitation has been partially addressed in [2, 8, 10, 24]. A general method for analysis of a network of servers employing heterogeneous non-work conserving algorithms has been developed in [24]. This approach has been generalized in [8] to analyze a network employing a larger class of work conserving as well as non-work conserving algorithms. A class of Guaranteed Rate scheduling algorithms (which was shown to include Virtual Clock, Packet-by-Packet Generalized Processor Sharing (PGPS), and Self Clocked Fair Queuing (SCFQ)), and a method for determining end-to-end delay bounds for sources with different characteristics in a network of Guaranteed Rate servers has been presented in [10]. However, none of these approaches analyze a network of servers when variable rate may be allocated to the packets of a flow and packet fragmentation and/or reassembly may occur.

1.3 Research Contributions of This Paper

In this paper, we generalize the class of Guaranteed Rate scheduling algorithms to include scheduling algorithms that may allocate variable rate to the packets of a flow. The class of generalized Guaranteed Rate (GR) scheduling algorithms guarantee a deadline (referred to as *delay guarantee*) to a packet based on its expected arrival time. The delay guarantee of these algorithms is independent of a traffic specification and the behavior of other flows at the server. We define work conserving generalized Virtual Clock, PGPS, and SCFQ scheduling algorithms that can allocate variable rate to the packets of a flow, and show that they belong to GR. To prove that many scheduling algorithms belong to GR, we employ a proof methodology (similar to that in [12, 18]) in which we first show that a preemptive equivalent of the algorithm belongs to GR, and then utilize a relationship (also derived in this paper) between preemptive and non-preemptive scheduling algorithms to show that the non-preemptive algorithm belongs to GR. This methodology leads to the definition of several scheduling algorithms in GR that are suitable for servers at which packet fragmentation may occur. The algorithms that we define for such servers reduce computational complexity as well as delay incurred by packets. Finally, we demonstrate that if a rate control element is employed in conjunction with any scheduling algorithm in GR on a *per flow* basis, the resulting non-work conserving algorithm also belongs to GR (this leads to the definition of several scheduling algorithms) and enables a network to support flows with different delay-jitter requirements.

The delay guarantee of the GR class enables a single server to provide service guarantees to flows conforming to any specification. To enable a sequence of servers to provide similar service guarantees, we generalize the method presented in [10] to derive the delay guarantee of a network of servers, each of which employs a scheduling algorithm in the GR class, when: (1) different rates are allocated to the packets of a flow at different servers along the path and the bottleneck server for each packet may be different, and (2) packet fragmentation and/or reassembly may occur in the network. We utilize the delay guarantee of a network of servers to obtain an upper bound on end-to-end delay. We illustrate the end-to-end delay bound computation for flows conforming to Leaky Bucket, Exponentially Bounded Burstiness, and Flow Specification [15].

The rest of the paper is organized as follows: In Section 2, we define the class of generalized GR scheduling algorithms. The method for deriving delay guarantee for a network of servers is presented in Section 3. We utilize the delay guarantee to derive end-to-end delay bounds in Section 4. Finally, Section 5 summarizes our results.

2 Generalized Guaranteed Rate Scheduling Algorithms

Many of the scheduling algorithms proposed in the literature guarantee a deadline (referred to as *delay guarantee* [21]) to a packet of a flow based on its *expected arrival time*. In [10], we defined the delay guarantee of a packet by associating a *guaranteed rate clock* value with each packet. The class of guaranteed rate scheduling algorithms was defined to consist of algorithms that guarantee that a packet would be transmitted by its guaranteed rate clock value plus some constant. However, since the guaranteed rate clock value was defined based on the constant rate associated with a flow, the guaranteed rate class did not include scheduling algorithms that assign variable rate to the packets of a flow. We generalize the definition of the guaranteed rate class to include such scheduling algorithms.

The generalized guaranteed rate class (hereafter referred to as GR) is defined based on the generalized guaranteed rate clock value (hereafter referred to as guaranteed rate clock (GRC) value) of a packet. To define the guaranteed rate clock values, let p_f^j and l_f^j denote the j^{th} packet of flow f and its length, respectively, and let $r_f^{j,i}$ (bits/s) be the rate allocated to packet p_f^j at server i (observe that each packet may be allocated a different rate). Additionally, let $A^i(p_f^j)$ denote the arrival time of packet p_f^j at server i. Then, guaranteed rate clock value for packet p_f^j at server i, denoted by $GRC^i(p_f^j, r_f^{j,i})$, is given as:

$$GRC^i(p_f^j, r_f^{j,i}) = \max\{A^i(p_f^j), GRC^i(p_f^{j-1}, r_f^{j-1,i})\} + \frac{l_f^j}{r_f^{j,i}} \tag{1}$$

where $GRC^i(p_f^0, r_f^{0,i}) = 0$ [1]. The first term on the right hand side of (1) can be interpreted as the expected arrival

[1]Since link-level header length may vary in an internetwork, the length of a packet does not include the link-level header. To accommodate the effect of link-level headers, we require the actual rate allocation, denoted by $\overline{r}_f^{j,i}$, at server i to be such that $\frac{l_f^j}{r_f^{j,i}} = \frac{l_f^j + \overline{l}_f^{j,i}}{\overline{r}_f^{j,i}}$

time and the second term as the deadline of a packet. This definition of guaranteed rate clock is based on virtual clock [26]. Guaranteed rate clock value and virtual clock value of a packet are same when constant rate is allocated to the packets of a flow. We have coined a new term for the virtual clock concept to differentiate between the general concept of virtual clock and as it relates to the Virtual Clock scheduling algorithm. We use the guaranteed rate clock value of a packet to define the class of GR scheduling algorithms as follows.

Definition 1 A scheduling algorithm at server i belongs to class GR for flow f if it guarantees that packet p_f^j will be transmitted by $GRC^i(p_f^j, r_f^{j,i}) + \beta^i$ where β^i is a constant which depends on the scheduling algorithm and the server.

As is evident from the definition, two key properties of the class of GR scheduling algorithms are: (1) they provide a delay guarantee for a source independent of the behavior of other sources in the network, and thereby isolate the sources; and (2) the delay guarantee is independent of a traffic characterization. Whereas isolation of sources enables a network to provide stronger guarantees and is highly desirable, especially in large heterogeneous networks where sources may be malicious [5, 15], independence of delay guarantee from traffic characterization enables a server to provide various QoS guarantees to flows conforming to any specification.

In the following subsections, we show that many of the work conserving as well as non-work conserving scheduling algorithms that either allocate only rate or separate rate and delay allocation belong to GR. To show that a scheduling algorithm belongs to GR, we would be required to prove a bound on the departure time of a packet. As observed in [5, 12], it is typically easier to bound the departure time of a packet in *preemptive* scheduling algorithms. Hence, even though packet scheduling algorithms are inherently *non-preemptive* in nature, to show that a scheduling algorithm belongs to GR, we employ a proof methodology, similar to the methodology in [12, 18], in which we first prove a bound on the departure time of a packet in preemptive scheduling algorithm, and then use a relationship between the departure times of a packet in equivalent preemptive and non-preemptive scheduling algorithm. In what follows, we establish a relationship between preemptive and non-preemptive scheduling algorithms. Due to lack of space, the proofs of the results are omitted and can be found in [11].

where $\bar{l}_f^{j,i}$ is the length of link-level header at server i. Consequently, guaranteed rate clock value of a packet with and without the link-level headers is the same. Hence, in this paper we will always refer to the guaranteed rate clock value of a packet without the link-level headers. Note that an upper bound on overhead due to link-level headers can be derived using a lower bound on packet lengths.

2.1 Preemptive and Non-Preemptive Scheduling

Many of the scheduling algorithms that we will consider, assign a priority to a packet on its arrival and then schedule the packets in the priority order. In these scheduling algorithms, a packet with higher priority may arrive after a packet with lower priority has been scheduled. In *non-preemptive* scheduling algorithms, transmission of a lower priority packet is not preempted even after a higher priority packet arrives. Consequently, such algorithms ensure that the packet in service is the packet with the highest priority only after the transmission of every packet. On the other hand, a *preemptive* scheduling algorithm always ensures that the packet in service is the packet with the highest priority by possibly preempting the transmission of a packet with lower priority. In contrast to preemptive and non-preemptive algorithms, a *partially-preemptive* scheduling algorithm ensures that the packet in service is the packet with the highest priority after the transmission of every fragment of a packet.

Partially-preemptive algorithms would be helpful in defining scheduling algorithms suitable for servers where fragmentation may occur. Furthermore, non-preemptive algorithms are a subset of partially-preemptive algorithms. Hence, in Theorem 1 we establish a relationship between *equivalent* preemptive and partially-preemptive scheduling algorithms. A partially-preemptive scheduling algorithm is considered *equivalent* to a preemptive algorithm if the priority assigned to all the packets is the same in both the algorithms.

Theorem 1 Let PS be a work conserving preemptive scheduling algorithm that assigns priority $\hat{L}_{PS}(p^j)$ to packet p^j, schedules packets in the increasing order of priority and guarantees that regardless of the arrival pattern of other packets, packet p^j will be transmitted by $\hat{L}_{PS}(p^j)$. Let PPS be its equivalent partially-preemptive scheduling algorithm, then

$$L_{PPS}(p^j) - \hat{L}_{PS}(p^j) \leq \frac{\hat{l}_{max}}{C} \tag{2}$$

where $L_{PPS}(p^j)$ denotes the time a packet leaves the server when PPS scheduling algorithm is employed. Also, $\hat{l}_{max}$ is the maximum length of a packet fragment and C is the capacity of the server.

Though Theorem 1 is superficially similar to Theorem 1 of [18], as illustrated in [7], Theorem 1 of [18] is not applicable to some of the algorithms (for example, Virtual Clock) that we will be considering.

We now show that, using Theorem 1 wherever applicable, most of the work conserving and non-work conserving scheduling algorithms that either allocate only rate or separate rate and delay allocation belong to GR.

2.2 Work Conserving Algorithms

2.2.1 Variable Rate Allocation Algorithms

In this section, we define work conserving generalized Virtual Clock, PGPS, and SCFQ algorithms that allocate variable rate to packets of a flow and show that all of these scheduling algorithms belong to GR. These algorithms have different delay and fairness properties as well as implementation complexity, and hence demonstrate that the GR class is broad (an exposition of these algorithms along these dimensions can be found in [9, 18]). We will analyze the algorithms assuming that rate allocation may vary with every packet. However, a particular network architecture may choose to allocate variable rate for every message [15], in response to change in reservation [13], or employ some other policy. These policy issues are the subject of future investigation and beyond the scope of this paper.

Virtual Clock We define generalized Virtual Clock (VC) scheduling algorithm analogous to the Virtual Clock algorithm [26]. The generalized VC algorithm is defined as follows:

1. On arrival at server i, packet p_f^j that is assigned rate $r_f^{j,i}$ is stamped with virtual clock value, denoted by $VC^i(p_f^j, r_f^{j,i})$, computed as:

$$VC^i(p_f^j, r_f^{j,i}) = \max\{A^i(p_f^j), VC^i(p_f^{j-1}, r_f^{j-1,i})\} + \frac{l_f^j}{r_f^{j,i}}$$

where $VC^i(p_f^0, r_f^{0,i}) = 0$.

2. Packets are serviced in increasing order of the virtual clock value.

We show that generalized VC belongs to GR by first proving a bound on the departure time of a packet in preemptive VC. To do so, let us first define $R_f^i(t)$ for flow f as follows:

$$R_f^i(t) = \begin{cases} r_f^{j,i} & \text{if } \exists j \ni \left(A^i(p_f^j) \le t\right) \wedge \\ & \left(VC^i(p_f^{j-1}, r_f^{j-1,i}) < t \le VC^i(p_f^j, r_f^{j,i})\right) \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

Let S be the set of flows served by server i. Then server i with capacity C^i is defined to have exceeded its capacity at time t if $\sum_{n \in S} R_n^i(t) > C^i$. The following theorem bounds the departure time of a packet in preemptive VC.

Theorem 2 If a server's capacity is not exceeded, then the departure time of packet p_f^j in preemptive VC, denoted by $L_{P-VC}^i(p_f^j)$, is at most $VC^i(p_f^j, r_f^{j,i})$, i.e., $L_{P-VC}^i(p_f^j) \le VC^i(p_f^j, r_f^{j,i})$.

Since preemptive VC algorithm is work conserving and guarantees that packet p_f^j will be transmitted by $VC^i(p_f^j, r_f^{j,i})$ for any arrival pattern of packets, using Theorems 1 and 2 we conclude that the departure time of packet p_f^j in generalized Virtual Clock, denoted by $L_{VC}^i(p_f^j)$, is at most $VC^i(p_f^j, r_f^{j,i}) + \frac{l_{max}^i}{C^i}$, i.e., $L_{VC}^i(p_f^j) \le VC^i(p_f^j, r_f^{j,i}) + \frac{l_{max}^i}{C^i}$ where C^i is the capacity of the server and l_{max}^i is the maximum length of a packet serviced by server i. Since the equations for virtual clock and guaranteed rate clock are the same, we conclude that generalized Virtual Clock scheduling algorithm belongs to GR for flow f with $\beta^i = \frac{l_{max}^i}{C^i}$.

Observe that generalized Virtual Clock is work conserving and permits variable rate to be allocated to the packets of a flow as long as a servers capacity is not exceeded. This is in contrast to the non-work conserving Burst Scheduling algorithm presented in [15] in which Virtual Clock scheduling algorithm has been employed to allocate variable rate by defining the notion of an *active flow*. A flow has a constant rate allocated to it as long as it is active. Rate assignment of a flow is changed only after it makes a transition from active to inactive state. Hence, to allocate variable rate, a *flow regulator* which enforces such transitions is required to be implemented at each server. Additionally, a server is required to timestamp a packet with the difference between the packets deadline and actual departure time. Generalized Virtual Clock does not have any such requirements, and hence reduces the implementation complexity.

Packet-by-Packet Generalized Processor Sharing The Packet-by-Packet Generalized Processor Sharing scheduling algorithm is a practical realization of Generalized Processor Sharing(GPS) service discipline [18]. We first show that GPS belongs to GR and then show that a generalized virtual time implementation of PGPS belongs to GR.

In GPS, each flow f is assigned a constant ϕ_f^i at server i. To allocate variable rate to packets of a flow, we associate $\phi_f^{j,i}$ with packet p_f^j. From the definition of GPS we know that at time t, packet p_f^j will be serviced at the rate of $\frac{\phi_f^{j,i} C^i}{\sum_{k \in b^i(t)} \phi_k^{a,i}}$ where packet p_k^a of flow k is in service at time t, $b^i(t)$ is the set of backlogged flows at GPS server i at time t, and C^i is the capacity of the server. Hence, we define a GPS server to have assigned rate $r_f^{j,i}$ to packet p_f^j if $\frac{\phi_f^{j,i} C^i}{\sum_{k \in b^i(t)} \phi_k^{a,i}} \ge r_f^{j,i}$ as long as the packet is in service. Therefore, if packet p_f^j is assigned rate $r_f^{j,i}$, it is served at least at rate $r_f^{j,i}$. Since GPS serves packets of a flow in FCFS order, we get:

$$L_{GPS}^i(p_f^j) \le \max\{A^i(p_f^j), L_{GPS}^i(p_f^{j-1})\} + \frac{l_f^j}{r_f^{j,i}} \quad j \ge 1 \tag{4}$$

Let $L_{GPS}^i(p_f^0) = GRC^i(p_f^0, r_f^{0,i}) = 0$. From (4) and (1), it can be shown that

$$L_{GPS}^i(p_f^j) \le GRC^i(p_f^j, r_f^{j,i}) \tag{5}$$

Hence, GPS belongs to GR. We now define a virtual time implementation of packet-by-packet GPS which is a generalization of the implementation in [18]. Let $v^i(t)$ be the virtual time associated with server i at time t. Let $v^i(0) = 0$ and $v^i(t)$ not change when no packet is backlogged. Otherwise, define $v^i(t)$ as:

$$\frac{dv^i(t)}{dt} = \frac{C^i}{\sum_{k \in b^i(t)} \phi_k^{a,i}} \quad (6)$$

Define finish time of packet p_f^j, denoted by $F^i(p_f^j, \phi_f^{j,i})$ as:

$$F^i(p_f^j, \phi_f^{j,i}) = \max\{v^i(A^i(p_f^j)), F^i(p_f^{j-1}, \phi_f^{j-1,i})\} + \frac{l_f^j}{\phi_f^{j,i}} \quad (7)$$

where $F^i(p_f^0, \phi_f^{0,i}) = 0$. Using Theorem 1 of [18], it has been shown in [11] that $L_{PGPS}^i(p_f^j) \leq L_{GPS}^i(p_f^j) + \frac{l_{max}^i}{C^i}$. Thus, from (5), we get $L_{PGPS}^i(p_f^j) \leq GRC^i(p_f^j, r_f^{j,i}) + \frac{l_{max}^i}{C^i}$. Hence, PGPS scheduling algorithm belongs to GR for flow f with $\beta^i = \frac{l_{max}^i}{C^i}$.

Self Clocked Fair Queuing The Self Clocked Fair Queuing scheme, proposed in [9], was designed to facilitate the implementation of a fair queuing scheme in broadband networks. We define a generalized SCFQ algorithm which can allocate variable rate to packets of a flow analogous to SCFQ as follows:

1. On arrival, a packet p_f^j is stamped with service tag $F^i(p_f^j, r_f^{j,i})$, computed as:

   $$F^i(p_f^j, r_f^{j,i}) = \max\{v^i(A^i(p_f^j)), F^i(p_f^{j-1}, r_f^{j-1,i})\} + \frac{l_f^j}{r_f^{j,i}}$$

   where $F^i(p_f^0, r_f^{j,0}) = 0$.
2. The server virtual time at time t, $v^i(t)$, is defined to be equal to the service tag of the packet in service at time t. $v^i(t) = t$ when the server is idle.
3. Packets are serviced in increasing order of their service tags.

Define $R_f^i(v)$ for flow f as follows:

$$R_f^i(v) = \begin{cases} r_f^{j,i} & \text{if } \exists j \ni \left(v^i(A^i(p_f^j)) \leq v\right) \wedge \\ & \left(F^i(p_f^{j-1}, r_f^{j-1,i}) < v \leq F^i(p_f^j, r_f^{j,i})\right) \\ 0 & \text{otherwise} \end{cases}$$

Let S be the set of flows served by server i. Then server i with capacity C^i is defined to have exceeded its capacity at virtual time v if $\sum_{n \in S} R_n^i(v) > C^i$. The following theorem proves that generalized SCFQ algorithm also belongs to the class of GR scheduling algorithms.

Theorem 3 If the server's capacity is not exceeded, then the departure time of packet p_f^j in SCFQ, denoted by $L_{SCFQ}^i(p_f^j)$, is given by

$$L_{SCFQ}^i(p_f^j) \leq GRC^i(p_f^j, r_f^{j,i}) + \sum_{n \in S \wedge n \neq f} \frac{l_n^{max}}{C^i}$$

where l_n^{max} is the maximum length for packets in flow n.

Hence, SCFQ scheduling algorithm belongs to GR for flow f with $\beta^i = \sum_{n \in S \wedge n \neq f} \frac{l_n^{max}}{C^i}$.

2.2.2 Delay Allocation Algorithms

Though several algorithms which allocate rate have been proposed, Delay Earliest Due Date (Delay EDD) is the only work conserving scheduling algorithm that separates delay and rate allocation in a networking environment [2]. In this section, we show that Delay EDD also belongs to GR.

It is not known whether Delay EDD can separate delay and rate allocation while assigning variable rate to packets of a flow. Hence, we assume that rate r_f^i is assigned to all packets of a flow. Furthermore, in Delay EDD, the length of the packets is assumed to be the same, i.e., $l_f = l_f^j$. Delay EDD assigns a deadline to a packet on its arrival and schedules packets in increasing order of deadlines. Deadline of packet p_f^j, denoted by $D^i(p_f^j)$, is computed as:

$$D^i(p_f^j) = \max\{A^i(p_f^j), GRC^i(p_f^{j-1}, r_f^i)\} + d_f^i \quad (8)$$

where d_f^i is the delay bound for flow f at server i.

It was shown in [6, 27] that if certain schedulability conditions are met and the minimum inter-arrival time of packets is at least $\frac{l_f}{r_f^i}$, then a packet would depart by $D^i(p_f^j)$. However, in a networking environment even if the minimum inter-arrival time is at least $\frac{l_f}{r_f^i}$ at the network entry point, it may become smaller than $\frac{l_f}{r_f^i}$ at a server which is downstream on the path of a flow. This problem was addressed in [14, 27] by requiring the clocks of the servers to be synchronized. We demonstrate that this is an unnecessary restriction by proving in Theorem 4 that regardless of the inter-arrival time of packets, preemptive Delay EDD guarantees that packet p_f^j will be transmitted by $D^i(p_f^j)$. We then use this property to show that Delay EDD belongs to GR.

Theorem 4 If S is the set of flows serviced by the server and

$$\forall t > 0: \quad \sum_{n \in S} \max\left\{0, \left\lceil \frac{(t - d_n^i)r_n}{l_n} \right\rceil \frac{l_n}{C^i}\right\} \leq t \quad (9)$$

then the departure time of packet p_f^j in preemptive Delay EDD, denoted by $L_{P-EDD}^i(p_f^j)$, is at most $D^i(p_f^j)$, i.e., $L_{P-EDD}^i(p_f^j) \leq D^i(p_f^j)$.

Since Delay EDD guarantees that packet p_f^j will be transmitted by $D^i(p_f^j)$ regardless of the arrival pattern of the packets, from Theorems 1 and Theorem 4, we conclude that the time at which transmission of packet p_f^j is completed in Delay EDD, denoted by $L_{EDD}^i(p_f^j)$, is at most

[2]Consistent Relative Session Treatment (CRST) rate assignment has been used in PGPS networks to separate rate and delay allocation. However, since this rate assignment requires all the flows to conform to Leaky Bucket specification, we do not consider it further.

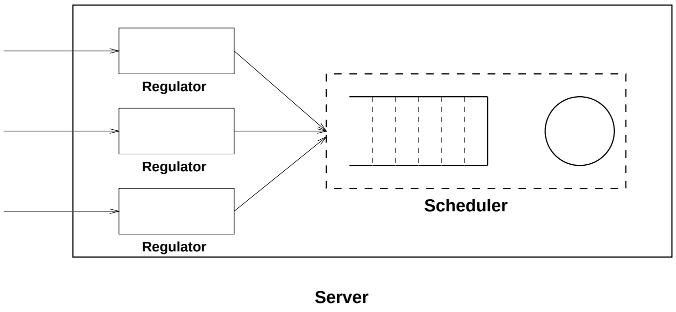

Figure 1: Rate Controlled Service Disciplines

$D^i(p_f^j) + \frac{l^i_{max}}{C^i}$, i.e., $L^i_{EDD}(p_f^j) \leq D^i(p_f^j) + \frac{l^i_{max}}{C^i}$. To observe that Delay EDD belongs to GR, rewrite (8) as:

$$D^i(p_f^j) = \max\{A^i(p_f^j), GRC^i(p_f^{j-1}, r_f^i)\} + \frac{l_f}{r_f^i} - (\frac{l_f}{r_f^i} - d_f^i)$$

Since $\max\{A^i(p_f^j), GRC^i(p_f^{j-1}, r_f^i)\} + \frac{l_f}{r_f^i} = GRC^i(p_f^j, r_f^i)$, we conclude that Delay EDD belongs to GR with $\beta^i = \frac{l^i_{max}}{C^i} - (\frac{l_f}{r_f^i} - d_f^i)$. If schedulability conditions for non-preemptive Delay EDD are used, then one can similarly show that β^i is given as $\beta^i = -(\frac{l_f}{r_f^i} - d_f^i)$.

2.3 Non-Work Conserving Algorithms

A general framework for reasoning about the end-to-end performance guarantee of a class of non-work conserving algorithms termed *Rate Controlled Service Disciplines* has been presented in [24]. In this section, we show that rate controlled service disciplines also belong to the GR class.

Rate controlled service disciplines consist of a rate regulator and a scheduler (see Figure 1). The rate regulator ensures that the traffic entering the scheduler conforms to a negotiated traffic specification and the scheduler guarantees that each packet of flow f would experience a maximum delay of d_f^i. In such disciplines, different types of rate regulators may be employed. Since these disciplines have been studied predominantly for constant rate allocation (see section 2.4.2 for non-work conserving disciplines that allocate variable rate), a common characteristic of most of the rate controllers is that they do not delay packets more than necessary to enforce the average rate. Hence, if r_f^i is the rate assigned to flow f and l_f the length of packets, then the time at which a packet p_f^j departs the rate controller, denoted by $L^i_{RC}(p_f^j, r_f^i)$, is given as:

$$L^i_{RC}(p_f^j, r_f^i) \leq \max\{A^i(p_f^j), GRC^i(p_f^{j-1}, r_f^i)\} + \gamma^i \qquad (10)$$

where γ^i is a constant for a rate controller. Since the scheduler guarantees a maximum delay of d_f^i to each packet, the time that packet p_f^j departs server i which employs a rate controlled service discipline, denoted by $L^i_{RCSD}(p_f^j)$, is given as:

$$\begin{aligned} L^i_{RCSD}(p_f^j) &\leq \max\{A^i(p_f^j), GRC^i(p_f^{j-1}, r_f^i)\} + \gamma^i + d_f^i \\ &\leq \max\{A^i(p_f^j), GRC^i(p_f^{j-1}, r_f^i)\} + \frac{l_f}{r_f^i} \\ &\quad -(\frac{l_f}{r_f^i} - \gamma^i - d_f^i) \end{aligned}$$

Since $\max\{A^i(p_f^j), GRC^i(p_f^{j-1}, r_f^i)\} + \frac{l_f}{r_f^i} = GRC^i(p_f^j, r_f^i)$, we conclude that rate controlled service disciplines that employ rate regulators consistent with (10) also belong to the GR class with $\beta^i = -(\frac{l_f}{r_f^i} - \gamma^i - d_f^i)$. The rate regulator for Rate Controlled Static Priority queuing is consistent with (10) and hence belongs to GR.

2.4 Packet Fragmentation and Rate Control

In heterogeneous networks, packets may be fragmented. Furthermore, to reduce the buffer requirement in the network, some flows may require rate controllers to be employed on the path. We now consider: (1) scheduling algorithms suitable for servers where packet fragmentation may occur, and (2) the effect of employing rate controller for a flow on the delay guarantee.

2.4.1 Scheduling in Presence of Fragmentation

Consider a server that can receive packets of size larger than it can transmit [3]. In such a case, the server has to fragment a packet before transmitting it. A fragmented packet can be scheduled for transmission by the server in various ways:

- The server can compute a priority for each packet fragment and hence schedule each fragment individually.
- The server can compute a priority for each packet and schedule each packet.

Whereas scheduling each packet fragment increases the computation overhead, from Theorem 1 we know that scheduling packets and disabling preemption of a packet transmission increases the maximum delay incurred. Hence, a scheduling algorithm that minimizes computational overhead while minimizing delay incurred due to non-preemption is desirable. Observe that since the transmission unit is a packet fragment, even if a server schedules packet, it can allow packet preemption to occur after the transmission of every packet fragment, i.e., it can schedule the packets in a *partially-preemptive* manner. This would simultaneously reduce the computational overhead and the delay incurred by the packets. Partially-preemptive equivalents of the generalized Virtual Clock, Delay EDD and PGPS can be defined using the definition of partially-preemptive scheduling algorithms. Also using Theorem 1, Theorem 2 and Theorem 4, we know that partially preemptive generalized Virtual Clock and Delay EDD also belong to GR

[3] In Internet packet fragmentation and reassembly can be avoided at the network layer by using path Maximum Transmission Unit (MTU) discovery [16]. However, fragmentation of scheduling entity, which can be smaller than the network level packet, cannot be avoided by such a method in an Internet consisting of ATM and FDDI subnetworks.

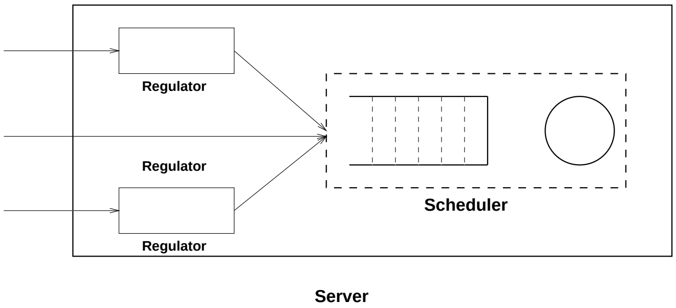

Figure 2: Rate controller employed for a subset of flows

with β^i being derived by substituting $\frac{l^i_{max}}{C^i}$ with $\frac{\hat{l}^i_{max}}{C^i}$ where $\hat{l}^i_{max}$ is the maximum length of a packet fragment served by server i. Similarly, Theorem 1 of [18] can be generalized in a manner analogous to Theorem 1 to show that partially-preemptive PGPS also belongs to GR with $\beta^i = \hat{l}^i_{max}$.

2.4.2 Effect of Rate Control

It has been observed that if a network employs a work conserving scheduling algorithm, then the burstiness of a flow and hence buffer requirement increases at the downstream nodes. Several non-work conserving scheduling algorithms have been proposed to address this limitation [24]. Though non-work conserving algorithms reduce the buffer requirement in the network, they increase the average delay for all the flows. Even though this may be desirable for some flows, it may not be desirable for other flows. We address this limitation of non-work conserving algorithms by employing rate controllers on a per flow basis (see Figure 2). However, if arbitrary rate controllers are employed, then they may increase not only the average delay, but also the worst case delay for the flows for which they are employed. To avoid increase in the worst case delay, we require that if a rate controller is employed for flow f at the server, then it should guarantee that departure time of packet p^j_f from the rate controller, denoted by $L^i_{RC}(p^j_f, r^{j,i}_f)$, is given as:

$$L^i_{RC}(p^j_f, r^{j,i}_f) \leq \max\{A^i(p^j_f), GRC^i(p^{j-1}_f, r^{j-1,i}_f)\} \quad (11)$$

In the special case of $r^i_f = r^{j,i}_f$, this definition captures the characteristics of rate control elements like leaky bucket. Let such a rate control element be employed at server i for flow f. Theorem 5 demonstrates that such a rate control element does not change the bound on the departure time of a packet when the scheduling algorithm employed is in GR.

Theorem 5 If a rate control element that satisfies (11) is employed at server i for flow f , then $GRC^i(p^j_f, r^{j,i}_f) = \widehat{GRC}^i(p^j_f, r^{j,i}_f)$ where $GRC^i(p^j_f, r^{j,i}_f)$ is the guaranteed rate clock value at the scheduler when no rate controller is employed for the flow and $\widehat{GRC}^i(p^j_f, r^{j,i}_f)$ is the value when a rate controller is employed.

Theorem 5 demonstrates that if a scheduling algorithm belongs to GR for a flow, then the equivalent non-work conserving algorithm obtained by employing any rate controller element satisfying (11) also belongs to GR. Observe that this defines non-work conserving equivalents of generalized Virtual Clock, PGPS, and SCFQ algorithms. It also defines a scheduling algorithm that combines Delay EDD and a variant of Jitter EDD.

2.5 Discussion

Several algorithms, other than those considered in the previous sections, also belong to GR. For example, since Frame-based Fair Queuing [20] and Worst-case Fair Weighted Fair Queuing [1] also guarantee that, regardless of behaviour of other flows in the network, the bound on departure time of a packet is the same as in PGPS, they also belong to GR. Similarly, since Deficit Round Robin (DRR [19]) also isolates flows and guarantees a minimum throughput, it can be shown that it also belongs to GR (although β for DRR will be very high). Hence, the class of GR algorithms is broad.

If a server employs any of these scheduling algorithm in GR, then it guarantees that packet p^j_f will be transmitted by $GRC^i(p^j_f, r^{j,i}_f) + \beta^i$. Since $GRC^i(p^j_f, r^{j,i}_f)$, and hence the delay guarantee, is independent of a traffic characterization, a server employing a scheduling algorithm in GR can provide various service guarantees to flows conforming to any traffic specification. For example, it enables a server to guarantee an upper bound on delay and tail distribution of delay to packets of a flow conforming to leaky bucket and Exponentially Bounded Burstiness (EBB) process, respectively [10]. In a network environment, however, packets of a flow are serviced by a sequence of servers. In what follows, we present a method for deriving the delay guarantee of a network of servers each of which employs a scheduling algorithm in the GR class.

3 Delay Guarantee of a Network of Servers

To derive the delay guarantee for a network of servers, each of which employs a scheduling algorithm in GR, consider flow f that is serviced by K servers. Let the i^{th} server on the path be denoted by i. Then, the network guarantees that packet p^j_f will depart from the network by $GRC^K(p^j_f, r^{j,K}_f) + \beta^K$. This delay guarantee depends on the arrival process at the K^{th} server, i.e., $A^K(p^j_f)$. Though $A^K(p^j_f)$ depends on the arrival process of a flow, i.e., $A^1(p^j_f)$, due to the variability in the delay experienced by the packets of a flow, it may not always be possible to determine relationship between $A^K(p^j_f)$ and $A^1(p^j_f)$. Since, $A^1(p^j_f)$, and not $A^K(p^j_f)$, is always known, it is desirable to characterize the delay guarantee of a network of servers such that it is determined by $A^1(p^j_f)$.

Observe that $GRC^K(p^j_f, r^{j,K}_f)$ depends on $A^K(p^j_f)$, which in turn depends on $GRC^{K-1}(p^j_f, r^{j,K-1}_f)$. Applying this argument recursively, $GRC^K(p_f^j, r_f^{j,K-1})$ can be related to $GRC^1(p_f^j, r_f^{j,1})$. Consequently, the delay guarantee of a network of servers can be characterized based on $GRC^1(p_f^j, r_f^{j,1})$ which is completely determined by $A^1(p_f^j)$ (i.e., the arrival process of a flow), and the rate assigned to the packets. This enables a network of servers, as in the case of a single server, to provide service guarantee to flows conforming to any specification. This methodology was employed in [10] when constant rate is allocated to the packets of a flow and packet fragmentation and reassembly does not occur. In what follows, we extend that methodology to networks where: (1) different rates are allocated to the packets of a flow at different servers along the path and the bottleneck server for each packet may be different, and (2) packet fragmentation and/or reassembly may occur.

To derive the delay guarantee of a network of servers, we will first relate the guaranteed rate clock value of a packet at two adjacent servers. A similar approach of analyzing two adjacent servers to analyze end-to-end behaviour of a network of servers has been used in [2, 18]. Henceforth we will always refer to a single flow f and hence, for ease of presentation, drop the subscript f from all the variables.

3.1 Two Server Case

In large networks, due to the variability in load at different servers, different rates may be allocated to packets at different servers. Since throughput of a network for a flow is governed by throughput of the bottleneck server, instead of relating $GRC^{i+1}(p^j, r^{j,i+1})$ and $GRC^i(p^j, r^{j,i})$, we will establish a relationship between $GRC^{i+1}(p^j, \widehat{r}^{j,i})$ and $GRC^i(p^j, \widehat{r}^{j,i})$ where $GRC^i(p^j, \widehat{r}^{j,i})$ denotes the guaranteed rate clock value computed using $\widehat{r}^{k,i}$; $k \leq j$; and $\widehat{r}^{j,i}$ represents the bottleneck rate for p^j which is defined as $\min\{r^{j,i}, r^{j,i+1}\}$[4]. Observe that this definition captures the scenario where different servers may be the bottleneck server for different packets of the same flow. Although we relate the guaranteed rate clock values computed using bottleneck rate, our analysis will demonstrate that allocating different rates at different servers leads to smaller end-to-end delay than allocating bottleneck rate at each server.

We would find the following inequality useful in establishing the relationship between guaranteed rate clock values at adjacent nodes. Since the guaranteed rate clock value of a packet computed using a smaller rate is larger, $GRC^i(p^j, r^{j,i}) \leq GRC^i(p^j, \widehat{r}^{j,i})$. Hence,

$$\begin{aligned} GRC^i(p^j, r^{j,i}) &= \max\{A^i(p^j), GRC^i(p^{j-1}, r^{j-1,i})\} \\ &\quad + \frac{l^j}{r^{j,i}} \\ &\leq \max\{A^i(p^j), GRC^i(p^{j-1}, \widehat{r}^{j-1,i})\} \\ &\quad + \frac{l^j}{r^{j,i}} \end{aligned}$$

In heterogeneous networks, the data transmission unit may vary and hence packet fragmentation and reassembly may occur. Such a scenario, for example, would occur in an internetwork consisting of ATM, ethernet, and FDDI subnetworks. The relationship between $GRC^{i+1}(p_f^j, r_f^{j,i+1})$ and $GRC^i(p_f^j, r_f^{j,i})$ depends on the occurrence of such a scenario. Hence, in the following subsections, we first establish the relationship when packet fragmentation and reassembly do not occur and then consider their effects.

3.1.1 No Fragmentation and/or Reassembly

Theorem 6 If the scheduling algorithm at server i belongs to GR for flow f , then

$$GRC^{i+1}(p^j, \widehat{r}^{j,i}) \leq GRC^i(p^j, \widehat{r}^{j,i}) + \max_{k \in [1..j]} \frac{l^k}{r^{k,i}} + \alpha^i \quad (12)$$

where $\alpha^i = \beta^i + \tau^{i,i+1}$ and $\tau^{i,i+1}$ is an upper bound on the propagation delay between servers i and $i+1$.

The proof of Theorem 6 only requires $A^{i+1}(p_f^j) \leq GRC^i(p^j, r^{j,i}) + \alpha^i$. Hence, it holds even if a server employs a jitter controller, similar to that employed by Jitter EDD, to ensure that the arrival time of packet p_f^j is exactly $GRC^i(p^j, r^{j,i}) + \alpha^i$. Also, observe that even though we have related guaranteed clock values computed based on the bottleneck rate, the denominator in the second term of the right hand side of (12) is $r^{j,i}$ instead of $\widehat{r}^{j,i}$. This would enable us to derive tighter end-to-end delay bounds in section 4.

3.1.2 Packet Fragmentation

Let server i receive a packet larger than it can transmit. As we had mentioned in Section 2.4.1, server i may schedule packets or packet fragments. Let us consider the two cases:

- If server i schedules packets (see Figure 3(a)), it may or may not allow packet transmission to be preempted after transmission of every packet fragment. In either case, $GRC^i(p^j, \widehat{r}^{j,i-1})$ can be related to $GRC^{i-1}(p^j, \widehat{r}^{j,i-1})$ using Theorem 6. However, we need to relate the guaranteed rate clock value of packet at server i and $i+1$.
- If server i schedules packet fragments(see Figure 3(b)), then guaranteed rate clock values of packet fragments at server i and $i+1$ can be related using Theorem 6. However, we need to relate the guaranteed rate clock value of packet at server $i-1$ and i.

From the two cases, we can infer that, in the presence of fragmentation, we need to relate the guaranteed rate clock values of packets between two adjacent servers that have different scheduling units, i.e, packets and packet fragments. Let packet p^j be fragmented into $\theta(j)$ fragments, $p^{j,k}$ denote the k^{th} fragment of p^j and let $l^{j,k}$ denote the length of packet fragment $p^{j,k}$ [5]. Since the delay guarantee

[4] To facilitate the proof of multiple server case, we have chosen an inequality, rather than an equality.

[5] The length of a packet fragment does not include the headers that may be added due to fragmentation. To account for the overhead due to these headers, as in case of link-level headers, we assume that the actual rate assigned to a packet fragment is higher. A bound on increase in rate can be derived using a bound on maximum packet length of a flow.

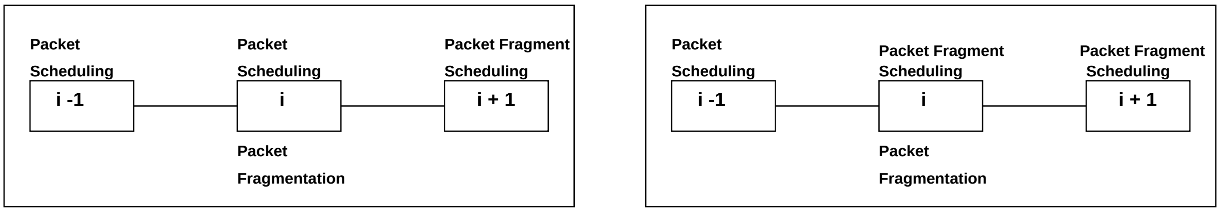

Figure 3: At the switch where fragmentation occurs: (a) Packet scheduling, (b) Packet fragment scheduling

of a packet is determined by the delay guarantee of the last fragment of a packet, we relate the guaranteed rate clock value of the last fragment of a packet at adjacent servers in Theorem 7.

Theorem 7 If the scheduling algorithm at server i schedules packets and belongs to GR for flow f, and server $i+1$ schedules packet fragments, then

$$GRC^{i+1}(p^{j,\theta(j)}, \widehat{r}^{j,i}) \leq GRC^{i}(p^{j}, \widehat{r}^{j,i}) + \max_{k \in [1..j]} \frac{l^k}{r^{k,i}} + \alpha^i \tag{13}$$

where $\alpha^i = \beta^i + \tau^{i,i+1}$ and $\tau^{i,i+1}$ is an upper bound on the propagation delay between servers i and $i+1$.

3.1.3 Packet Reassembly

Let server $i+1$ perform reassembly of packet fragments. Theorem 8 relates the guaranteed rate clock value of the last cell of a packet at server i and $i+1$.

Theorem 8 If the scheduling algorithm at server i belongs to GR for flow f and server $i+1$ performs reassembly, then

$$GRC^{i+1}(p^{j}, \widehat{r}^{j,i}) \leq GRC^{i}(p^{j,\theta(j)}, \widehat{r}^{j,i}) + \max_{k \in [1..j]} \frac{l^k}{\widehat{r}^{k,i}} + \alpha^i \tag{14}$$

where $\alpha^i = \beta^i + \tau^{i,i+1}$ and $\tau^{i,i+1}$ is an upper bound on the propagation delay between servers i and $i+1$.

Observe that the second term in the right hand side of (14) is $\max_{k \in [1..j]} \frac{l^k}{\widehat{r}^{k,i}}$ which is different from the corresponding term, $\max_{k \in [1..j]} \frac{l^k}{r^{k,i}}$, in (12) and (13).

3.2 Multiple Server Case

In Theorems 6, 7, and 8, we have related the guaranteed rate clock values of a packet at adjacent servers under various scenarios. These relationships can be employed to relate the guaranteed rate clock value of the packet at K^{th} server to that at the first server. Clearly, such a relationship depends on the configuration of the path of a flow. We illustrate the relationship between the guaranteed rate clock value of the packet at the K^{th} server and its value at the first server for the path configuration shown in Figure 4. In Figure 4 packet fragmentation occurs at server F and packet reassembly occurs at server R. Let server O be the first server on the path of the flow that schedules packet fragments. If server F schedules packets, then the next server schedules packet fragments and hence $O = F+1$; otherwise $O = F$.

Theorem 9 If the scheduling algorithm at each of the servers on the path of a flow belongs to GR for flow f, then

$$\begin{aligned} GRC^{K}(p^j, \widehat{r}^j) \leq GRC^{1}(p^j, \widehat{r}^j) + \sum_{i=1}^{i=O-1} \max_{n \in [1..j]} \frac{l^n}{r^{n,i}} \\ + \sum_{i=O}^{i=R-2} \max_{n \in [1..j]} \frac{\hat{l}^n}{r^{n,i}} + \max_{n \in [1..j]} \frac{l^n}{\widehat{r}^n} + \sum_{i=R}^{i=K-1} \max_{n \in [1..j]} \frac{l^n}{r^{n,i}} \\ + \sum_{n=1}^{n=K-1} \alpha^n \end{aligned}$$

where $\alpha^n = \beta^n + \tau^{n,n+1}$, K is the number of servers on the path of the flow, $\hat{l}^n$ is the length of the biggest fragment of p^n, and $\widehat{r}^j$ is the bottleneck rate for packet p^j, i.e., $\widehat{r}^j = \min_{i \in [1..K]} r^{j,i}$.

It can be similarly shown that if packet fragmentation does not occur, then:

$$\begin{aligned} GRC^{K}(p^j, \widehat{r}^j) \leq \; & GRC^{1}(p^j, \widehat{r}^j) + \sum_{i=1}^{i=K-1} \max_{n \in [1..j]} \frac{l^n}{r^{n,i}} \\ & + \sum_{n=1}^{n=K-1} \alpha^n \end{aligned} \tag{15}$$

Observe that even though we have derived the delay guarantee based on the bottleneck rate, the denominator in $\sum_{i=1}^{i=K-1} \max_{n \in [1..j]} \frac{l^n}{r^{n,i}}$ is the actual rate allocation at each server. Clearly, this term is smaller than the similar term that would have been derived assuming bottleneck rate allocation at each server.

The delay guarantee of a network of servers enables a network to provide various service guarantees to flows conforming to any specification. For example, as we illustrate in the next section, the delay guarantee can be employed to guarantee an upper bound on end-to-end delay of packets of a flow.

4 End-to-End Delay Bound

To determine an upper bound on the end-to-end delay of packets of a flow, consider a flow which is served by K servers. Also, let server 0 be the source and server $K+1$ be the destination. Let d^j be the delay experienced by the packets of a flow. Since server K guarantees that packet p^j will be transmitted by time $GRC^K(p^j, r^{j,K}) + \beta^K$ and

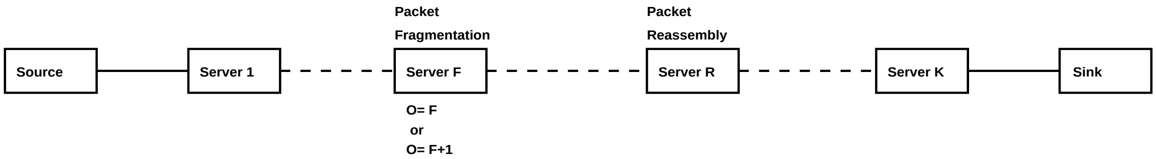

Figure 4: The path configuration

the packet arrives at the first node at time $A^1(p^j)$, we get:

$$d^j \leq GRC^K(p^j, r^{j,K}) + \alpha^K - A^1(p^j)$$

where $\alpha^K = \beta^K + \tau^{K,K+1}$. Observe that $GRC^K(p^j, \widehat{r}^j) - GRC^K(p^j, r^{j,K}) \geq \frac{l^j}{\widehat{r}^j} - \frac{l^j}{r^{j,K}}$. Hence,

$$d^j \leq GRC^K(p^j, \widehat{r}^j) - \left(\frac{l^j}{\widehat{r}^j} - \frac{l^j}{r^{j,K}}\right) + \alpha^K - A^1(p^j)$$

If each server on the path of the flow employs a scheduling algorithm in GR, then given the path configuration, $GRC^K(p^j, \widehat{r}^j)$ can be related to $GRC^1(p^j, \widehat{r}^j)$. For instance, when packet fragmentation and reassembly does not occur along the path, then using (15) we get:

$$d^j \leq \left(GRC^1(p^j, \widehat{r}^j) - A^1(p^j)\right) + \left(\sum_{i=1}^{i=K-1} \max_{n \in [1..j]} \frac{l^n}{r^{n,i}} - \left(\frac{l^j}{\widehat{r}^j} - \frac{l^j}{r^{j,K}}\right)\right) + \left(\sum_{n=1}^{n=K} \alpha^n\right)$$

Hence, the end-to-end delay of a packet consists of three components:

- $\sum_{n=1}^{n=K} \alpha^n$: Since $\alpha^n = \beta^n + \tau^{n,n+1}$, this term is completely characterized by the scheduling algorithms and the propagation delay in the network.
- $\sum_{i=1}^{i=K-1} \max_{n \in [1..j]} \frac{l^n}{r^{n,i}} - (\frac{l^j}{\widehat{r}^j} - \frac{l^j}{r^{j,K}})$: This term depends on the length of the packets transmitted by a source and the rate allocated to it at various servers. Hence, this term is known if the length of the packets transmitted by a source and the rate assignments are known.
- $GRC^1(p^j, \widehat{r}^j) - A^1(p^j)$: This is the only term that depends on arrival process characteristics of a flow. This term can be interpreted as the queuing delay experienced by a packet at a single server with variable capacity; the capacity being the bottleneck rate for the packet in service. Hence, the network can be abstracted as a single server with variable capacity and consequently the problem of determining end-to-end delay is reduced to the problem of determining delay at a single node. Therefore, a single server queuing analysis can be employed to determine an upper bound on the delay or the tail distribution of delays experienced by packets of a flow for any traffic specification. For example:
  - If a flow conforms to Leaky Bucket with parameters (σ, r) and $\widehat{r}$ is the minimum rate allocated to the packets of the flow such that $r \leq \widehat{r}$, then from [10] we get:

    $$GRC^1(p^j, \widehat{r}) - A^1(p^j) \leq \frac{\sigma}{\widehat{r}} \qquad (16)$$

    Substituting (16) in (16) gives an upper bound on the end-to-end delay.
  - If a flow conforms to Exponentially Bounded Burstiness (EBB) process with parameters (r, Λ, γ) [22], and $\widehat{r}$ is the minimum rate allocated to the packets of the flow such that $r \leq \widehat{r}$, then from [10] we get:

    $$Pr\left(GRC^1(p^j, \widehat{r}) - A^1(p^j) \geq \frac{y}{\widehat{r}}\right) \leq \Lambda\, e^{-\gamma\, y} \qquad (17)$$

    An upper bound on the tail distribution of the end-to-end delay is derived by substituting (17) in (16). An upper bound on tail distribution of end-to-end delay of a markovian process can be similarly derived.
  - If a flow conforms to the variable rate Flow Specification introduced in [15], then for the first packet of a burst of a flow:

    $$GRC^1(p^j, \widehat{r}^j) - A^1(p^j) \leq \frac{l^j}{\widehat{r}^j} \qquad (18)$$

    Substituting (18) in (16) gives an upper bound on end-to-end delay which is a generalization of the bound for constant length packets in [15]. This generalization can be exploited to design algorithms that, by appropriately selecting packet lengths, reduce the jitter of VBR video flows.

In the above analysis, $\tau^{i,i+1}$ is an upper bound on the propagation delay. Hence, the delay bounds also hold in networks where the propagation delay may be variable but is bounded. This property is desirable in internetworks [24]. The above method also leads to tighter results. If $r^{j,K} = \widehat{r}^j$ (assumed for ease of exposition) then the only variable term in (16), that depends on network and not flow characteristics is:

$$\sum_{i=1}^{i=K-1} \max_{n \in [1..j]} \frac{l^n}{r^{n,i}} \qquad (19)$$

This term is smaller than other similar analysis in the literature in several ways. We illustrate the subtleties in (19) by comparing it with the seminal analysis presented in [17] for Rate Proportional Processor Sharing (RPPS) rate assignment of PGPS networks. The delay bound in [17] for a flow that conforms to Leaky Bucket with parameters (σ, r) and has minimum rate $\widehat{r} \geq r$ assigned to the packets ($\widehat{r} \geq r$) is:

$$d^j \leq \frac{\sigma}{\widehat{r}} + 2 \sum_{i=1}^{i=K-1} \frac{l^{max}}{\widehat{r}} + \sum_{n=1}^{n=K} \alpha^n$$

where l^{max} is the maximum length of a packet of the flow. Hence, the term corresponding to (19) is:

$$2 \sum_{i=1}^{i=K-1} \frac{l^{max}}{\widehat{r}} \tag{20}$$

(20) is larger than (19) in several ways:

- The factor 2 makes (20) larger than (19).
- Even when different rates may be allocated at different servers, the denominator in (20) is $\widehat{r}$. In contrast, the denominator in (19) is $r^{n,i}$ where $r^{n,i} \geq \widehat{r}$. To illustrate the differences numerically, consider a flow with packets of length 100 bytes that is being served by two servers such that the rate allocations at server 1 and 2 are 64Kb/s and 32Kb/s, respectively. Then, neglecting factor 2, (20) evaluates to 24.4 ms which is larger than the 12.2 ms computed from (19).
- The numerator in (20) is l^{max} which may be larger than the numerator, $\max_{n \in [1..j]} l^n$ (assuming a constant rate allocation for packets of a flow), in (19).
  This difference can be made larger by defining a network *busy period.* For ease of exposition, let no fragmentation and reassembly occur. Then, a network is considered busy for flow f at time t if

  $$GRC^1(p^j, \widehat{r}^j) + \sum_{i=1}^{i=K-1} \max_{n \in [1..j]} \frac{l^n}{r^{n,i}} + \sum_{n=1}^{n=K-1} \alpha^n \leq t$$

  where p^j is the last packet to have arrived before t. Let the packets be renumbered such that packet p^j is the j^{th} packet to arrive in a network busy period. From the proofs of Theorems 6, 7 and 8, we know that Theorem 9 still continues to hold. Consequently, in (19), the maximum is over a subsequence of packets rather than all the previous packets.
  To illustrate the difference numerically, consider a flow that has 5 servers on the path and has 1Mb/s rate allocated to its packets. Consider two busy periods 1 and 2 such that the maximum packet length during the periods is 1000 bytes and 100 bytes, respectively. Then, (19) evaluates to 30.5 ms and 3.05 ms for busy periods 1 and 2, respectively. If the maximum packet length the flow ever transmits is 1500 bytes, then (20) will always yield 45.7 ms.
- Whereas (20) does not quantify the effect of variable rate allocation to packets of a flow, (19) does. Furthermore, (19) when used in conjunction with the concept of busy period enables derivation of better delay bounds. To illustrate numerically, consider a flow that has 5 servers on the path and transmits 1000 byte packets. Consider two busy periods 1 and 2 in which the rate allocations are 100Kb/s and 1Mb/s, respectively. Then, (19) evaluates to 305 ms and 30.5 ms for busy periods 1 and 2, respectively. Without the notion of busy period, the bound would have been 305 ms in both the busy periods.

Since (19) is the only term that depends on the network, these improvements are significant.

5 Concluding Remarks

In this paper, we have defined the class of GR scheduling algorithms that includes scheduling algorithms that allocate variable rate to the packets of a flow. We defined work conserving generalized Virtual Clock, Packet-by-Packet Generalized Processor Sharing and Self Clocked Fair Queuing scheduling algorithms that can allocate variable rate to the packets of a flow. We also defined scheduling algorithms suitable for servers where packet fragmentation may occur. We demonstrated that if a class of rate controllers is employed for a flow in conjunction with any scheduling algorithm in GR, then the resulting non-work conserving algorithm also belongs to GR. This lead to the definition of non-work conserving equivalents of generalized Virtual Clock, PGPS and SCFQ algorithms and a combination of Delay EDD and a variant of Jitter EDD.

We presented a method for deriving the delay guarantee of a network of servers when: (1) different rates are allocated to packets of a flow at different servers on the path and the bottleneck server for each packet may be different, and (2) packet fragmentation and/or reassembly may occur. The delay guarantee enables a network to provide various service guarantees to flows conforming to any specification. The delay guarantee was then employed to illustrate the derivation of delay bounds for flows conforming to Leaky Bucket, Exponentially Bounded Burstiness and Flow Specification. The method for determining these bounds is valid in internetworks and leads to tighter results.

Acknowledgements

This research was supported in part by IBM Graduate Fellowship, IBM Faculty Development Award, Intel, the National Science Foundation (Research Initiation Award CCR-9409666 and CAREER award CCR-9624757), NASA, Mitsubishi Electric Research Laboratories (MERL), and Sun Microsystems Inc.

References

[1] J.C.R. Bennett and H. Zhang. WF^2Q: Worst-case Fair Weighted Fair Queuing. In *Proceedings of INFOCOM'96*, pages 120–127, March 1996.

[2] R.L. Cruz. Service Burstiness and Dynamic Burstiness Measures: A Framework. *Journal of High Speed Networks*, 2:105–127, 1992.

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

[4] R.L. Cruz. A Calculus for Network Delay, Part II : Network Analysis. *IEEE Transactions on Information Theory*, 37:132–141, Jan 1991.

[5] A. Demers, S. Keshav, and S. Shenker. Analysis and Simulation of a Fair Queueing Algorithm. In *Proceedings of ACM SIGCOMM*, pages 1–12, September 1989.

[6] D. Ferrari. Client Requirements for Real-Time Communication Services. *IEEE Communications Magazine*, pages 65–72, November 1990.

[7] L. Georgadis, R. Guerin, and A. Parekh. Optimal Multiplexing on a Single Link: Delay and Buffer Requirements. In *Proceeding of INFOCOM'94*, 1994.

[8] L. Georgiadis, R. Guerin, V. Peris, and K.N. Sivarajan. Efficient Network QoS Provisioning Based on per Node Traffic Shaping. In *Proceedings of INFOCOM'96*, pages 102–110, March 1996.

[9] S.J. Golestani. A Self-Clocked Fair Queueing Scheme for High Speed Applications. In *Proceedings of INFOCOM'94*, 1994.

[10] P. Goyal, S. S. Lam, and H. M. Vin. Determining End-to-End Delay Bounds In Heterogeneous Networks. In *ACM/Springer-Verlag Multimedia Systems Journal (to appear)*. Also appeared in the Proceedings of the Workshop on Network and Operating System Support for Digital Audio and Video, Pages 287-298, April 1995.

[11] P. Goyal and H. M. Vin. Generalized Guaranteed Rate Scheduling Algorithms: A Framework. Technical Report TR-95-30, Department of Computer Sciences, The University of Texas at Austin, 1995. Available via URL http://www.cs.utexas.edu/users/dmcl.

[12] A. Greenberg and N. Madras. How Fair is Fair Queuing. *The Journal of ACM*, 39(3):568–598, July 1992.

[13] M. Grossglauser, S. Keshav, and D. Tse. RCBR: A Simple and Efficient Service for Multiple Time-Scale Traffic. In *Proceedings of ACM SIGCOMM'95*, 1995.

[14] D. D. Kandlur, K. G. Shin, and D. Ferrari. Real-Time Communication in Multihop Networks. *IEEE Transactions on Parallel and Distributed Systems*, 5(10):1044–1056, October 1994.

[15] S.S. Lam and G.G. Xie. Burst Scheduling: Architecture and Algorithm for Switching Packet Video. In *Proceedings of INFOCOM'95*, April 1995.

[16] J. Mogul and S. Deering. Path MTU discovery. RFC 1191, 1990.

[17] A. K. Parekh and R. G. Gallager. A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Multiple Node Case. *IEEE/ACM Transactions On Networking*, 2(2):137–150, April 1994.

[18] A.K. Parekh. *A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks.* PhD thesis, Department of Electrical Engineering and Computer Science, MIT, 1992.

[19] M. Shreedhar and G. Varghese. Efficient Fair Queuing Using Deficit Round Robin. In *Proceedings of ACM SIGCOMM'95*, pages 231–242, 1995.

[20] D. Stiliadis and A. Varma. Design and Analysis of Frame-based Fair Queueing: A New Traffic Scheduling Algorithm for Packet Switched Networks. In *Proceedings of SIGMETRICS'96*, May 1996.

[21] G.G. Xie and S.S. Lam. Delay Guarantee of Virtual Clock Server. *IEEE/ACM Transactions on Networking*, 3(6):683–689, December 1995.

[22] O. Yaron and M. Sidi. Generalized Processor Sharing Networks with Exponentially Bounded Burstiness Arrivals. In *Proceedings of INFOCOM'94*, 1994.

[23] H. Zhang. Service Disciplines For Guaranteed Performance Service in Packet-Switching Networks. *Proceedings of the IEEE*, 83(10), October 1995.

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

[25] H. Zhang and S. Keshav. Comparison of Rate-Based Service Disciplines. In *Proceedings of ACM SIGCOMM*, pages 113–121, August 1991.

[26] L. Zhang. VirtualClock: A New Traffic Control Algorithm for Packet Switching Networks. In *Proceedings of ACM SIGCOMM'90*, pages 19–29, August 1990.

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

Pawan Goyal Pawan Goyal received his B.Tech. in Computer Sciences from the Indian Institute of Technology, Kanpur in 1992, and is currently a doctoral candidate at the Department of Computer Sciences at the University of Texas at Austin. His research interests are in computer networks, multimedia systems, operating systems and distributed systems. He has received several awards for academic excellence including the MCD Fellowship (awarded by the University of Texas at Austin) and the IBM Doctoral Fellowship.

Harrick M. Vin Harrick M. Vin received his Ph.D. in Computer Science from the University of California at San Diego in 1993. He is currently an Assistant Professor of Computer Sciences, and the Director of the Distributed Multimedia Computing Laboratory at the University of Texas at Austin. His research interests are in the areas of multimedia systems, high-speed networking, mobile computing, and large-scale distributed systems. Over the past 5 years, he has co-authored more than 55 papers in leading journals and conferences in the area of multimedia systems. He has been a recipient of several awards including the National Science Foundation CAREER award, IBM Faculty Development Award, AT&T Foundation Award, IBM Doctoral Fellowship, NCR Innovation Award, and the San Diego Supercomputer Center Creative Computing Award.