Generalized Guaranteed Rate Scheduling Algorithms: A Framework

Pawan Goyal and **Harrick M. Vin**

Department of Computer Sciences
University of Texas at Austin
Taylor Hall 2.124
Austin, Texas 78712-1188

E-mail: {pawang,vin}@cs.utexas.edu, Telephone: (512) 471-9732, Fax: (512) 471-8885

Citation: *Technical Report TR-95-30, Department of Computer Sciences, University of Texas at Austin*

This research was supported in part by an IBM Graduate Fellowship, the National Science Foundation (Research Initiation Award CCR-9409666), NASA, Mitsubishi Electric Research Laboratories (MERL), Sun Microsystems Inc., and the University of Texas at Austin.

Generalized Guaranteed Rate Scheduling Algorithms: A Framework

Pawan Goyal and **Harrick M. Vin**

Distributed Multimedia Computing Laboratory,
Department of Computer Sciences, University of Texas at Austin
Taylor Hall 2.124, Austin, Texas 78712-1188
E-mail: @cs.utexas.edu, Telephone: (512) 471-9718

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 demonstrate that several work conserving and non-work conserving algorithms that either only allocate rate or separate rate and delay allocation belong to GR. 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 not only simple and valid in internetworks, but also leads to tighter results. We finally present architectural principles for the design of networks that employ scheduling algorithms in GR class. We demonstrate that GR class not only simplifies the design of networks, but also provides support for application with different characteristics and requirements.

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). Whereas the guaranteed bandwidth must be large enough to accommodate motion video and audio streams at acceptable resolutions, 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. 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 would 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 audio applications require constant bit rate, resource requirement of applications transmitting Variable Bit Rate (VBR) compressed video sequences varies significantly over time (Figure 1 shows the short-term as well as the long-term variations in the bit rate variation of a MPEG compressed video sequence).

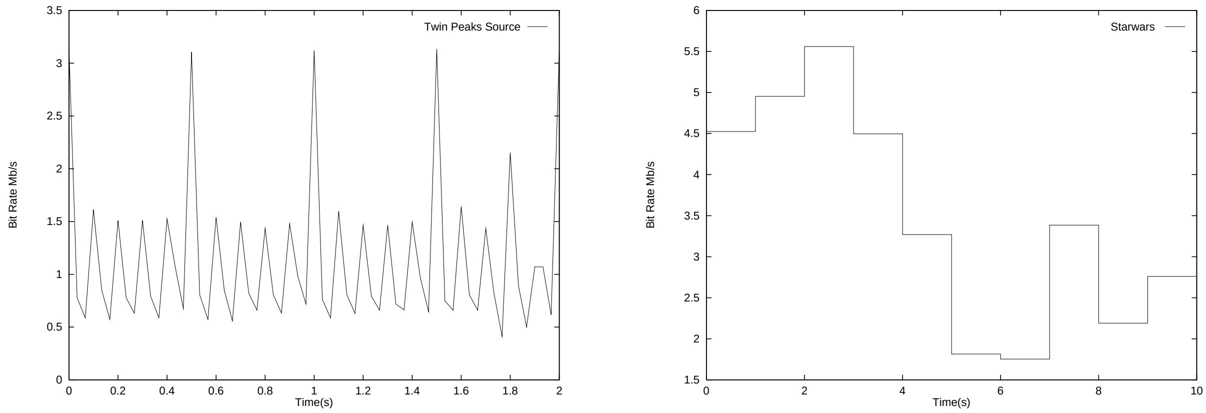

Figure 1 : Short-term and long-term bit rate variations in a MPEG compressed video sequence

- *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 (e.g., work conserving, non-work conserving, ones that separate delay and rate allocation, and the ones that only allocate rate). 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 channel, (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

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* [25]. 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, 10, 12, 18, 22, 24, 25, 26]. Most of these algorithms can be classified along two dimensions (see Table 1):

- *Work conserving vs. non-work conserving*: Work conserving algorithms schedule a packet whenever a packet is present in the system, and thereby yield better average delay performance [23]. Non-work conserving algorithms, on the other hand, reduce the buffer requirement in the network by keeping the link idle even when a packet is waiting to be served [23].

- *Rate-only vs. rate-and-delay allocation algorithms*: Whereas scheduling algorithms like Virtual Clock [25] only allocate rate, algorithms like Delay Earliest Due Date (Delay EDD) [14] separate the allocation of delay and rate. Such a separation is desirable for low throughput, low delay applications (e.g., interactive voice conferencing).

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 have also been proposed in the literature [6, 15, 18, 21]. However, most of the techniques for providing QoS guarantees have only investigated specific combinations of traffic specification and scheduling algorithms [2, 3, 15, 18, 21]. Recently, some efforts have partially addressed this limitation. For example, [8, 23] demonstrate that non-work conserving scheduling algorithms can interoperate to provide QoS guarantees to flows conforming to restricted traffic

	Rate Allocation	Delay Allocation
Work Conserving	Virtual Clock Packet-by-packet GPS (PGPS) Self Clocked Fair Queuing (SCFQ)	Delay EDD
Non-work Conserving	Hierarchical Round Robin Stop and Go Queuing	Jitter EDD Rate Controlled Static Priority Queuing

Table 1 : Classification of scheduling algorithms

specifications. Similarly, a general framework which enables a network employing work conserving or non-work conserving scheduling algorithms to provide QoS guarantees to flows conforming to any specification was introduced in [11]. However, none of these approaches provide QoS guarantees when variable rate may be allocated to the packets of a flow and packet fragmentation and/or reassembly may occur in the network.

1.3 Research Contributions of This Paper

In this paper, we develop a comprehensive framework for providing QoS guarantees by: (1) defining a class of generalized Guaranteed Rate (GR) scheduling algorithms that includes scheduling algorithms which may allocate variable rate to the packets of a flow, and (2) developing a general method for providing QoS guarantees in a heterogeneous networking environment.

The class of 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. This enables a single server employing a scheduling algorithm in GR to isolate flows as well as provide service guarantees to flows conforming to any specification. We demonstrate that the class of GR scheduling algorithms is broad, and includes work conserving and non-work conserving scheduling algorithms as well as algorithms that only allocate rate and those that separate rate and delay allocation (e.g. Virtual Clock, Packet-by-Packet Generalized Processor Sharing (PGPS), Self Clocked Fair Queuing (SCFQ), Delay Earliest Due Date (EDD), Jitter EDD, and Rate Controlled Static Priority Queuing). 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 also belong to GR. To prove that a scheduling algorithm belongs to GR, we employ a proof methodology 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 a broad class of preemptive and non-preemptive scheduling algorithms to show that the non-preemptive algorithm belongs to GR. This methodology not only simplifies the proofs, but also leads to new results for real-time scheduling algorithms. Moreover, it 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, the resulting non-work conserving algorithm also belongs to GR (this leads to the definition of several scheduling algorithms). Furthermore, such rate control elements do not change the delay guarantee of the scheduling algorithm.

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 present a method for deriving the delay guarantee of a network of servers each of which employs a scheduling algorithm in the GR class. This method enables the derivation of delay guarantee for a network of servers even 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 in the network. We utilize the delay guarantee of a network of servers to obtain an upper bound on end-to-end delay. We demonstrate that our method of deriving delay guarantee for a network of servers reduces the problem of determining end-to-end delay bound to a single server problem. We illustrate the end-to-end delay bound computation for flows conforming to Leaky Bucket, Exponentially Bounded Burstiness and Flow Specification [15]. We demonstrate that our method for determining these bounds is not only simple, but also leads to tighter results (e.g., it significantly improves upon the delay bound presented in [17] for flows conforming to Leaky Bucket in PGPS networks).

Finally, based on the properties of GR class, we present architectural principles for designing networks that provide guaranteed deterministic QoS. We demonstrate that GR class simplifies the design of a network while enabling it to support application with different characteristics and requirements.

The rest of the paper is organized as follows: In Section 2, we define the class of 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, and then present the architectural principles for designing networks that provide guaranteed service in Section 5. Finally, Section 6 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* [20]) to a packet of a flow based on its *expected arrival time*. In [11], we defined the delay guarantee of a packet by associating a *guaranteed rate clock* value with each packet. Moreover, we defined a class of guaranteed rate scheduling algorithms 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 GR 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 associated with p_f^j at server i (observe that each packet may be associated with 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:

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

where $GRC^i(p_f^0, r_f^{0,i}) = 0$. Observe that the first term in the right hand side of (1) can be interpreted as the expected arrival time and the second term as the deadline of a packet. 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 [1, 4, 15, 19], 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. 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, in the proof methodology that we employ to show that a scheduling algorithm belongs to GR, 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 to show that the scheduling algorithm belongs to GR. In what follows, we establish the general relationship between a broad class of preemptive and non-preemptive scheduling algorithms.

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 (referred to as a cell).

Observe that non-preemptive algorithms are a subset of partially preemptive algorithms. Moreover, partially preemptive algorithms would be helpful to define scheduling algorithms suitable for servers where fragmentation may occur. 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 *If PS is a work conserving preemptive scheduling algorithm, PPS its equivalent partially-preemptive scheduling algorithm and the priority assignment of a packet is not changed dynamically, then*

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

where $L_{PS}(p^j)$ and $L_{PPS}(p^j)$ denote the time a packet leaves the server when PS and PPS scheduling algorithms are employed, respectively. Also, $\hat{l}_{max}$ is the maximum length of a cell and the C is the capacity of the server.

Proof: Observe that since the PS and its equivalent PPS are work conserving, their busy periods are the same. Hence, it is sufficient to show that (2) holds for all the packets served in a busy period. Let a busy period begin at time t_0 and the packets be indexed by the order in which they complete service in the PPS, that is the i^{th} packet to complete service in PPS is packet p^i.

The proof is by contradiction (the structure of the proof is similar to the proof of Theorem 1 of [16] and Theorem 1 of [18]). Let (2) not hold for packet p^j. There are two cases to consider:

- A cell of packet p^n such that $L_{PS}(p^n) > L_{PS}(p^j)$ is served in the interval $[t_0, L_{PPS}(p^j)]$: Let t_1 be the largest time at which such a cell is scheduled. Also, let t_2 be the least time greater than t_1 such that if a cell of packet p^k is scheduled in the interval $[t_2, L_{PPS}(p^j)]$, then $L_{PS}(p^k) \leq L_{PS}(p^j)$. Let S be the set of packets which have at least one cell scheduled in $[t_2, L_{PPS}(p^j)]$. Since PPS schedules cells based on the priority of the packets and the priority is not changed dynamically, all the packets in S must have arrived after t_1, i.e.,

  $$\min_{k \in S}\{A(p^k)\} > t_1$$

  Hence, all the cells of packets in S are scheduled after t_1 by PS and PPS. Therefore,

  $$L_{PS}(p^j) > t_1 + \sum_{k \in S} \frac{l^k}{C}$$

  Since $t_2 - t_1 \leq \frac{\hat{l}_{max}}{C}$,

  $$L_{PS}(p^j) > t_2 - \frac{\hat{l}_{max}}{C} + \sum_{k \in S} \frac{l^k}{C}$$

  Since $t_2 + \sum_{k \in S} \frac{l^k}{C} = L_{PPS}(p^j)$

  $$\begin{aligned} L_{PS}(p^j) &> L_{PPS}(p^j) - \frac{\hat{l}_{max}}{C} \\ \frac{\hat{l}_{max}}{C} &> L_{PPS}(p^j) - L_{PS}(p^j) \end{aligned}$$

  which is a contradiction.

- No cell of packet p^n such that $L_{PS}(p^n) > L_{PS}(p^j)$ is served in the interval $[t_0, L_{PPS}(p^j)]$: Since set of cells served by PPS is a subset of cells served by PS, $L_{PPS}(p^j) \leq L_{PS}(p^j)$ which is a contradiction.

■

Since the only restriction on the scheduling algorithm is that it should not change the priority assignment of a packet dynamically, Theorem 1 can be employed for a wide variety of scheduling algorithms. In particular, it can be employed for Virtual Clock, PGPS, Delay EDD, Earliest Deadline First, Rate monotonic algorithm, and Rate controlled static priority queuing. The Theorem when applied to these algorithms, leads to several new results. Specifically, it leads to new results for Earliest Deadline First and Rate monotonic scheduling algorithms which are widely used for real-time scheduling of processes. Partially preemptive and non-preemptive equivalents of these algorithms enable a scheduler to reduce the context switch overhead.

We now use the relationship between preemptive and non-preemptive scheduling algorithms, to show that most of the work conserving and non-work conserving scheduling algorithms that either only allocate 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 [10, 18]).

2.2.1.1 Virtual Clock

We define generalized Virtual Clock (VC) scheduling algorithm analogous to the Virtual Clock algorithm [25]. The generalized VC algorithm is defined as follows:

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

$$VC^i(p_f^j, r_f^{j,i}) = \max\{A^i(p_f^j), VC^i(p_f^{j-1}, r_f^{j-1,i})\} + \frac{l_f^j}{r_f^{j,i}} \quad j \geq 1 \tag{3}$$

   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:

$$R_f^i(t) = \begin{cases} r_f^{j,i} & \text{if } \exists j \ni \left(A^i(p_f^j) \leq t\right) \wedge \left(VC^i(p_f^{j-1}, r_f^{j-1,i}) < t \leq VC^i(p_f^j, r_f^{j,i})\right) \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

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 time at which the transmission of packet p_f^j is completed in preemptive VC, denoted by $L_{P-VC}^i(p_f^j)$ is:*

$$L_{P-VC}^i(p_f^j) \leq VC^i(p_f^j, r_f^{j,i})$$

Proof: It is easily observed from (3) and (4) that the cumulative length of all flow f packets that arrive in interval $[t_1, t_2]$ and have virtual clock value no greater than t_2, denoted by $AP_f(t_1, t_2)$, is given as:

$$AP_f(t_1, t_2) \leq \int_{t_1}^{t_2} R_f^i(t)dt$$

We now prove the theorem by contradiction. Let for packet p_f^j, $L_{P-VC}^i(p_f^j) > VC^i(p_f^j, r_f^{j,i})$. Also, let t_0 be the beginning of the busy period in which p_f^j is served and $t_2 = VC^i(p_f^j)$. Let t_1 be the least time smaller than t_2 during the busy period such that no packet with virtual clock value greater than t_2 is served in the interval $[t_1, t_2]$ (such a t_1 exists). Clearly, all the packets served in the interval $[t_1, t_2]$ arrive in this interval and have virtual clock value less than or equal to t_2. Since the server is busy in the interval $[t_1, t_2]$ and packet p_f^j is not serviced by t_2, we have:

$$\begin{aligned}
\sum_{n \in S} AP_n(t_1, t_2) &> C^i(t_2 - t_1) \\
\sum_{n \in S} \int_{t_1}^{t_2} R_n^i(t)dt &> C^i(t_2 - t_1) \\
\int_{t_1}^{t_2} \sum_{n \in S} R_n^i(t)dt &> C^i(t_2 - t_1)
\end{aligned}$$

Since the server capacity is not exceeded, $\sum_{n \in S} R_n^i(t) \leq C^i$. Hence, $\int_{t_1}^{t_2} \sum_{n \in S} R_n^i(t)dt \leq C^i(t_2 - t_1)$. This contradicts (5), and hence the theorem follows. ■

Since preemptive VC algorithm is work conserving and does not dynamically change the priority of a packet, Corollary 1 immediately follows from Theorems 1 and 2.

Corollary 1 *If a server's capacity is not exceeded, then the time at which the transmission of packet p_f^j is completed in generalized Virtual Clock, denoted by $L_{VC}^i(p_f^j)$ is:*

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

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, it is easily observed 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 rates to be allocated to packets of a flow as long as servers capacity is not exceeded. This is in contrast to the non-work conserving Burst Scheduling algorithm [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.

2.2.1.2 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 [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 associated with a constant ϕ_f^i at server i. To allocate variable rate to packets of a flow, we associate a constant $\phi_f^{j,i}$ 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 $\frac{\phi_f^{j,i} C^i}{\sum_{k \in b^i(t)} \phi_k^{a,i}} \geq r_f^{j,i}$ 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:

$$L_{GPS}^i(p_f^j) \leq \max\{A^i(p_f^j), L_{GPS}^i(p_f^{j-1})\} + \frac{l_f^j}{r_f^{j,i}} \quad j \geq 1 \tag{5}$$

Let $L_{GPS}^i(p_f^0) = GRC^i(p_f^0, r_f^{0,i}) = 0$. From (5) and (1), it can be shown that

$$L_{GPS}^i(p_f^j) \leq GRC^i(p_f^j, r_f^{j,i}) \quad j \geq 1 \tag{6}$$

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:

$$\frac{dv^i(t)}{dt} = \frac{C^i}{\sum_{k \in b^i(t)} \phi_k^{a,i}} \tag{7}$$

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

$$F^i(p_f^j, \phi_f^{j,i}) = \max\{v^i(A^i(p_f^j)), F^i(p_f^{j-1}, \phi_f^{j-1,i})\} + \frac{l_f^j}{\phi_f^{j,i}} \quad j \geq 1 \tag{8}$$

where $F^i(p_f^0, \phi_f^{0,i}) = 0$. If finish time of a packet is defined as above and $\hat{t}$ is the time at which packet p_f^j departs, then $v^i(\hat{t}) = F^i(p_f^j, \phi_f^{j,i})$. To observe this, consider packet p_f^j of flow f. Let $W_f^{j,i}(t)$ be the amount of service that packet p_f^j has received within time interval t after it begins service. From the definition of GPS, we get:

$$\frac{dW_f^{j,i}(t)}{dt} = \frac{C^i \phi_f^{j,i}}{\sum_{k \in b^i(t)} \phi_k^{a,i}} \tag{9}$$

Hence from (7) and (9), we get:

$$\frac{dv^i(t)}{dW_f^{j,i}(t)} = \frac{1}{\phi_f^{j,i}} \tag{10}$$

From (10) we conclude, if packet p_f^j begins service at virtual time $\hat{v}$, it will depart at virtual time $\hat{v} + \frac{l_f^j}{\phi_f^{j,i}}$. Since GPS serves packets of a flow in FCFS order, it is easy to observe from (8) that if $\hat{t}$ is the time at which packet p_f^j departs, then $v^i(\hat{t}) = F^i(p_f^j, \phi_f^{j,i})$. Since $v^i(t)$ is monotonically increasing, packets leave a GPS server in the increasing order of finish time. Hence, a scheduling algorithm that schedules packets in increasing order of the finish time will simulate GPS. However, as PGPS is non-preemptive, from (6) and Theorem 1 (or alternatively, Theorem 1 of [18]), we get

$$L_{PGPS}^i(p_f^j) \leq GRC^i(p_f^j, r_f^{j,i}) + \frac{l_{max}^i}{C^i} \quad j \geq 1$$

Hence, PGPS scheduling algorithm belongs to GR for flow f with $\beta^i = \frac{l_{max}^i}{C^i}$.

2.2.1.3 Self Clocked Fair Queuing

The Self Clocked Fair Queuing scheme, proposed in [10], 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:

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

   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 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.

Proof: Let set B_f^i be defined as follows.

$$B_f^i = \{n | n > 0 \wedge F^i(p_f^{n-1}, r_f^{n-1,i}) < v^i(A^i(p_f^n))\}$$

Let $k \leq j$ be largest integer in B_f^i. Also, let $v_1 = v^i(A^i(p_f^k))$ and $v_2 = F^i(p_f^j, r_f^{j,i})$. The set of packets served by the server in the virtual time interval $[v_1, v_2]$ can be partitioned into two sets:

- This set consists of packets p_n^m such that $F^i(p_n^m, r_n^{m,i}) \leq v_2$ and $\max\{v^i(A^i(p_n^m)), F^i(p_n^{m-1}, r_n^{m-1,i})\} \geq v_1$. Then, from the definition of $R_n^i(v)$ and $F^i(p_n^m, r_f^{m,i})$, we know that the cumulative length of such flow n packets served by the server in the virtual time interval $[v_1, v_2]$, denoted by $AP_n(v_1, v_2)$, is given as:

$$AP_f(v_1, v_2) \leq \int_{v_1}^{v_2} R_n^i(v)dv$$

  Hence, aggregate length of packets in this set, $\sum_{n \in S} AP_n(v_1, v_2)$, is given as:

$$\begin{aligned} \sum_{n \in S} AP_n(v_1, v_2) &\leq \sum_{n \in S} \int_{v_1}^{v_2} R_n^i(v)dv \\ &\leq \int_{v_1}^{v_2} \sum_{n \in S} R_n^i(v)dv \\ &\leq \int_{v_1}^{v_2} C^i dv \\ &\leq C^i(v_2 - v_1) \end{aligned}$$

  But $v_2 - v_1 = \sum_{n=0}^{n=j-k} \frac{l_f^{k+n}}{r_f^{k+n,i}}$. Hence,

$$\sum_{n \in S} AP_n(v_1, v_2) \leq C^i \sum_{n=0}^{n=j-k} \frac{l_f^{k+n}}{r_f^{k+n,i}}$$

- This set consists of packets p_n^m such that $F^i(p_n^m, r_n^{m,i}) \geq v_1$ and $\max\{v^i(A^i(p_n^m)), F^i(p_n^{m-1}, r_n^{m-1,i})\} < v_1$. At most one packet of all flows other than f can belong to this set. Consequently, the maximum aggregate of packets in this set is $\sum_{n \in S \wedge n \neq f} l_n^{max}$.

Hence, the aggregate length of packets served by the server in the interval $[v_1, v_2]$, is less than or equal to:

$$C^i \sum_{n=0}^{n=j-k} \frac{l_f^{k+n}}{r_f^{k+n,i}} + \sum_{n \in S \wedge n \neq f} l_n^{max}$$

Since packet p_f^j departs at system virtual time v_2, we get:

$$\begin{aligned} A^i(p_f^k) + \frac{C^i \sum_{n=0}^{n=j-k} \frac{l_f^{k+n}}{r_f^{k+n,i}} + \sum_{n \in S \wedge n \neq f} l_n^{max}}{C^i} &\geq L^i_{SCFQ}(p_f^j) \\ A^i(p_f^k) + \sum_{n=0}^{n=j-k} \frac{l_f^{k+n}}{r_f^{k+n,i}} + \sum_{n \in S \wedge n \neq f} \frac{l_n^{max}}{C^i} &\geq L^i_{SCFQ}(p_f^j) \end{aligned}$$

From (1) we get

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

■

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 [1]. 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., $l_f = l_f^j$. Then Delay-EDD is defined as follows:

1. If d_f^i is the delay bound for flow f at server i, then on arrival packet p_f^j is stamped with a deadline, denoted by $D^i(p_f^j)$, which is computed as follows:

$$D^i(p_f^j) = \max\{A^i(p_f^j), GRC^i(p_f^{j-1}, r_f^i)\} + d_f^i \tag{11}$$

2. Packets are served in the increasing order of deadline.

It was shown in [5, 26] 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 [13, 26] by requiring the clocks of the servers to be synchronized. We demonstrate that this is an unnecessary restriction by proving 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.

[1] Consistent Relative Session Treatment (CRST) rate assignment has been used in PGPS networks to separate rate and delay allocation. However, this rate assignment requires all the flows to conform to Leaky Bucket specification. Furthermore, it has been shown in [8] that Delay EDD has a larger schedulability region than CRST PGPS. Hence, we do not consider CRST rate assignment of PGPS.

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

$$\forall t > 0: \quad \sum_{n \in S} \max\{0, \left\lceil \frac{(t - d_n^i) r_n}{l_n} \right\rceil \frac{l_n}{C^i}\} \leq t \tag{12}$$

then the time at which the transmission of packet p_f^j is completed in preemptive Delay EDD, denoted by $L_{P-EDD}^i(p_f^j)$ is:

$$L_{P-EDD}^i(p_f^j) \leq D^i(p_f^j)$$

Proof: From (11) and (1), we conclude that the cumulative length of packets of flow f that arrive in the interval $[t_1, t_2]$ and have deadline less than or equal to t_2, denoted by $AP_f(t_1, t_2)$, is given as:

$$AP_f(t_1, t_2) \leq \left\lceil \frac{(t_2 - t_1 - d_f^i) r_f}{l_f} \right\rceil l_f \tag{13}$$

We now prove the theorem by contradiction. Let for packet p_f^j, $L_{P-EDD}^i(p_f^j) > D^i(p_f^j)$. Also, let t_0 be the beginning of the busy period in which p_f^j is served and $t_2 = D^i(p_f^j)$. Let t_1 be the least time less than t_2 during the busy period such that no packet with deadline greater than t_2 is served in the interval $[t_1, t_2]$ (such t_1 exists). Clearly, all the packets served in the interval $[t_1, t_2]$ arrive in this interval and have deadline less than or equal to t_2. Since the server is busy in the interval $[t_1, t_2]$ and packet p_f^j is not serviced by t_2, from (13) we have:

$$\sum_{n \in S} \max\{0, \left\lceil \frac{(t_2 - t_1 - d_n^i) r_n}{l_n} \right\rceil \frac{l_n}{C^i}\} > (t_2 - t_1) \tag{14}$$

Substituting $t = (t_2 - t_1)$ in (14) contradicts (12) and hence theorem follows. ■

Due to high computational complexity, it may not be feasible to employ (12) as schedulability test. Hence, conditions stronger than (12) which have lower computational complexity have been developed in [26]. Clearly, the theorem holds under the stronger conditions developed in [26] as well.

Since Delay EDD does not dynamically change the priority assignment of packets, the following corrollary is immediate from Theorem 1 and Theorem 4.

Corollary 2 *If (12) is satisfied, then the time at which transmission of packet p_f^j is completed in Delay-EDD, denoted by $L_{EDD}^i(p_f^j)$, is given as:*

$$L_{EDD}^i(p_f^j) \leq D^i(p_f^j) + \frac{l_{max}^i}{C^i}$$

where C^i is the capacity of the server and l_{max}^i is the maximum length of a packet serviced by server i.

To observe that Delay EDD belongs to GR, rewrite (11) as:

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

Hence, Delay EDD belongs to GR with $\beta^i = \frac{l_{max}^i}{C^i} - (\frac{l_f}{r_f^i} - d_f^i)$. If schedulability conditions for non-preemptive Delay EDD are used, then one can similarly show that β^i is given as $\beta^i = -(\frac{l_f}{r_f^i} - d_f^i)$.

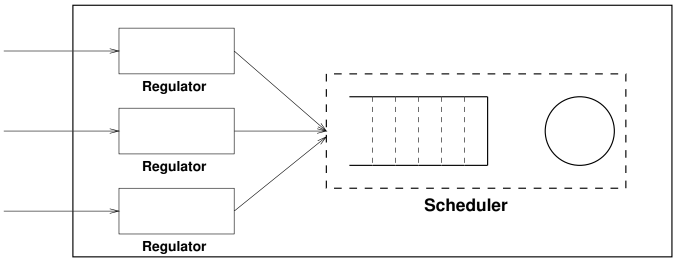

Figure 2 : Rate Controlled Service Disciplines

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 [23]. 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 2). 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^i_f. 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^i_f is the rate associated with flow f and l_f the length of packets, then the time at which a packet p^j_f departs the rate controller, denoted by $L^i_{RC}(p^j_f, r^i_f)$, is given as:

$$L^i_{RC}(p^j_f, r^i_f) \leq \max\{A^i(p^j_f), GRC^i(p^{j-1}_f, r^i_f)\} + \gamma^i \tag{15}$$

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

$$\begin{aligned} L^i_{RCSD}(p^j_f) &\leq \max\{A^i(p^j_f), GRC^i(p^{j-1}_f, r^i_f)\} + \gamma^i + d^i_f \\ &\leq \max\{A^i(p^j_f), GRC^i(p^{j-1}_f, r^i_f)\} + \frac{l_f}{r^i_f} - (\frac{l_f}{r^i_f} - \gamma^i - d^i_f) \end{aligned}$$

From (1) we get:

$$L^i_{RCSD}(p^j_f) \leq GRC^i(p^j_f, r^i_f) - (\frac{l_f}{r^i_f} - \gamma^i - d^i_f)$$

Therefore, rate controlled service disciplines which employ rate regulators consistent with (15) also belong to the GR class. The rate regulators for Jitter EDD and Rate controlled static priority queuing are consistent with (15) and hence they belong 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.

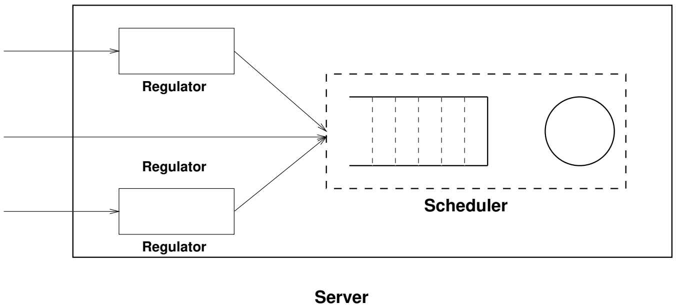

Figure 3 : Rate controller employed for a subset of flows

2.4.1 Scheduling in Presence of Fragmentation

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

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

Whereas scheduling each cell 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 cell, even if a server schedules packet, it can allow packet preemption to occur after the transmission of every cell, 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, PGPS, and Delay EDD can be defined using the definition of partially preemptive scheduling algorithms. Also using Theorem 1, Theorem 2, (6), and Theorem 4, we know that partially preemptive generalized Virtual Clock, PGPS and Delay EDD also belong to GR 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 cell served by server i. Self Clocked Fair Queuing is a non-preemptive algorithm by its definition and hence does not have an equivalent partially preemptive algorithm.

2.4.2 Effect of Rate Control

Rate controllers have conventionally been employed to enable a non-work conserving scheduling algorithm to guarantee bounded delay for packets [23]. In such scheduling algorithms, a rate controller is required to be employed for all the flows serviced by a server. Though rate control elements reduce the burstiness of a flow and consequently delay jitter and the buffer requirement of a flow, they increase the average delay of a flow as well as the implementation complexity. Whereas reduced buffer requirement and delay jitter would be desirable for some flows, low average delay may be desirable for other flows. Hence, it is desirable to be able to employ a rate controller for only a subset of the flows that may be serviced by a server (see Figure 3). In this section, we consider the effect of rate controllers in such scenarios.

Though a number of rate controllers which utilize the average rate have been studied [18, 23], rate controllers for flows which have variable rate have not received much attention. We define rate controllers for variable rate flows analogous to average rate controllers. Let $L^i_{RC}(p^j_f, r^{j,i}_f)$ denote the time at which packet p^j_f associated with rate $r^{j,i}_f$ leaves a rate controller. Then:

$$L^i_{RC}(p^j_f, r^{j,i}_f) \leq \max\{A^i(p^j_f), GRC^i(p^{j-1}_f, r^{j-1,i}_f)\} \tag{16}$$

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. Since a rate controller is not employed for all the flows, unlike rate controlled service disciplines, a scheduler may not be able to guarantee an upper bound on delay for all packets. However, 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 (16) is employed at server i for flow f, then:*

$$GRC^i(p_f^j, r_f^{j,i}) = \widehat{GRC}^i(p_f^j, r_f^{j,i})$$

where $GRC^i(p_f^j, r_f^{j,i})$ is the guaranteed rate clock value at the scheduler when no rate controller is employed for the flow and $\widehat{GRC}^i(p_f^j, r_f^{j,i})$ is the value when a rate controller is employed.

Proof: The proof is by induction on j and is presented in Appendix A.1. ■

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 (16) 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 Jitter EDD.

2.5 Summary

In the previous sections, we have defined the class of GR scheduling algorithms and shown that several work conserving and non-work conserving algorithms that either allocate only rate or separate rate and delay allocation belong to GR. We defined generalized Virtual Clock, PGPS, and SCFQ scheduling algorithms that can allocate variable rate to the packets of a flow. These algorithms are work conserving and do not require a flow regulator. 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 as well as combination of Delay EDD and Jitter EDD.

If a server employs any of these scheduling algorithm in GR, then it guarantees that packet p_f^j will be transmitted by $GRC^i(p_f^j, r_f^{j,i}) + \beta^i$. Since $GRC^i(p_f^j, r_f^{j,i})$, 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 [11]. 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_f^j will depart from the network by $GRC^K(p_f^j, r_f^{j,K}) + \beta^K$. This delay guarantee depends on the arrival process at the K^{th} server, i.e., $A^K(p_f^j)$. Though $A^K(p_f^j)$ depends on the arrival process of a flow, i.e., $A^1(p_f^j)$, due to the variability in the delay experienced by the packets of a flow, it may not always be possible to determine relationship between $A^K(p_f^j)$ and $A^1(p_f^j)$. Since, $A^1(p_f^j)$, and not $A^K(p_f^j)$, is always known, it is desirable to characterize the delay guarantee of the network of server such that it is determined by $A^1(p_f^j)$.

Observe that $GRC^K(p_f^j, r_f^{j,K})$ depends on $A^K(p_f^j)$, which in turn depends on $GRC^{K-1}(p_f^j, r_f^{j,K-1})$. 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.

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. Henceforth we will always refer to a single flow f and hence, for ease of presentation, we would 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 $GRC^{i+1}(p^j, \hat{r}^{j,i})$ and $GRC^i(p^j, \hat{r}^{j,i})$ where $GRC^i(p^j, \hat{r}^{j,i})$ denotes the guaranteed rate clock value computed using $\hat{r}^{k,i};\ \ k \leq j$, and $\hat{r}^{j,i}$ represents the bottleneck rate for p^j which is defined as[2]:

$$\hat{r}^{j,i} \leq \min\{r^{j,i}, r^{j,i+1}\}$$

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, $GRC^i(p^j, r^{j,i}) \leq GRC^i(p^j, \hat{r}^{j,i})$. Hence,

$$\begin{aligned} GRC^i(p^j, r^{j,i}) &= \max\{A^i(p^j), GRC^i(p^{j-1}, r^{j-1,i})\} + \frac{l^j}{r^{j,i}} \quad j \geq 1 \\ &\leq \max\{A^i(p^j), GRC^i(p^{j-1}, \hat{r}^{j-1,i})\} + \frac{l^j}{r^{j,i}} \quad j \geq 1 \end{aligned} \tag{17}$$

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 No Reassembly

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

$$GRC^{i+1}(p^j, \hat{r}^{j,i}) \leq GRC^i(p^j, \hat{r}^{j,i}) + \max_{k \in [1..j]} \frac{l^k}{r^{k,i}} + \alpha^i \quad j \geq 1 \tag{18}$$

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 $i+1$.

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

$$GRC^{i+1}(p^1, \hat{r}^{1,i}) = A^{i+1}(p^1) + \frac{l^1}{\hat{r}^{1,i}} \tag{19}$$

Since scheduling algorithm at server i belongs to GR for flow f, $A^{i+1}(p^1) \leq GRC^i(p^1, r^{1,i}) + \alpha^i$. From (19) and (17) we get:

$$\begin{aligned} GRC^{i+1}(p^1, \hat{r}^{1,i}) &\leq \max\{A^i(p^1), GRC^i(p^0, \hat{r}^{0,i})\} + \frac{l^1}{r^{1,i}} + \frac{l^1}{\hat{r}^{1,i}} + \alpha^i \\ &\leq \left(\max\{A^i(p^1), GRC^i(p^0, \hat{r}^{0,i})\} + \frac{l^1}{\hat{r}^{1,i}}\right) + \frac{l^1}{r^{1,i}} + \alpha^i \\ &\leq GRC^i(p^1, \hat{r}^{1,i}) + \max_{k \in [1..1]} \frac{l^k}{r^{k,i}} + \alpha^i \end{aligned}$$

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

Therefore (18) holds for $j = 1$
Induction Hypothesis: Assume (18) holds for $1 \leq j \leq m$.
Induction: We need to show (18) holds for $1 \leq j \leq m+1$. From (1) (definition of GRC), we get:

$$GRC^{i+1}(p^{m+1}, \hat{r}^{m+1,i}) = \max\{A^{i+1}(p^{m+1}), GRC^{i+1}(p^m, \hat{r}^{m,i})\} + \frac{l^{m+1}}{\hat{r}^{m+1,i}} \tag{20}$$

There are two cases to consider:

1. $A^{i+1}(p^{m+1}) > GRC^{i+1}(p^m, \hat{r}^{m,i})$: Since scheduling algorithm at server i belongs to GR for flow f, we know $A^{i+1}(p^{m+1}) \leq GRC^i(p^{m+1}, r^{m+1,i}) + \alpha^i$. Hence, from (20) we get:

$$GRC^{i+1}(p^{m+1}, \hat{r}^{m+1,i}) \leq GRC^i(p^{m+1}, r^{m+1,i}) + \alpha^i + \frac{l^{m+1}}{\hat{r}^{m+1,i}}$$

   Using (17) we get:

$$\begin{aligned} GRC^{i+1}(p^{m+1}, \hat{r}^{m+1,i}) &\leq \max\{A^i(p^{m+1}), GRC^i(p^m, \hat{r}^{m,i})\} + \frac{l^{m+1}}{r^{m+1,i}} + \alpha^i + \frac{l^{m+1}}{\hat{r}^{m+1,i}} \\ &\leq \left(\max\{A^i(p^{m+1}), GRC^i(p^m, \hat{r}^{m,i})\} + \frac{l^{m+1}}{\hat{r}^{m+1,i}}\right) + \frac{l^{m+1}}{r^{m+1,i}} + \alpha^i \\ &\leq GRC^i(p^{m+1}, \hat{r}^{m+1,i}) + \max_{k \in [1..m+1]} \frac{l^k}{r^{k,i}} + \alpha^i \end{aligned} \tag{21}$$

2. $A^{i+1}(p^{m+1}) \leq GRC^{i+1}(p^m, \hat{r}^{m,i})$: From (20) we get:

$$GRC^{i+1}(p^{m+1}, \hat{r}^{m+1,i}) \leq GRC^{i+1}(p^m, \hat{r}^{m,i}) + \frac{l^{m+1}}{\hat{r}^{m+1,i}}$$

   Using induction hypothesis we get,

$$\begin{aligned} GRC^{i+1}(p^{m+1}, \hat{r}^{m+1,i}) &\leq GRC^i(p^m, \hat{r}^{m,i}) + \max_{k \in [1..m]} \frac{l^k}{r^{k,i}} + \alpha^i + \frac{l^{m+1}}{\hat{r}^{m+1,i}} \\ &\leq \left(GRC^i(p^m, \hat{r}^{m,i}) + \frac{l^{m+1}}{\hat{r}^{m+1,i}}\right) + \max_{k \in [1..m]} \frac{l^k}{r^{k,i}} + \alpha^i \\ &\leq GRC^i(p^{m+1}, \hat{r}^{m+1,i}) + \max_{k \in [1..m+1]} \frac{l^k}{r^{k,i}} + \alpha^i \end{aligned} \tag{22}$$

From (21), (22) and the induction hypothesis, we conclude that (18) holds for $1 \leq j \leq m+1$. Hence the theorem follows. ■

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 (18) is $r^{j,i}$ instead of $\hat{r}^{j,i}$. This would enable us to derive tighter end-to-end delay bounds in section 4.

3.1.2 Packet Fragmentation

Let server i fragment a packet into cells. As we had mentioned in Section 2.4.1, server i may schedule packets or cells (i.e., packet fragments). Let us consider the two cases:

- If server i schedules packets (see Figure (4)), it may or may not allow packet transmission to be preempted after transmission of every cell. In either case, $GRC^i(p^j, \hat{r}^{j,i-1})$ can be related to $GRC^{i-1}(p^j, \hat{r}^{j,i-1})$ 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 cells (see Figure (5)), then guaranteed rate clock values of cells 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.

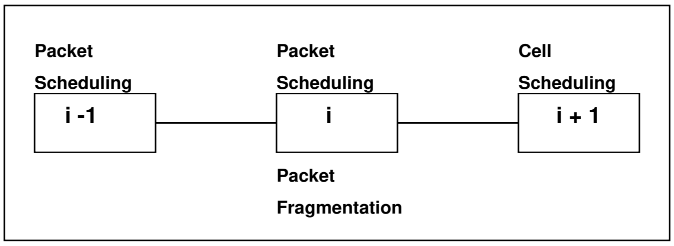

Figure 4 : Packet scheduling at the switch where fragmentation occurs

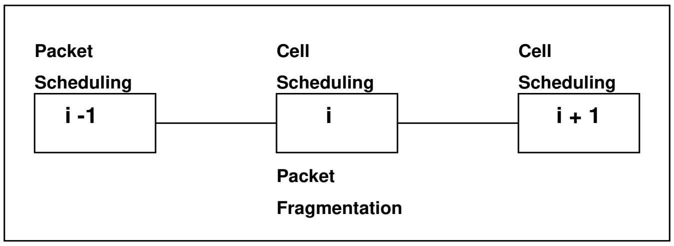

Figure 5 : Cell scheduling at the switch where fragmentation occurs

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 cells. Let packet p^j be fragmented into $\theta(j)$ cells, $p^{j,k}$ denote the k^{th} cell of p^j and let $l^{j,k}$ denote the length of cell $p^{j,k}$. Since the delay guarantee of a packet is determined by the delay guarantee of the last cell of a packet, we relate the guaranteed rate clock value of the last cell 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 cells, then*

$$GRC^{i+1}(p^{j,\theta(j)}, \widehat{r}^{j,i}) \leq GRC^{i}(p^{j}, \widehat{r}^{j,i}) + \max_{k\in[1..j]} \frac{l^k}{r^{k,i}} + \alpha^i \quad j \geq 1 \tag{23}$$

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 $i+1$.

Proof: The proof is presented in Appendix A.2. ■

3.1.3 Packet Reassembly

Let server $i+1$ perform reassembly of cells. 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*

$$GRC^{i+1}(p^{j}, \widehat{r}^{j,i}) \leq GRC^{i}(p^{j,\theta(j)}, \widehat{r}^{j,i}) + \max_{k\in[1..j]} \frac{l^k}{\widehat{r}^{k,i}} + \alpha^i \quad j \geq 1 \tag{24}$$

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 $i+1$.

Proof: The proof is presented in Appendix A.3. ■

Observe that the second term in the right hand side of (24) 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 (18) and (23).

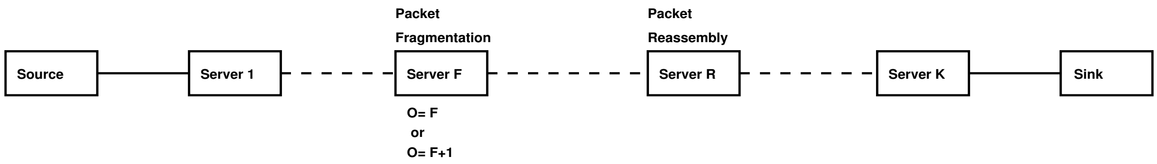

Figure 6 : The path configuration

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 6. In Figure 6 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 cells. If server F schedules packets, then the next server schedules cells 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*

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

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., $\widehat{r^j} = \min_{i \in [1..K]} r^{j,i}$.

Proof: Since $\widehat{r^j} \leq \widehat{r^{j,i}}$ for each server on the path of the flow and the scheduling algorithm at each server belongs to GR, Theorem 6 can be employed. By repeated application of Theorem 6:

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

Since server R does packet reassembly, using Theorem 8 we get:

$$GRC^R(p^j, \widehat{r^j}) \leq GRC^{R-1}(p^{j,\theta(j)}, \widehat{r^j}) + \max_{n \in [1..j]} \frac{l^n}{\widehat{r^n}} + \alpha^{R-1}$$

Using Theorem 6 again, we get:

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

Since server O schedules cells and server $O - 1$ schedules packets, from Theorem 7 we get:

$$GRC^O(p^{j,\theta(j)}, \widehat{r^j}) \leq GRC^{O-1}(p^j, \widehat{r^j}) + \max_{n \in [1..j]} \frac{l^n}{r^{n,O-1}} + \alpha^{O-1}$$

Using Theorem 6 again , we get:

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

The theorem follows from the above steps.

■

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

$$GRC^K(p^j, \widehat{r}^j) \le GRC^1(p^j, \widehat{r}^j) + \sum_{i=1}^{i=K-1} \max_{n \in [1..j]} \frac{l^n}{r^{n,i}} + \sum_{n=1}^{n=K-1} \alpha^n \tag{25}$$

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

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

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

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

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 (25) we get:

$$d^j \le \left(GRC^1(p^j, \widehat{r}^j) - A^1(p^j)\right) + \left(\sum_{i=1}^{i=K-1} \max_{n \in [1..j]} \frac{l^n}{r^{n,i}} - \left(\frac{l^j}{\widehat{r}^j} - \frac{l^j}{r^{j,K}}\right)\right) + \left(\sum_{n=1}^{n=K} \alpha^n\right) \tag{26}$$

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 algorithm 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 $r \le \widehat{r}$, then from [11] we get:

$$GRC^1(p^j, \widehat{r}) - A^1(p^j) \le \frac{\sigma}{\widehat{r}} \tag{27}$$

  Substituting (27) in (26) 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) [21], and $\widehat{r}$ is the minimum rate allocated to the packets of the flow such that $r \leq \widehat{r}$, then from [11] we get:

$$Pr\left(GRC^1(p^j, \widehat{r}) - A^1(p^j) \geq \frac{y}{\widehat{r}}\right) \leq \Lambda\, e^{-\gamma\, y} \quad y \geq 0 \tag{28}$$

  An upper bound on the tail distribution of the end-to-end delay is derived by substituting (28) in (26). 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:

$$GRC^1(p^j, \widehat{r^j}) - A^1(p^j) \leq \frac{l^j}{\widehat{r^j}} \tag{29}$$

  Substituting (29) in (26) 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 which reduce the jitter of VBR video.

Observe that 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 highly desirable in internetworks [23].

The above method not only determines an upper bound on end-to-end delay for any source specification in an internetwork in a conceptually simple manner but also leads to tighter results. Observe that if $r^{j,K} = \widehat{r^j}$ (assumed for ease of exposition) then the only variable term in (26), that depends on network and not flow characteristics is:

$$\sum_{i=1}^{i=K-1} \max_{n \in [1..j]} \frac{l^n}{r^{n,i}} \tag{30}$$

This term is smaller than other similar analysis in the literature in several ways. We illustrate the differences by comparing it with the 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 (30) is:

$$2\sum_{i=1}^{i=K-1} \frac{l^{max}}{\widehat{r}} \tag{31}$$

(31) is larger than (30) in several ways:

- The factor 2 makes (31) significantly larger than (30).
- Even when different rates may be allocated at different servers, the denominator in (31) is $\widehat{r}$. In contrast, the denominator in (30) 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, even after neglecting factor 2, (31) evaluates to 24.4 ms which is significantly larger than the 12.2 ms computed from (30).
- Whereas (31) does not quantify the effect of variable rate allocation to packets of a flow, (30) does.
- The numerator in (31) is l^{max} which is larger than the numerator, $\max_{n \in [1..j]} l^n$ (assuming a constant rate allocation for packets of a flow), in (30).

  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 (30), 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, (30) 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 (31) will always yield 45.7 ms.

Since (30) is the only variable term that depends on the network, these improvements are significant.

5 Architectural Principles

The class of GR scheduling algorithms has several desirable properties which simplifies the design of a network that provide guaranteed deterministic QoS while enabling it to support application with different characteristics and requirements. We now present a few important architectural principles, based on the properties of GR, for the design of networks employing scheduling algorithms in GR class.

- *A source is not required to specify the shape of the traffic to the network*: It has been argued in the literature that a network provides QoS guarantees such as packet delay and throughput based on the traffic specification of a source. However, as is evident from Theorem 9, a network employing scheduling algorithms in GR can provide a delay guarantee without requiring the source to specify the shape of the traffic. Furthermore, by keeping track of guaranteed rate clock values associated with its flow at the first server, a source can determine end-to-end delays without specifying the shape of the traffic to the network. Hence, a network can provide QoS guarantees without requiring a traffic specification. Such a network architecture is desirable as a 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 a network.

- *Policing of the traffic is not required*: Observe that the guaranteed rate clock values of a packet of a flow are independent of the behaviour of the other flows in the network. Consequently, the guarantees offered by a network employing scheduling algorithms in GR are independent of the behaviour of other flows, i.e., the network provides isolation between sources. Hence, in such networks, policing of traffic is not required.

  If a network employs scheduling algorithm which does not provide isolation between sources, then it may have to employ policing devices to guard against greedy sources. However, as the probability of failure of one of the numerous pieces of enforcement hardware that may be employed may not be negligible, the guarantees provided by such networks would be weak. Hence, a network which does not employ policing devices is not only simple but also provides stronger guarantees.

- *A source should be able to request buffer reservation in the network*: If a source does not specify the shape of the traffic to the network, then a network may not know the buffer space that should be reserved for a flow in order to provide the desired packet loss. A network, rather than requiring a source traffic specification and then deriving the buffer requirements, should allow a source to explicitly request buffer space. This would keep the design of the network simple and enable it to support applications with different characteristics and requirements.

- *A source should be able to request a network to employ rate controllers*: As Theorem 5 illustrates, in GR networks the worst case delay of packet does not change when one or more servers on the path of a flow employ a rate controller. Though rate controllers do not change the worst case end-to-end delay, they increase the average case delay while reducing the buffer requirement in the network. Instead of a network deciding the tradeoff between the buffer requirement and the average case delay for all the sources, a source should be provided with the flexibility of deciding the tradeoff. A source would have the flexibility if it can request a network to employ a rate controller.

- *GR networks should provide good average case performance as well*: As is evident from the end-to-end delay bound determination method, a GR network which is designed to provide only worst case guarantee, can do so by buffering all the data at the network periphery and serving it at the rate desired by the packets. However, such a method would increase the average case delay significantly and may be undesirable for applications requiring good average case performance. Hence, even though a network may not provide any guarantees, it is desirable to design it to provide good average case performance as well. If a source does not require good average case delay, it can reduce its buffer requirement by buffering the data or requesting a network to employ rate controllers on its path. Hence, if GR networks provide good average performance, they can support applications with different requirements.

6 Concluding Remarks

In this paper, we have defined the class of GR scheduling algorithms and demonstrated that several work conserving and non-work conserving algorithms that either allocate only rate or separate rate and delay allocation belong to GR. 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 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. Our method for determining these bounds is not only simple and valid in internetworks but also leads to tighter results. We finally presented architectural principles for the design of networks that employ scheduling algorithms in GR class. GR class not only simplifies the design of networks but also enables support for application with different characteristics and requirements.

The variable rate allocation algorithms that we have introduced in this paper, are suitable not only for supporting variable rate video but also for achieving dynamic link sharing objectives [7]. Such algorithms can also be employed for reducing the maximum delay incurred by bursty flows in a controlled manner in networks providing guarantees weaker than deterministic or statistical guarantees [1]. We expect to explore these benefits in our future work.

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

[6] 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.

[7] S. Floyd and V. Jacobson. Link-sharing and Resource Management Models for Packet Networks. *IEEE/ACM Transactions on Networking*, 3(4), August 1995.

[8] L. Georgiadis, R. Guerin, and V. Peris. Efficient Network QoS Provisioning Based on per Node Traffic Shaping. Technical Report RC 20064, IBM, T.J. Watson Research Division, May 1995.

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

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

[11] P. Goyal, S. S. Lam, and H. M. Vin. Determining End-to-End Delay Bounds In Heterogeneous Networks. In *Proceedings of the Workshop on Network and Operating System Support for Digital Audio and Video*, April 1995.

[12] 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.

[13] 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.

[14] S. Keshav. A Control-Theoretic approach to Flow Control. In *Proceedings of ACM SIGCOMM'91*, pages 3–15, 1991.

[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] A.K. Mok, P. Amarsinghe, M. Chen, S. Sutanthavibul, and K. Tanstisirivat. Synthesis of a Real-Time Message Processing System with Data-driven Timing Constraints. In *IEEE 7th Real-Time Systems Symposium*, pages 133–143, 1987.

[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] S. Shenker. Making Greed Work in Networks: A Game-Theoretic Analysis of Switch Service Disciplines. In *Proceedings of ACM SIGCOMM'94*, pages 47–57, 1994.

[20] 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.

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

[22] H. Zhang and D. Ferrari. Rate Controlled Static Priority Queueing. In *Proceedings of INFOCOM'93*, volume 2, pages 227–236, 1993.

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

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

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

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

A Appendix

A.1 Proof of Theorem 5

Proof: Let $A^i(p_f^j)$ be the arrival time of p_f^j at the rate controller when a rate controller is employed and at the scheduler when it is not. Also, let $\widehat{A}^i(p_f^j)$ be the arrival time of the packet at the scheduler (or equivalently, departure time at the rate controller) when a rate control element is employed. The proof is by induction on j.
Base Case: $j = 1$. As $GRC^i(p_f^0) = 0$, from (16) we know $A^i(p_f^j) = \widehat{A}^i(p_f^j)$. Hence we conclude:

$$\begin{aligned}\widehat{GRC}^i(p_f^1, r_f^{1,i}) &= \widehat{A}^i(p_f^1) + \frac{l_f^1}{r_f^{1,i}} \\ &= A^i(p_f^1) + \frac{l_f^1}{r_f^{1,i}} \\ &= GRC^i(p_f^1, r_f^{1,i})\end{aligned}$$

Therefore the theorem holds for $j = 1$.
Induction Hypothesis: Let us assume that the theorem holds for $1 \leq j \leq m$.
Induction: We now show that the theorem holds for $1 \leq j \leq m + 1$. From the definition of GRC:

$$\widehat{GRC}^i(p_f^{m+1}, r_f^{m+1,i}) = \max\{\widehat{A}^i(p_f^{m+1,i}), \widehat{GRC}^i(p_f^m, r_f^{m,i})\} + \frac{l_f^{m+1}}{r_f^{m+1,i}} \tag{32}$$

There are two cases to consider:

1. $\widehat{A}^i(p_f^{m+1}) \leq \widehat{GRC}^i(p_f^m, r_f^{m,i})$: From (32) we get:

$$\widehat{GRC}^i(p_f^{m+1}, r_f^{m+1,i}) = \widehat{GRC}^i(p_f^m, r_f^{m,i}) + \frac{l_f^{m+1}}{r_f^{m+1,i}}$$

   Using induction hypothesis we get:

$$\begin{aligned}\widehat{GRC}^i(p_f^{m+1}, r_f^{m+1,i}) &= GRC^i(p_f^m, r_f^{m,i}) + \frac{l_f^{m+1}}{r_f^{m+1,i}} \\ &= GRC^i(p_f^{m+1}, r_f^{m+1,i})\end{aligned} \tag{33}$$

2. $\widehat{A}^i(p_f^{m+1}) > \widehat{GRC}^i(p_f^m, r_f^{m,i})$: From induction hypothesis, $\widehat{GRC}^i(p_f^m, r_f^{m,i}) = GRC^i(p_f^m, r_f^{m,i})$. Hence, we conclude $\widehat{A}^i(p_f^{m+1}) > GRC^i(p_f^m, r_f^{m,i})$. But from (16) this implies $\widehat{A}^i(p_f^{m+1}) = A^i(p_f^{m+1})$. Hence, from (32) we get:

$$\begin{aligned}\widehat{GRC}^i(p_f^{m+1}, r_f^{m+1,i}) &= \widehat{A}^i(p_f^{m+1}) + \frac{l_f^{m+1}}{r_f^{m+1,i}} \\ &= A^i(p_f^{m+1}) + \frac{l_f^{m+1}}{r_f^{m+1,i}} \\ &= \max\{A^i(p_f^{m+1}), GRC^i(p_f^m, r_f^{m,i})\} + \frac{l_f^{m+1}}{r_f^{m+1,i}} \\ &= GRC^i(p_f^{m+1}, r_f^{m+1,i})\end{aligned} \tag{34}$$

From (33), (34) and the induction hypothesis, we conclude that the theorem holds for $1 \leq j \leq m + 1$. Hence the theorem follows. ■

A.2 Proof of Theorem 7

Proof: The proof is by induction on j.
Base Case: $j = 1$. Let $p^{1,o}$ be the last cell to have arrived before $p^{1,\theta(1)}$ such that $A^{i+1}(p^{1,o}) > GRC^{i+1}(p^{1,o-1}, \widehat{r}^{1,i})$. Then, from (1) (definition of GRC) we get:

$$GRC^{i+1}(p^{1,\theta(1)}, \widehat{r}^{1,i}) = A^{i+1}(p^{1,o}) + \sum_{k=o}^{k=\theta(1)} \frac{l^{1,k}}{\widehat{r}^{1,i}}$$

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

$$\begin{aligned} GRC^{i+1}(p^{1,\theta(1)}, \widehat{r}^{1,i}) &\leq GRC^i(p^1, r^{1,i}) + \sum_{k=o}^{k=\theta(1)} \frac{l^{1,k}}{\widehat{r}^{1,i}} + \alpha^i \\ &\leq A^i(p^1) + \frac{l^1}{r^{1,i}} + \sum_{k=o}^{k=\theta(1)} \frac{l^{1,k}}{\widehat{r}^{1,i}} + \alpha^i \end{aligned}$$

Since $\sum_{k=o}^{k=\theta(1)} \frac{l^{1,k}}{\widehat{r}^{1,i}} \leq \frac{l^1}{\widehat{r}^{1,i}}$, we get

$$\begin{aligned} GRC^{i+1}(p^{1,\theta(1)}, \widehat{r}^{1,i}) &\leq A^i(p^1) + \frac{l^1}{\widehat{r}^{1,i}} + \frac{l^1}{r^{1,i}} + \alpha^i \\ &\leq GRC^i(p^1, \widehat{r}^{1,i}) + \frac{l^1}{r^{1,i}} + \alpha^i \\ &\leq GRC^i(p^1, \widehat{r}^{1,i}) + \max_{k \in [1..1]} \frac{l^1}{r^{1,i}} + \alpha^i \end{aligned}$$

Therefore (23) holds for $j = 1$
Induction Hypothesis: Assume (23) holds for $1 \leq j \leq m$.
Induction: We need to show (23) holds for $1 \leq j \leq m+1$.

$$GRC^{i+1}(p^{m+1,\theta(m+1)}, \widehat{r}^{m+1,i}) = \max\{A^{i+1}(p^{m+1,\theta(m+1)}), GRC^{i+1}(p^{m+1,\theta(m+1)-1}, \widehat{r}^{m+1,i})\} + \frac{l^{m+1,\theta(m+1)}}{\widehat{r}^{m+1,i}} \tag{35}$$

Thus, there are two cases to consider:

1. $A^{i+1}(p^{m+1,\theta(m+1)}) > GRC^{i+1}(p^{m+1,\theta(m+1)-1}, \widehat{r}^{m+1,i})$: From (35) we get:

$$GRC^{i+1}(p^{m+1,\theta(m+1)}, \widehat{r}^{m+1,i}) \leq A^{i+1}(p^{m+1,\theta(m+1)}) + \frac{l^{m+1,\theta(m+1)}}{\widehat{r}^{m+1,i}}$$

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

$$GRC^{i+1}(p^{m+1,\theta(m+1)}, \widehat{r}^{m+1,i}) \leq GRC^i(p^{m+1}, r^{m+1,i}) + \frac{l^{m+1,\theta(m+1)}}{\widehat{r}^{m+1,i}} + \alpha^i$$

Hence, using (17) we get:

$$\begin{aligned} GRC^{i+1}(p^{m+1,\theta(m+1)}, \widehat{r}^{m+1,i} &\leq \max\{A^i(p^m), GRC^i(p^m, \widehat{r}^{m,i})\} + \frac{l^{m+1}}{r^{m+1,i}} + \frac{l^{m+1,\theta(m+1)}}{\widehat{r}^{m+1,i}} + \alpha^i \\ &\leq \left(\max\{A^i(p^m), GRC^i(p^m, \widehat{r}^{m,i})\} + \frac{l^{m+1,\theta(m+1)}}{\widehat{r}^{m+1,i}}\right) + \frac{l^{m+1}}{r^{m+1,i}} + \alpha^i \\ &\leq GRC^i(p^{m+1}, \widehat{r}^{m+1,i}) + \frac{l^{m+1}}{r^{m+1,i}} + \alpha^i \\ &\leq GRC^i(p^{m+1}, \widehat{r}^{m+1,i}) + \max_{k \in [1..m+1]} \frac{l^k}{r^{k,i}} + \alpha^i \end{aligned} \tag{36}$$

2. $A^{i+1}(p^{m+1,\theta(m+1)}) \leq GRC^{i+1}(p^{m+1,\theta(m+1)-1}, \widehat{r}^{m+1,i})$: From (35) we get:

$$GRC^{i+1}(p^{m+1,\theta(m+1)}, \widehat{r}^{m+1,i}) \leq GRC^{i+1}(p^{m+1,\theta(m+1)-1}, \widehat{r}^{m+1,i}) + \frac{l^{m+1,\theta(m+1)}}{\widehat{r}^{m+1,i}} \tag{37}$$

Let $p^{m+1,0}$ refer to $p^{m,\theta(m)}$. There are two cases to consider:

(a) $\forall k \in [1..\theta(m+1)] : A^{i+1}(p^{m+1,k}) \leq GRC^{i+1}(p^{m+1,k-1}, \widehat{r}^{m+1,i})$: From (37) we get:

$$GRC^{i+1}(p^{m+1,\theta(m+1)}, \widehat{r}^{m+1,i}) \leq GRC^{i+1}(p^{m,\theta(m)}, \widehat{r}^{m,i}) + \sum_{k=1}^{k=\theta(m+1)} \frac{l^{m+1,k}}{\widehat{r}^{m+1,i}}$$

Using induction hypothesis, we get:

$$\begin{aligned} GRC^{i+1}(p^{m+1,\theta(m+1)}, \widehat{r}^{m+1,i}) &\leq GRC^{i}(p^{m,\theta(m)}, \widehat{r}^{m,i}) + \max_{k \in [1..m]} \frac{l^k}{r^{k,i}} + \alpha^i + \frac{l^{m+1}}{\widehat{r}^{m+1,i}} \\ &\leq \left(GRC^{i}(p^{m,\theta(m)}, \widehat{r}^{m,i}) + \frac{l^{m+1}}{\widehat{r}^{m+1,i}}\right) + \max_{k \in [1..m]} \frac{l^k}{r^{k,i}} + \alpha^i \\ &\leq GRC^{i}(p^{m+1}, \widehat{r}^{m+1,i}) + \max_{k \in [1..m+1]} \frac{l^k}{r^{k,i}} + \alpha^i \end{aligned} \tag{38}$$

(b) $\exists k \in [1..\theta(m+1)] : A^{i+1}(p^{m+1,k}) > GRC^{i+1}(p^{m+1,k-1}, \widehat{r}^{m+1,i})$: Let o be the greatest integer less than or equal to $\theta(m+1)$ such that $A^{i+1}(p^{m+1,o}) > GRC^{i+1}(p^{m+1,o-1}, \widehat{r}^{m+1,i})$. Then, from (37) we get:

$$\begin{aligned} GRC^{i+1}(p^{m+1,\theta(m+1)}, \widehat{r}^{m+1,i}) &\leq A^{i+1}(p^{m+1,o}) + \sum_{k=o}^{k=\theta(m+1)} \frac{l^{m+1,k}}{\widehat{r}^{m+1,i}} \\ &\leq A^{i+1}(p^{m+1,o}) + \frac{l^{m+1}}{\widehat{r}^{m+1,i}} \end{aligned}$$

Since scheduling algorithm at server i belongs to GR for flow f, we get:

$$\begin{aligned} GRC^{i+1}(p^{m+1,\theta(m+1)}, \widehat{r}^{m+1,i}) &\leq \max\{A^i(p^{m+1}), GRC^i(p^m, \widehat{r}^{m,i})\} + \frac{l^{m+1}}{r^{m+1,i}} + \alpha^i + \frac{l^{m+1}}{\widehat{r}^{m+1,i}} \\ &\leq \left(\max\{A^i(p^{m+1}), GRC^i(p^m, \widehat{r}^{m,i})\} + \frac{l^{m+1}}{\widehat{r}^{m+1,i}}\right) + \frac{l^{m+1}}{r^{m+1,i}} + \alpha^i \\ &\leq GRC^{i}(p^{m+1}, \widehat{r}^{m+1,i}) + \frac{l^{m+1}}{r^{m+1,i}} + \alpha^i \\ &\leq GRC^{i}(p^{m+1}, \widehat{r}^{m+1,i}) + \max_{k \in [1..m+1]} \frac{l^k}{r^{k,i}} + \alpha^i \end{aligned} \tag{39}$$

From (36), (38), (39), and the induction hypothesis, we conclude that (23) holds for $1 \leq j \leq m+1$. Hence the theorem follows. ■

A.3 Proof of Theorem 8

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

$$GRC^{i+1}(p^1, \widehat{r}^{1,i}) = A^{i+1}(p^1) + \frac{l^1}{\widehat{r}^{1,i}}$$

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

$$\begin{aligned} GRC^{i+1}(p^1, \widehat{r}^{1,i}) &\leq GRC^{i}(p^{1,\theta(1)}, \widehat{r}^{1,i}) + \frac{l^1}{\widehat{r}^{1,i}} + \alpha^i \\ &\leq GRC^{i}(p^{1,\theta(1)}, \widehat{r}^{1,i}) + \max_{k \in [1..1]} \frac{l^k}{\widehat{r}^{1,i}} + \alpha^i \end{aligned}$$

Therefore (24) holds for $j = 1$
Induction Hypothesis: Assume (24) holds for $1 \leq j \leq m$.
Induction: We need to show (24) holds for $1 \leq j \leq m+1$.

$$GRC^{i+1}(p^{m+1}, \widehat{r}^{m+1,i}) = \max\{A^{i+1}(p^{m+1}), GRC^{i+1}(p^m, \widehat{r}^{m,i})\} + \frac{l^{m+1}}{\widehat{r}^{m+1,i}} \tag{40}$$

There are two cases to consider:

1. $A^{i+1}(p^{m+1}) > GRC^{i+1}(p^m, \widehat{r}^{m,i})$: From (40) we get:

$$GRC^{i+1}(p^{m+1}, \widehat{r}^{m+1,i}) \leq A^{i+1}(p^{m+1}) + \frac{l^{m+1}}{\widehat{r}^{m+1,i}}$$

Since scheduling algorithm at server i belongs to GR for flow f, we get:

$$\begin{aligned} GRC^{i+1}(p^{m+1}, \widehat{r}^{m+1,i}) &\leq GRC^{i}(p^{m+1,\theta(m+1)}, \widehat{r}^{m+1,i}) + \alpha^i + \frac{l^{m+1}}{\widehat{r}^{m+1,i}} \\ &\leq GRC^{i}(p^{m+1,\theta(m+1)}, \widehat{r}^{m+1,i}) + \max_{k \in [1..m+1]} \frac{l^k}{\widehat{r}^{k,i}} + \alpha^i \end{aligned} \tag{41}$$

2. $A^{i+1}(p^{m+1}) \leq GRC^{i+1}(p^m, \widehat{r}^{m,i})$: From (40) we get:

$$GRC^{i+1}(p^{m+1}, \widehat{r}^{m+1,i}) \leq GRC^{i+1}(p^m, \widehat{r}^{m,i}) + \frac{l^{m+1}}{\widehat{r}^{m+1,i}}$$

Using induction hypothesis we get,

$$\begin{aligned} GRC^{i+1}(p^{m+1}, \widehat{r}^{m+1,i}) &\leq GRC^{i}(p^{m,\theta(m)}, \widehat{r}^{m,i}) + \max_{k \in [1..m]} \frac{l^k}{\widehat{r}^{k,i}} + \frac{l^{m+1}}{\widehat{r}^{m+1,i}} + \alpha^i \\ &\leq \left(GRC^{i}(p^{m,\theta(m)}, \widehat{r}^{m,i}) + \frac{l^{m+1}}{\widehat{r}^{m+1,i}} \right) + \max_{k \in [1..m+1]} \frac{l^k}{\widehat{r}^{k,i}} + \alpha^i \\ &\leq GRC^{i}(p^{m+1,\theta(m+1)}, \widehat{r}^{m+1,i}) + \max_{k \in [1..m+1]} \frac{l^k}{\widehat{r}^{k,i}} + \alpha^i \end{aligned} \tag{42}$$

From (41), (42), and the induction hypothesis, we conclude that (24) holds for $1 \leq j \leq m+1$. Hence the theorem follows. ■