Determining End-to-End Delay Bounds In Heterogeneous Networks *

Pawan Goyal, **Simon S. Lam** and **Harrick M. Vin**

Department of Computer Sciences, University of Texas at Austin
Taylor Hall 2.124, Austin, Texas 78712-1188
E-mail: {pawang,lam,vin}@cs.utexas.edu, Telephone: (512) 471-9718

Abstract

We define a class of Guaranteed Rate (GR) scheduling algorithms. The GR class includes Virtual Clock, Packet-by-Packet Generalized Processor Sharing and Self Clocked Fair Queuing. For networks that employ scheduling algorithms belonging to GR, we present a method for determining an upper bound on end-to-end delay. The method facilitates determination of end-to-end delay bounds for a variety of sources. We illustrate the method by determining end-to-end delay bounds for sources conforming to Leaky Bucket and Exponentially Bounded Burstiness.

1 Introduction

Computer networks have advanced to a point where they can support multimedia applications like audio and video conferencing and multimedia information retrieval. Such applications require the network to provide a wide range of Quality of Service (QoS) guarantees (including minimum bandwidth, packet delay, delay jitter and loss). Whereas the minimum guaranteed bandwidth must be large enough to accommodate motion video of acceptable resolution, the end-to-end delay must be small enough for interactive communication. In order to avoid breaks in continuity of audio and video playback, delay jitter and loss must be sufficiently small. Techniques for determining an upper bound on end-to-end delay in a network is the subject matter of this paper.

The end-to-end delay of a packet depends on the source traffic characteristics and the scheduling algorithm at the network switches. In the recent past, several source specifications including Leaky Bucket [12], Exponentially Bounded Burstiness (EBB) [15], Flow Specification [10] and the Tenet model [5] have been studied. The scheduling algorithms that have been proposed include Stop and Go Queuing [6], Delay EDD [17], Jitter EDD [17], Hierarchical Round Robin [9], Rate Control Static Priority Queuing

*This research was supported in part by IBM Graduate Fellowship, the National Science Foundation (Research Initiation Award CCR-9409666), National Science Foundation Grant No. NCR-9004464, NASA, Mitsubishi Electric Research Laboratories (MERL), and Sun Microsystems Inc.

[16], Self Clocked Fair Queuing (SCFQ) [7], Virtual Clock [18] and Packet-by-Packet Generalized Processor Sharing (PGPS) [12]. The problem of determining an upper bound on end-to-end delay has also received considerable attention [2, 3, 10, 12, 15]. However, most of the techniques determine end-to-end delay by considering a specific source traffic specification and scheduling algorithm. In an integrated network supporting audio, video and data services, the sources have widely varying characteristics. Moreover, in a wide area networking environment, each switch may employ a different scheduling algorithm. Methods for determining end-to-end delay of packets in such heterogeneous environments have not received much attention.

In this paper, we take a step towards addressing the above limitation by (1) defining a class of *Guaranteed Rate (GR)* scheduling algorithms, and (2) developing a method for determining an upper bound on end-to-end delays for a network of switches each of which employs a scheduling algorithm in the GR class. We demonstrate that many of the scheduling algorithms proposed in the literature (e.g. Virtual Clock, PGPS and SCFQ) belong to the class of GR scheduling algorithms. We also show that the method for determining an upper bound on end-to-end delay is general and can be used to determine delay bounds for various source traffic specifications. We employ the method to derive a deterministic end-to-end delay bound for Leaky Bucket sources (a deterministic source characterization) and an upper bound on the tail distribution of the delay for EBB sources (a stochastic characterization). The end-to-end delay bounds that we derive are parameterized by the scheduling algorithm used at each switch and can be instantiated to derive delay bounds for a specific scheduling algorithm or a combination of scheduling algorithms.

The rest of the paper is organized as follows. In Section 2, we define the class of GR scheduling algorithms. In section 3, we present a method for determining end-to-end delay bounds and finally, Section 4 summarizes our results.

2 Guaranteed Rate Scheduling Algorithms

Each unit of data transmission at the network level is a packet. We refer to the sequence of packets transmitted by a source as a *flow* [18]. Each packet within a flow is serviced by a sequence of servers (switching elements) along the path from the source to the destination in the network. To provide guaranteed performance, the servers reserve a rate for a flow and employ a rate-based scheduling algorithm. Based on this rate reservation, many scheduling algorithms can guarantee a deadline by which a packet of a flow will be transmitted. This guarantee is referred to as *delay guarantee*. We refer to the class of rate based scheduling algorithms which provide such guarantees as *Guaranteed Rate (GR)* scheduling algorithms.

The delay guarantees provided by these algorithms are based on the *guaranteed rate clock*[1] values (intuitively, expected arrival time values) associated with each packet. To define the guaranteed rate clock values, consider a flow f that is associated with

[1] The concept of guaranteed rate clock is the same as the concept of virtual clock in [18]. We coin a new term for the virtual clock concept to avoid any confusion between the general concept of virtual clock and as it relates to the Virtual Clock scheduling algorithm.

a rate r_f (in bits/sec). Let p_f^j and l_f^j denote the j^{th} packet of flow f and its length, respectively. Additionally, let $GRC^i(p_f^j)$ and $A^i(p_f^j)$ denote the guaranteed rate clock value and arrival time of packet p_f^j at server i, respectively. Then, guaranteed rate clock value for a packet is given by:

$$GRC^i(p_f^0) = 0 \tag{1}$$

$$GRC^i(p_f^j) = \max\{A^i(p_f^j), GRC^i(p_f^{j-1})\} + \frac{l_f^j}{r_f} \quad j \geq 1 \tag{2}$$

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 $GRC^i(p_f^j) + \beta^i$ where β^i is a constant which depends on the scheduling algorithm and the server.*

The concept of delay guarantee to a packet based on its expected arrival time was introduced in [10, 14]. As is evident from the definition (also observed in [10]), a key property of the class of GR scheduling algorithms is that they provide a delay guarantee for a source independent of the behavior of other sources in the network, and hence isolate the sources. Isolation of sources has been considered to be a very desirable property of scheduling algorithms, especially in large heterogeneous networks where sources may be malicious [1, 4, 10, 13]. This property is the basis for development of a conceptually simple method for determining end-to-end delay bounds in section 3.

In the next few subsections we demonstrate that the GR class includes Virtual Clock, PGPS and SCFQ with certain rate assignments.

2.1 Virtual Clock

The Virtual Clock scheduling algorithm assigns each packet a virtual clock value on its arrival and orders the transmission of packets by increasing virtual clock values. If flow f is assigned rate r_f at server i, then the virtual clock value for packet p_f^j of flow f at server i, denoted by $VC^i(p_f^j)$, is computed as follows [18]:

$$VC^i(p_f^0) = 0 \tag{3}$$

$$VC^i(p_f^j) = \max\{A^i(p_f^j), VC^i(p_f^{j-1})\} + \frac{l_f^j}{r_f} \quad j \geq 1 \tag{4}$$

The following delay guarantee was presented in [14]. Define a flow to be active at time t if $VC^i(p_f^j) \geq t$ where p_f^j is the last packet of flow f that has arrived before time t. Let $a^i(t)$ denote the set of active flows at server i at time t. Server i with capacity C^i is defined to have exceeded its capacity at time t if $\sum_{n \in a^i(t)} r_n > C^i$. Let l_{max}^i be the maximum length of the packet served by server i and $L_{VC}^i(p_f^j)$ be the time at which the transmission of packet p_f^j is completed. Then, if a server's capacity has not been exceeded and flow f is assigned rate r_f,

$$L_{VC}^i(p_f^j) \leq VC^i(p_f^j) + \frac{l_{max}^i}{C^i} \tag{5}$$

Since the equations for virtual clock and guaranteed rate clock are the same, it can be easily observed that if rate r_f is assigned to flow f, then the Virtual Clock scheduling algorithm belongs to GR for flow f with $\beta^i = \frac{l^i_{max}}{C^i}$.

2.2 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 [12]. In fact, PGPS simulates GPS such that

$$L^i_{PGPS}(p^j_f) \leq L^i_{GPS}(p^j_f) + \frac{l^i_{max}}{C^i} \tag{6}$$

where $L^i_{PGPS}(p^j_f)$ and $L^i_{GPS}(p^j_f)$ are the times at which packet p^j_f leaves server i employing the PGPS scheduling algorithm and the GPS service discipline, respectively, and l^i_{max} is the maximum length of the packet served by server i [12]. Hence, to show that PGPS belongs to the class of GR scheduling algorithms, we establish a relationship between $L^i_{GPS}(p^j_f)$ and $GRC^i(p^j_f)$.

In GPS, each flow f is associated with a constant ϕ^i_f at server i. GPS is defined such that if flow f is backloged at time t, it receives service at the rate of $\frac{\phi^i_f C^i}{\sum_{k \in b^i(t)} \phi^i_k}$ where C^i is the capacity of the server and $b^i(t)$ is the set of backlogged flows at GPS server i at time t. Thus, a GPS server is defined to have assigned rate r_f to flow f if $\frac{\phi^i_f C^i}{\sum_{k \in b^i(t)} \phi^i_k} \geq r_f$ whenever flow f is backlogged. Hence, if flow f has been assigned rate r_f, then

$$L^i_{GPS}(p^j_f) \leq \max\{A^i(p^j_f), L^i_{GPS}(p^{j-1}_f)\} + \frac{l^j_f}{r_f} \quad j \geq 1 \tag{7}$$

Let $L^i_{GPS}(p^0_f) = GRC^i(p^0_f) = 0$. From (7) and (2), it can be easily shown that

$$L^i_{GPS}(p^j_f) \leq GRC^i(p^j_f) \quad j \geq 1 \tag{8}$$

From (8) and (6), we get

$$L^i_{PGPS}(p^j_f) \leq GRC^i(p^j_f) + \frac{l^i_{max}}{C^i} \quad j \geq 1 \tag{9}$$

Hence if rate r_f is assigned to flow f at server i , then the PGPS scheduling algorithm belongs to GR for flow f with $\beta^i = \frac{l^i_{max}}{C^i}$.

2.3 Self Clocked Fair Queuing

The Self Clocked Fair Queuing scheme, proposed in [7], was designed to facilitate the implementation of a fair queuing scheme in broadband networks. The scheme is defined as follows.

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

$$F^i(p_f^0) = 0 \tag{10}$$

$$F^i(p_f^j) = \max\{F^i(p_f^{j-1}), v^i(A^i(p_f^j))\} + \frac{l_f^j}{\phi_f^i} \quad j \geq 1 \tag{11}$$

where ϕ_f^i is a constant associated with flow f at server i.

2. The server virtual time at time t, $v^i(t)$, is defined to be equal to the service tag of the packet being service at time t. $v^i(t) = t$ when the server is idle.

3. Packets are serviced in increasing order of their service tags.

The following theorem proves that SCFQ algorithm also belongs to the class of GR scheduling algorithms. Let c^i be the set of flows serviced by server i employing SCFQ scheduling algorithm.

Theorem 1 *If $\sum_{m \in c^i} r_m \leq C^i$ and $\forall m \in c^i : \phi_m^i = r_m$, then the departure time of packet p_f^j in SCFQ (denoted by $L_{SCFQ}^i(p_f^j)$) is given by*

$$L_{SCFQ}^i(p_f^j) \leq GRC^i(p_f^j) + \sum_{m \in c^i \wedge m \neq f} \frac{l_m^{max}}{C^i} \tag{12}$$

where l_m^{max} is the maximum length for packets in flow m.

Proof: The proof has been omitted due to space constraints, and is presented in [8]. ■

Hence, if $\sum_{m \in c^i} r_m \leq C^i$ and $\forall m \in c^i : \phi_m^i = r_m$, then SCFQ scheduling algorithm belongs to GR for flow f with $\beta^i = \sum_{m \in c^i \wedge m \neq f} \frac{l_m^{max}}{C^i}$. Note also that theorem 1 provides a tight bound over all rate assignments. An arrival sequence in which the upper bound is realized can be constructed from the proof steps.

3 Determining End-to-End Delay Bounds

3.1 Method

A simple method for determining a bound on end-to-end packet delays is to consider each server in isolation, and compute the summation of the maximum delay at each server along the path from the source to the destination [2, 3]. Though this may be the only feasible approach for many scheduling algorithms, it has the following drawbacks:

- Due to the variability in the delay experienced by packets at a server, the shape of the traffic gets distorted as it traverses through the network. Therefore, even if a source traffic specification is known at the first server on the path of the flow, it is difficult to determine the specification at a server further down on the path. This makes determining the delay at each of the server difficult.

- In many service disciplines, if a packet experiences a high delay at a server, it may experience a lower delay at the next server along the path. If each server is considered in isolation, the dependence between the delay experienced by packets at different servers is not accounted for, and hence the bound on the delay may be very conservative.

In this section, we address these limitations by presenting a method for determining end-to-end delay bound by analyzing the path as a whole. If the scheduling algorithm at each of the server on the path of the flow belongs to GR, then the delay guarantee of the algorithm can be exploited to determine end-to-end delay bound. This was illustrated in [10] for the Virtual Clock scheduling algorithm in Burst Scheduling networks for a particular source specification. Our key insight is that by deriving a delay guarantee for a network of servers, the problem of determining end-to-end delay can be reduced to the problem of determining delay at a single server.

Let K be the total number of servers along the path of a flow, and let the i^{th} server on the path be denoted by i. Also, let server 0 be the source and server $K+1$ be the destination. Since server K guarantees that packet p_f^j will be transmitted by $GRC^K(p_f^j) + \beta^K$ and the packet arrives at the first server at time $A^1(p_f^j)$, the end-to-end delay of p_f^j is given by

$$d_f^j \le GRC^K(p_f^j) + \alpha^K - A^1(p_f^j) \tag{13}$$

where $\alpha^K = \beta^K + \tau^{K,K+1}$ and $\tau^{K,K+1}$ is the propagation delay between server K and the destination.

Observe that $GRC^K(p_f^j)$ depends on $A^K(p_f^j)$, which in turn depends on $GRC^{K-1}(p_f^j)$. Applying this argument recursively $GRC^K(p_f^j)$ can be related to $GRC^1(p_f^j)$. Since $GRC^1(p_f^j)$ is completely determined by the arrival characteristics of the source and the rate associated with the flow, the end-to-end delay can be determined if source specification is known. In what follows, we first relate $GRC^{i+1}(p_f^j)$ to $GRC^i(p_f^j)$ and then relate the end-to-end delay to $GRC^1(p_f^j)$.

Lemma 1 *If the scheduling algorithm at server i belongs to GR for flow f, then*

$$GRC^{i+1}(p_f^j) \le GRC^i(p_f^j) + \max_{k \in [1..j]} \frac{l_f^k}{r_f} + \alpha^i \quad j \ge 1 \tag{14}$$

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

Proof: The proof is by induction on j.
Base Case: $j = 1$

$$GRC^{i+1}(p_f^1) = A^{i+1}(p_f^1) + \frac{l_f^1}{r_f} \tag{15}$$

Since scheduling algorithm at server i belongs to GR for flow f, $A^{i+1}(p_f^1) \le GRC^i(p_f^1) + \alpha^i$. Hence

$$GRC^{i+1}(p_f^1) \quad \le \quad GRC^i(p_f^1) + \frac{l_f^1}{r_f} + \alpha^i \tag{16}$$

$$\leq \quad GRC^i(p_f^1) + \max_{k\in[1..1]} \frac{l_f^k}{r_f} + \alpha^i \tag{17}$$

Therefore (14) holds for $j = 1$

Induction Hypothesis: Assume (14) holds for $1 \leq j \leq m$.

Induction: We need to show (14) holds for $1 \leq j \leq m+1$.

$$GRC^{i+1}(p_f^{m+1}) = max\{A^{i+1}(p_f^{m+1}), GRC^{i+1}(p_f^m)\} + \frac{l_f^{m+1}}{r_f} \tag{18}$$

Since scheduling algorithm at server i belongs to GR for flow f, $A^{i+1}(p_f^{m+1}) \leq GRC^i(p_f^{m+1}) + \alpha^i$. Hence

$$GRC^{i+1}(p_f^{m+1}) \leq max\{GRC^i(p_f^{m+1}) + \alpha^i, GRC^{i+1}(p_f^m)\} + \frac{l_f^{m+1}}{r_f} \tag{19}$$

Thus, there are two cases to consider:

1. If $GRC^i(p_f^{m+1}) + \alpha^i > GRC^{i+1}(p_f^m)$, then from (19) we get

$$GRC^{i+1}(p_f^{m+1}) \leq GRC^i(p_f^{m+1}) + \frac{l_f^{m+1}}{r_f} + \alpha^i \tag{20}$$

   Since $\max_{k\in[1..m+1]} \frac{l_f^k}{r_f} \geq \frac{l_f^{m+1}}{r_f}$,

$$GRC^{i+1}(p_f^{m+1}) \leq GRC^i(p_f^{m+1}) + \max_{k\in[1..m+1]} \frac{l_f^k}{r_f} + \alpha^i \tag{21}$$

2. If $GRC^i(p_f^{m+1}) + \alpha^i \leq GRC^{i+1}(p_f^m)$, then from (19) we get

$$GRC^{i+1}(p_f^{m+1}) \leq GRC^{i+1}(p_f^m) + \frac{l_f^{m+1}}{r_f} \tag{22}$$

   Using induction hypothesis we get,

$$GRC^{i+1}(p_f^{m+1}) \leq GRC^i(p_f^m) + \max_{k\in[1..m]} \frac{l_f^k}{r_f} + \frac{l_f^{m+1}}{r_f} + \alpha^i \tag{23}$$

   Since $GRC^i(p_f^{m+1}) \geq GRC^i(p_f^m) + \frac{l_f^{m+1}}{r_f}$ and $\max_{k\in[1..m+1]} \frac{l_f^k}{r_f} \geq \frac{l_f^{m+1}}{r_f}$,

$$GRC^{i+1}(p_f^{m+1}) \leq GRC^i(p_f^{m+1}) + \max_{k\in[1..m+1]} \frac{l_f^k}{r_f} + \alpha^i \tag{24}$$

From (21), (24) and the induction hypothesis, we conclude that (14) holds for $1 \leq j \leq m+1$. Hence the lemma follows. ■

Theorem 2 *If the scheduling algorithm at each of the servers on the path of a flow belongs to GR for flow f, then the end-to-end delay of packet p_f^j denoted by d_f^j is given by*

$$d_f^j \le GRC^1(p_f^j) - A^1(p_f^j) + (K-1) \max_{n \in [1..j]} \frac{l_f^n}{r_f} + \sum_{n=1}^{n=K} \alpha^n \tag{25}$$

where $\alpha^n = \beta^n + \tau^{n,n+1}$ and K is the number of servers on the path of the flow.

Proof: Since server K guarantees that packet p_f^j will be transmitted by time $GRC^K(p_f^j) + \beta^K$ and the packet arrives at the first node at time $A^1(p_f^j)$,

$$d_f^j \le GRC^K(p_f^j) + \alpha^K - A^1(p_f^j) \tag{26}$$

Since the scheduling algorithm at each server on the path of the flow belongs to GR, by repeated application of lemma 1, we conclude that:

$$GRC^K(p_f^j) \le GRC^1(p_f^j) + (K-1) \max_{n \in [1..j]} \frac{l_f^n}{r_f} + \sum_{n=1}^{n=K-1} \alpha^n \tag{27}$$

The theorem follows from (26) and (27). ■

Notice that $\alpha^n = \beta^n + \tau^{n,n+1}$. Hence, it is completely characterized by the scheduling algorithm and the propagation delay in the network. The remaining terms in (25) depend on the source traffic specifications, and are evaluated in the next section.

3.2 Source Traffic Specifications

A source traffic is specified by characterizing the number of bits that arrive at the network over an interval of time. Let $AP_f^i(t_1, t_2)$ be a function that denotes the flow f bits that arrive in the interval $[t_1, t_2]$ at server i. The bits of a packet are considered to have arrived only after the arrival of the last bit of the packet. Hence, the arrival function consists of an impulse at each packet arrival instants and is right continuous. Also $AP_f^i(t,t)$ is the length of the packet that arrives at time instant t.

As may be evident from theorem 2, to determine an upper bound on end-to-end packet delays, we must relate the guaranteed rate clock value of a packet to its arrival time at the first server. To achieve this objective, we define set S_f^i for flow f at server i as follows:

$$S_f^i = \{n | n > 0 \wedge GRC^i(p_f^{n-1}) \le A^i(p_f^n)\}$$

From the definitions of guaranteed rate clock and set S_f^i, it can be shown that for each packet p_f^j

$$GRC^i(p_f^j) = A^i(p_f^k) + \sum_{n=0}^{n=j-k} \frac{l_f^{k+n}}{r_f} \tag{28}$$

where $k \le j$ is the largest integer belonging to set S_f^i. Since $\sum_{n=0}^{n=j-k} l_f^{k+n} = AP_f^i(A^i(p_f^k), A^i(p_f^j))$, we get

$$GRC^i(p_f^j) = A^i(p_f^k) + \frac{AP_f^i(A^i(p_f^k), A^i(p_f^j))}{r_f} \tag{29}$$

Using theorem 2 and (29), we will now determine deterministic end-to-end delay bound for Leaky Bucket sources (a deterministic source characterization) and upper bound on the tail distribution of the delay for EBB sources (a stochastic characterization).

3.2.1 Leaky Bucket

A flow f conforms to Leaky Bucket [12] with parameters (σ_f, r_f) if

$$AP_f(t_1, t_2) \leq \sigma_f + r_f(t_2 - t_1) \quad t_1 \leq t_2, \;\; t_1 \geq 0 \tag{30}$$

Theorem 3 *If flow f conforms to a Leaky Bucket with parameters (σ_f, r_f) and the scheduling algorithm at each of the server on the path of a flow belongs to GR for the flow, then the end-to-end delay of packet p_f^j denoted by d_f^j is given by*

$$d_f^j \leq \frac{\sigma_f + (K-1)\max_{n\in[1..j]} l_f^n}{r_f} + \sum_{n=1}^{n=K} \alpha^n \tag{31}$$

where $\alpha^n = \beta^n + \tau^{n,n+1}$ and K is the number of servers on the path of the flow.

Proof: Let $k \leq j$ be largest integer belonging to set S_f^1. Clearly, such a k must exist. Since flow f conforms to Leaky Bucket specification, we get

$$AP_f^1(A^1(p_f^k), A^1(p_f^j)) \leq \sigma_f + r_f(A^1(p_f^j) - A^1(p_f^k)) \tag{32}$$

Hence, from (29) we get,

$$GRC^1(p_f^j) \leq \frac{\sigma_f}{r_f} + A^1(p_f^j) \tag{33}$$

which implies

$$GRC^1(p_f^j) - A^1(p_f^j) \leq \frac{\sigma_f}{r_f} \tag{34}$$

Theorem 3 follows from (34) and theorem 2. ■

We note that the if the scheduling algorithm used at each of the servers is either of Virtual Clock and PGPS , then bound in (31) is tighter than the bound in [11, 12] for Rate Proportional Processor Sharing (RPPS) rate assignment of PGPS networks when the sources conform to Leaky Bucket.

3.2.2 Exponentially Bounded Burstiness

A flow f conforms to an Exponentially Bounded Burstiness (EBB) [15] process with parameters $(r_f, \Lambda_f, \gamma_f)$, if

$$Pr\left(AP_f^i(t_1, t_2) \geq r_f(t_2 - t_1) + x\right) \leq \Lambda_f e^{-\gamma_f x} \quad x \geq 0 \;\; t_1 \leq t_2 \tag{35}$$

Theorem 4 *If flow f conforms to EBB with parameters $(r_f, \Lambda_f, \gamma_f)$ and the scheduling algorithm at each of the server on the path of a flow belong to GR for the flow , then the end-to-end delay of packet p_f^j denoted by d_f^j is given by*

$$Pr\left(d_f^j \geq x + (K-1)\frac{\max_{n\in[1..j]} l_f^n}{r^f} + \sum_{n=1}^{n=K} \alpha^n\right) \leq \Lambda_f e^{-\gamma_f x r_f} \quad x \geq 0 \tag{36}$$

where $\alpha^n = \beta^n + \tau^{n,n+1} and$ K is the number of servers on the path of the flow.

Proof: Let $k \leq j$ be the largest integer belonging to set S_f^1. Clearly such a k must exist. By the definition of EBB process, we have

$$Pr\left(AP_f^1(A^1(p_f^k), A^1(p_f^j)) \geq r_f(A^1(p_f^j) - A^1(p_f^k)) + y\right) \leq \Lambda_f e^{-\gamma_f y} \quad y \geq 0 \quad (37)$$

which implies

$$Pr\left(\frac{AP_f^1(A(p_f^k), A^1(p_f^j))}{r_f} + A^1(p_f^k) - A^1(p_f^j)\frac{y}{r_f}\right) \leq \Lambda_f e^{-\gamma_f y} \quad y \geq 0 \qquad (38)$$

From (29 we have

$$Pr\left(GRC^1(p_f^j) - A^1(p_f^j) \geq \frac{y}{r_f}\right) \leq \Lambda_f e^{-\gamma_f y} \quad y \geq 0 \qquad (39)$$

The theorem follows from (39) and theorem 2. ■

3.3 Discussion

Theorem 2 demonstrates that a heterogeneous network of servers, each of which employs a scheduling algorithm in the GR class, can provide an upper bound on end-to-end packet delays for heterogeneous sources. Additionally, the salient features of the class of GR scheduling algorithms and the method for determining end-to-end delay bounds include:

- An end-to-end delay bound can be determined for any source traffic specification for which a bound on difference between the guaranteed rate clock and the arrival time of a packet at the first server can be determined. Since this difference is only a function of the source traffic characteristics and the rate associated with the flow, it is simple to determine an end-to-end delay bound. Moreover, this difference can be interpreted as delay experienced by a packet at a single queue server with capacity r_f. Hence, queuing analysis can also be used to determine a bound on the tail distribution of delays experienced by packets for a conventional stochastic process characterization.

- A source can determine end-to-end delays without specifying the shape of the traffic to the network by keeping track of guaranteed rate clock values associated with it's flow at the first server. Such a capability is important as the source may not have a good characterization of the traffic or the characterization may not be known a priori. Moreover, even if the characterization is known, it may not conform to the set of characterizations supported by the network. Finally, this capability allows the source to decide when to renegotiate the reserved rate.

The method derives it's simplicity from the constraint placed on the rate assignments in the scheduling algorithms. Note that the constraint we have placed on rate assignments for PGPS so that it belongs to GR is similar to Rate Proportional Processor Sharing (RPPS) [12]. Such constraints allow each flow to be considered in isolation and hence facilitate the determination of delay bounds for each flow independently. If this constraint is relaxed to derive delay bounds for more general rate assignments, assumptions about the behavior of the other sources in the network may have to be made. In such scenarios, however, the QoS guarantees provided to each flow may be conditional since sources may be greedy and not conform to the specifications [13]. Moreover, if hardware traffic enforcement mechanisms are used, the probability of failure of one of the numerous pieces of enforcement hardware may not be negligible, thereby weakening the guarantees. Hence, we believe that the constraints on rate assignments described above are not a limitation, but in fact are highly desirable.

4 Conclusions

In this paper, we have defined a class of Guaranteed Rate scheduling algorithms. The GR class includes Virtual Clock, Self Clocked Fair Queuing and Packet-by-Packet Generalized Processor Sharing. For networks that employ scheduling algorithms belonging to GR, we have presented a method for determining an upper bound on end-to-end packet delays. The method facilitates determining end-to-end delay for a variety of sources. We have illustrated the method by determining end-to-end delay for sources conforming to Leaky Bucket and Exponentially Bounded Burstiness. The delay bounds that we have derived are parameterized by the scheduling algorithms which when instantiated with Virtual Clock and SCFQ scheduling algorithms, lead to many new results.

REFERENCES

[1] D.D. Clark, S. Shenker, and L. Zhang. Supporting Real-Time Applications in an Integrated Services Packet Network. In *Proceedings of ACM SIGCOMM*, pages 14–26, August 1992.

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

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

[4] 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.

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

[6] S.J. Golestani. A Framing Strategy for Congestion Management. *IEEE Journal on Selected Areas in Communications*, pages 1064–1077, September 1991.

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

[8] P. Goyal, S.S. Lam, and H.M. Vin. Determining End-to-End Delay Bounds in Heterogeneous Networks. Technical report, Department of Computer Sciences, University of Texas at Austin, (available by anonymous ftp from ftp.cs.utexas.edu in directory pub/multimedia), March 1995.

[9] C.R. Kalmanek, H. Kanakia, and S. Keshav. Rate Controlled Servers for Very High-Speed Networks. In *Proceedings of IEEE GLOBECOM'90, San Diego, CA*, pages 300.3.1–300.3.9, December 1990.

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

[11] 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.

[12] 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.

[13] S. Shenker. Making Greed Work in Networks: A Game-Theoretic Analysis of Switch Service Disciplines. In *Proceedings of ACM SIGCOMM*, pages 47–57, 1994.

[14] G.G. Xie and S.S. Lam. Delay Guarantee of Virtual Clock Server. Technical Report TR-94-24, Dept. of Computer Sciences, UT-Austin, October 1994. Presented at 9th IEEE Workshop on Computer Communications, October 1994.

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

[16] H. Zhang and D. Ferrari. Rate Controlled Static Priority Queueing. In *Proceedings of INFOCOM*, volume 2, pages 227–236, 1993.

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

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