End-to-End Delay Bounds for Traffic Aggregates under Guaranteed-Rate Scheduling Algorithms

Wei Sun and Kang G. Shin

Real-Time Computing Laboratory
Department of Electrical Engineering and Computer Science
The University of Michigan
Ann Arbor, MI 48109-2122
Email: {wsunz, kgshin}@eecs.umich.edu

February 12, 2002. Revised on June 12, 2002 and May 19, 2003

***Abstract*— This paper evaluates, via both analysis and simulation, the end-to-end (e2e) delay performance of aggregate scheduling with guaranteed-rate (GR) algorithms. Deterministic e2e delay bounds for a *single* aggregation are derived under the assumption that all incoming flows at an aggregator conform to the token bucket model. Each aggregator can use any of three types of GR scheduling algorithms: stand-alone GR, two-level hierarchical GR, and rate-controlled two-level hierarchical GR. We also derive an e2e delay bound for the case of *multiple* aggregations when the rate-controlled two-level hierarchical GR is used at aggregators. By using the GR scheduling algorithms to handle traffic aggregates, we show not only the existence of delay bounds, but also the fact that under certain conditions (e.g., when the aggregate traverses a long path after the aggregation point) the bounds are even tighter than that of per-flow scheduling. We then compare the analytic delay bounds numerically, and conduct in-depth simulation to (i) confirm the analytic results, and (ii) compare the e2e delays of aggregate and per-flow scheduling. The simulation results have shown that aggregate scheduling is very robust and can take advantage of statistical multiplexing gains. It performs better than per-flow scheduling in most simulation scenarios we considered.**

Overall, aggregate scheduling is shown theoretically to provide bounded e2e delays, and practically to provide excellent e2e delay performance. Moreover, it incurs lower scheduling and state-maintenance overheads at routers than per-flow scheduling. All of these salient features make aggregate scheduling very attractive to Internet core networks.

***Index Terms*— Traffic aggregation, token bucket model, aggregate and per-flow scheduling, deterministic end-to-end delay bounds.**

I. Introduction

Real-time applications, such as voice-over-IP (VoIP) and video conferencing, require the network to provide better Quality-of-Service (QoS) in terms of delay, jitter, and loss rate. To provide such QoS support, several network architectures have been proposed. The IntServ architecture [1] supports QoS via *per-flow* resource reservation (e.g., RSVP) and packet scheduling. Numerous scheduling algorithms (e.g., see [2] for a survey) have been proposed to support IntServ QoS, such as fairness, bounded per-flow (per-node or e2e) delay and backlog under certain traffic models like the token bucket. Since both resource reservation and packet scheduling are per-flow-based, and hence, the routers in the network must keep a large number of flow states, IntServ is not scalable for use in the core of the Internet that carries millions of flows.

To solve the IntServ's scalability problem, the DiffServ (DS) architecture [3] has been proposed by classifying traffic into a number of predefined classes, such as Expedited Forwarding (EF), Assured Forwarding (AF), and Best-Effort (BE) at the edge of each DS domain. The traffic class is identified by the marking in the DS field of each packet. Flow information is visible only at edge routers of a DiffServ domain, and sophisticated packet classification, marking, policing, and shaping operations need only be implemented at edge routers. Within each DS domain, packets receive a particular per-hop forwarding behavior at routers on their path based on the DS field in their IP headers. In other words, flows are invisible and packet scheduling is done by the traffic class, *not* by individual flows. The EF class [4] receives priority over AF and BE classes. The DS architecture is more scalable than IntServ, but FIFO queueing—commonly used to schedule packets in each traffic class—is not suitable for hard QoS guarantees [5].

In this paper, we consider an alternative by extending the IntServ architecture to support traffic aggregation. This extension is made on the premise that there are *aggregation regions* in the network, which 'see' only aggregated (not individual) flows. Resource reservation and packet scheduling in an aggregation region are done on a per-aggregate basis. An example of this is the Virtual Paths (VPs) in ATM networks. Traffic aggregation has been studied extensively: see [6], [7], [8], [9] for resource reservation, [10], [11] for the admission control of aggregates, and [12] for flow state aggregation. This paper focuses on the scheduling issues associated with traffic aggregation. It evaluates the deterministic e2e delay bounds of aggregate scheduling when guaranteed-rate (GR) algorithms [13] are used.

Traffic aggregates discussed in this paper is similar to the *traffic trunk*—which is defined as an aggregate of traffic flows that belong to the same class—in Multiprotocol Label Switching (MPLS) [14], [15]. As shown in Fig. 1, some flows enter an aggregation region from ingress router S_1, and share the same path for a number of hops inside the region. Then

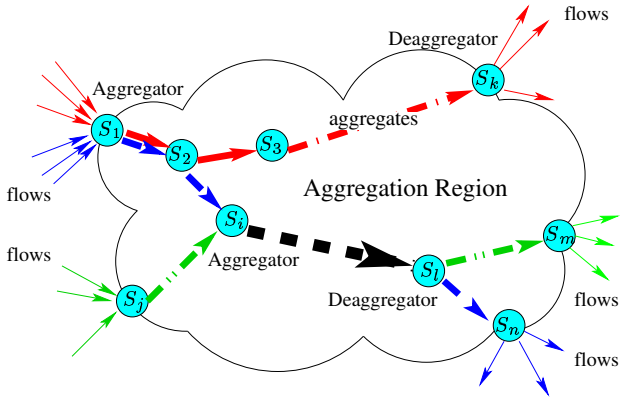


Fig. 1. Aggregate Scheduling

inside the region they are treated as a *single* aggregate at the routers on the path. S_1 aggregates the flows together by using GR scheduling algorithms. When the aggregate leaves S_n, it is split back into individual flows. Similarly, another aggregate can be set up between S_1 and S_k. In general, a traffic aggregate can be created and terminated at any point in the network, and it can be created recursively. For example, at S_i two aggregates are bundled again into another (higher level) aggregate, which is terminated later at S_l. The router (e.g., S_1) that aggregates flows (using GR algorithms) is called an *aggregator*; the router (e.g., S_k) that splits the aggregate is called a *deaggregator*. The flows before aggregation are called *flows*, and the traffic aggregates after aggregation are called *aggregate flows* or simply *aggregates*. We specify the aggregator and deaggregator as part of the aggregation region, although they can differentiate among the constituent flows of each aggregate.

The main contributions of the paper are twofold. First, by using the GR algorithms to schedule traffic aggregates, we show not only the existence of e2e delay bounds, but also the fact that the bounds could be tighter than those of per-flow scheduling. Second, using in-depth simulation we not only confirm the analytical results, but also show the advantages of aggregate scheduling.

Note that the delay bound problem of aggregate scheduling was also studied by Cobb [16], [17]. By using rate-based scheduling algorithms and *fair aggregators*, he showed that the e2e delay under aggregate is bounded, and the e2e delay bound can be even smaller than the per-flow e2e delay bound.

This paper extends the results in [16], [17] in the following *general* ways: First, the fair aggregators in [16], [17] (both the *basic fair aggregator* and *greedy fair aggregator*) use non-work-conserving scheduling algorithms. In contrast, our definition of fair aggregator does not have this requirement, and thus, is more general. Any work-conserving GR scheduling algorithms can be used at aggregators. Using the token-bucket traffic model, we derived the delay bounds under two types of work-conserving fair aggregators—stand-alone and hierarchical fair aggregators. Second, both the *basic fair aggregator* and *greedy fair aggregator* implicitly assume that there is only one outgoing traffic aggregate at an output interface, and thus, the derivation does not consider the interference from the packets outside the aggregate. In contrast, our delay bound results are more general: all of the three delay bounds are derived under the assumption that there could be multiple traffic aggregates through an output link. Third, we explicitly provide the delay bounds using the token-bucket traffic model, and establish the relationship between a flow's delay bound and burst sizes of itself and other flows sharing the same aggregate.

Organization of the Paper

The rest of the paper is organized as follows. Section II explains the definitions of GR scheduling [18] and $\mathcal{LR}$ server [19], discusses their relationship, and reviews the delay bound results for per-flow scheduling in [18]. Section III introduces the concept of fair aggregator, and derives the delay bound for aggregate scheduling under the token bucket traffic model. Two delay bounds are derived: the first for stand-alone aggregator and the second for hierarchical aggregator, both of which use work-conserving GR scheduling algorithms. Hierarchical aggregator is shown to improve the delay bound. Section IV improves the delay bound further by having the aggregators use non-work-conserving scheduling algorithms. Section V presents numerical results, comparing the above deterministic delay bounds with that of per-flow scheduling. Simulation is also used to compare the delay performance of both aggregate and per-flow scheduling, and the results confirm the benefits of aggregate scheduling derived from the analysis. Section VI discusses some related work on aggregate scheduling, putting our results in a comparative perspective. Finally, Section VII summarizes our contributions and discusses future directions.

II. Guaranteed-Rate Scheduling Algorithms

Before deriving the delay bound under aggregate scheduling, in this section we introduce the definitions of *Guaranteed-Rate (GR) scheduling algorithm* and *Latency-Rate ($\mathcal{LR}$) server*, discuss their relationship, and review the e2e delay bound results for per-flow scheduling.

A. Guaranteed-Rate (GR) Scheduling Algorithms

The authors of [18] defined a class of GR scheduling algorithms. The delay guarantees provided by these algorithms are based on the *Guaranteed Rate Clock* (GRC) value associated with each packet, which is defined as follows [18].

Definition 1 (GR Clock Value): Consider a flow f associated with a guaranteed rate r_f. Let p_f^j and ℓ_f^j denote the j^{th} packet of flow f and its length, respectively. Also, let $GRC^i(p_f^j)$ and $A^i(p_f^j)$ denote the GRC value and arrival time of packet p_f^j at router S_i, respectively. Then, the GRC values for packets of flow f are given by:

$$GRC^i(p_f^j) = \begin{cases} 0, & j = 0 \\ \max\{A^i(p_f^j), GRC^i(p_f^{j-1})\} + \frac{\ell_f^j}{r_f}, & j \geq 1. \end{cases} \quad (1)$$

Definition 2 (GR Scheduling Algorithm): A scheduling algorithm at router S_i is said to belong to the GR class for flow f if it guarantees that packet p_f^j is transmitted by time $GRC^i(p_f^j) + \beta^i$, where β^i is a *scheduling constant* [16] that depends on the scheduling algorithm and the router.

We will henceforth call a router equipped with the GR scheduling algorithm a *GR server*. Many scheduling algorithms are shown in [18] to belong to the GR class. For example, both Packet-level Generalized Processor Sharing (PGPS) [20] and Virtual Clock (VC) [21] are GR scheduling algorithms with $\beta^i = \frac{L^i_{max}}{C^i}$, where L^i_{max} is the maximum packet length seen by router S_i and C^i the output link capacity of S_i.

B. Latency-Rate ($\mathcal{LR}$) Server

In a related work, the authors of [19] defined a class of scheduling algorithms as *Latency-Rate ($\mathcal{LR}$) servers*, the definition of which is repeated below. To keep the paper consistent, some notations in the definition are changed.

Definition 3 ($\mathcal{LR}$ Server): Let τ be the starting time of a busy period of flow f in server S_i and τ^* the time at which the last bit of traffic arrived during the busy period leaves the server. Then, server S_i is an $\mathcal{LR}$ server if and only if a nonnegative constant C^i_f can be found such that, at every instant t in the interval $(\tau, \tau^*]$,

$$W^i_f(\tau, t) \geq \max\{0, r_f(t - \tau - C^i_f)\}, \tag{2}$$

where $W^i_f(\tau, t)$ denotes the total service provided by the server S_i to flow f that arrived during the busy period until time t. The minimum nonnegative constant C^i_f satisfying the above inequality is defined as the *latency* of the server, denoted by θ^i_f.

The definition of the $\mathcal{LR}$ server helps set up the relationship between the burst size of an output flow and that of the corresponding input flow at a router.

Lemma 1: Suppose an incoming flow f to router S_i conforms to the token bucket model (σ_f, ρ_f). If S_i is an $\mathcal{LR}$ server with parameters (θ^i_f, r_f) for flow f, where θ^i_f is the latency at S_i and r_f ($r_f \geq \rho_f$) the guaranteed rate for flow f, respectively, then the output traffic of flow f conforms to the token bucket model with parameters$(\tilde{\sigma}_f, \rho_f)$, where $\tilde{\sigma}_f \leq \sigma_f + \theta^i_f \cdot \rho_f$.

For example, both PGPS and VC are $\mathcal{LR}$ servers with $\theta^i_f = \frac{\ell^{max}_f}{r_f} + \frac{L^i_{max}}{C^i}$ [19], in which case we have

$$\tilde{\sigma}_f \leq \sigma_f + \ell^{max}_f + \frac{\rho_f \cdot L^i_{max}}{C^i}. \tag{3}$$

The proof of Lemma 1 is similar to that of Theorem 3 in [19], which assumes $\rho_f = r_f$; but the result can be easily extended to the case of $\rho_f \leq r_f$.

C. Relationship between GR Server and $\mathcal{LR}$ Server

From the definition of the $\mathcal{LR}$ server, it is easy to show that an $\mathcal{LR}$ server is also a GR server.

Theorem 1: For any flow f, an $\mathcal{LR}$ server S_i with latency θ^i_f is also a GR server with scheduling constant θ^i_f.

Proof: Without loss of generality, we consider only one busy period of flow f. Suppose the reserved rate for flow f is r_f, and the busy period starts at time τ and ends at τ^*. From the definition of GRC value in Eq. (1), it is easy to see that the GRC values of the packets in this busy period are:

$$GRC(p^k_f) = \frac{\sum_{n=1}^{k} \ell^n_f}{r_f} + \tau, \quad k \geq 1.$$

Suppose t_k is the time when the k^{th} packet leaves the server, then

$$W^i_f(\tau, t_k) = \sum_{n=1}^{k} \ell^n_f.$$

From the definition of $\mathcal{LR}$ server, we obtain

$$\sum_{n=1}^{k} \ell^n_f \geq \max\{0, r_f(t_k - \tau - \theta^i_f)\} \geq r_f(t_k - \tau - \theta^i_f).$$

Rearranging the terms, we obtain

$$t_k \leq \frac{\sum_{n=1}^{k} \ell^n_f}{r_f} + \tau + \theta^i_f = GRC(p^k_f) + \theta^i_f, \quad \forall k \geq 1. \tag{4}$$

The theorem is proved. ■

Similarly, a GR server is also an $\mathcal{LR}$ server.

Theorem 2: For any flow f, an GR server with scheduling constant β_f is also a $\mathcal{LR}$ server with latency smaller than or equal to $\beta_f + \frac{\ell^{max}_f}{r_f}$.

Proof: Similarly, without loss of generality, we consider only one busy period of flow f. Suppose the reserved rate for flow f is r_f, and the busy period starts at time τ and ends at τ^*. From the definition of GRC value, it is easy to see that the GRC values of the packets in this busy period are:

$$GRC(p^k_f) = \frac{\sum_{n=1}^{k} \ell^n_f}{r_f} + \tau, \quad k \geq 1.$$

First, consider the time instants when packets leave the server. Suppose the k^{th} packet leaves the server at time t_k, $k \geq 1$, then from the definition of GR server, we obtain

$$\begin{aligned} & t_k \leq GRC(p^k_f) + \beta_f, \quad k \geq 1, \\ \Longrightarrow\ & t_k \leq \frac{\sum_{n=1}^{k} \ell^n_f}{r_f} + \tau + \beta_f, \quad k \geq 1, \\ \Longrightarrow\ & W^i_f(\tau, t_k) = \sum_{n=1}^{k} \ell^n_f \geq r_f(t_k - \tau - \beta_f), \quad k \geq 1. \end{aligned} \tag{5}$$

Next, consider time instant t other than t_k (i.e., $t_k < t < t_{k+1}$ for all $k \geq 0$, assuming $t_0 = \tau$). Since $W^i_f(\tau, t)$ only increases when the last bit of a packet leaves the server, for any $k \geq 0$,

$$\begin{aligned} W^i_f(\tau, t) &= W^i_f(\tau, t_k) \\ &= W^i_f(\tau, t_{k+1}) - \ell^{k+1}_f \\ &\geq r_f(t_{k+1} - \tau - \beta_f - \frac{\ell^{k+1}_f}{r_f}) \\ &\geq r_f(t - \tau - (\beta_f + \frac{\ell^{max}_f}{r_f})), \quad t_k < t < t_{k+1}. \end{aligned} \tag{6}$$

Combining Eqs. (5) and (6), we obtain

$$W^i_f(\tau, t) \geq r_f(t - \tau - (\beta_f + \frac{\ell^{max}_f}{r_f})), \qquad \forall \tau \leq t \leq \tau^*.$$

Since $W^i_f(\tau, t) \geq 0$ for all $\tau \leq t \leq \tau^*$, we have

$$W^i_f(\tau, t) \geq \max\{0, r_f(t - \tau - (\beta_f + \frac{\ell^{max}_f}{r_f}))\}, \quad \forall \tau \leq t \leq \tau^*. \tag{7}$$

Then the theorem is proved. ■

D. End-to-End Delay Bound under Per-Flow Scheduling

Now we review the e2e delay bound results for per-flow scheduling. Both Lemma 2 and Theorem 3 stated below are proved in [18].

Lemma 2: Suppose routers S_i and S_{i+1} are two neighboring GR servers on the path of flow f. If both routers guarantee service rate r_f for flow f, then

$$GRC^{i+1}(p_f^j) \leq GRC^i(p_f^j) + \frac{\ell_f^{max}}{r_f} + \alpha^i, \tag{8}$$

where ℓ_f^{max} is the maximum packet size in flow f, and $\alpha^i = \beta^i + \tau^{i,i+1}$. Here $\tau^{i,i+1}$ is the propagation delay between S_i and S_{i+1}.

Lemma 2 states the relationship between the GRC values of a packet at two neighboring GR servers. Based on this relationship, the authors of [18] derived an e2e delay bound. Before introducing that delay bound, we define the *token bucket traffic model* as follows: flow f conforms to the token bucket (σ_f, ρ_f) if for any time instant τ and t satisfying $0 \leq \tau < t$, its traffic volume arrived in period $(\tau, t]$, $A_f(\tau, t)$, satisfies:

$$A_f(\tau, t) \leq \sigma_f + \rho_f \cdot (t - \tau), \tag{9}$$

where σ_f and ρ_f are the burst size and average rate of flow f, respectively.

Theorem 3: If flow f conforms to the token bucket model (σ_f, ρ_f) and all the routers on its path are GR servers with guaranteed rate $r_f \geq \rho_f$, then the e2e delay for p_f^j, d_f^j, is bounded as follows:

$$d_f^j \leq \frac{\sigma_f}{r_f} + \frac{(K-1)\ell_f^{max}}{r_f} + \sum_{n=1}^{K} \alpha^n, \tag{10}$$

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

From Theorem 3, one can see that the delay bound is inversely proportional to the flow's guaranteed rate, but is proportional to the number of hops (K), the size of packets, and the burst size of the flow. With a large number of hops and large-size packets, the delay can be substantially large. To understand the physical meaning of Eq. (10), let us consider the fluid traffic model, where packet size is infinitely small. Then, if we omit the propagation delay, the delay bound in Eq. (10) can be simplified as:

$$d_f \leq \frac{\sigma_f}{r_f}. \tag{11}$$

In other words, the delay is upper bounded by the burstiness of the flow. The larger the burst size, the larger the delay bound. If we assume $\rho_f = r_f$, then the delay bound is decided by the *burst ratio* ($\frac{\sigma_f}{\rho_f}$ for flow f) of the flow.

In the next two sections, we derive e2e delay bounds under aggregate scheduling with GR scheduling algorithms. We show that if the incoming flows at aggregators conform to the token bucket model, the e2e delay is bounded.

III. Work-Conserving Aggregator

To derive the e2e delay bound under aggregate scheduling, an important step is to derive the delay at the aggregator. We need to find the relationship between the GRC values at the aggregator and its next hop. In this section, we use work-conserving GR scheduling algorithms at aggregators. Two types of GR scheduling algorithms are examined: *stand-alone* (or non-hierarchical) algorithms and *hierarchical* algorithms.

TABLE I
SYMBOLS

S_i	the i^{th} router/server along the path of a flow or aggregate
p_f^j	the j^{th} packet of flow f
p_A^j	the j^{th} packet of aggregate flow A
ℓ_f^j	packet length of p_f^j
ℓ_A^j	packet length of p_A^j
ℓ_f^{max}	max. packet length in flow f
ℓ_A^{max}	max. packet length in aggregate flow A
L_{max}^i	max. packet length at router S_i
C^i	link capacity between S_i and S_{i+1}
σ_f	burst size of flow f under token bucket model
ρ_f	average rate of flow f under token bucket model
r_f	guaranteed rate for flow f ($r_f \geq \rho_f$)
R	sum of the guaranteed rates of all the flows in the aggregate flow, i.e., $R = \sum_{k=1}^{N} r_k$
$A_f(\tau, t)$	traffic arrived from flow f during $(\tau, t]$
$A^i(p_f^j)$	arrival time of packet p_f^j at router S_i
$GRC^i(p_f^j)$	guaranteed rate clock for p_f^j at S_i
$D^i(p_f^j)$	departure time of p_f^j at router S_i
α^i	$\alpha^i = \beta^i + \tau^{i,i+1}$
β^i	scheduling constant at router S_i
$\tau^{i,i+1}$	propagation delay between S_i and S_{i+1}
γ^i	aggregation constant at router S_i
d_f^j	e2e delay bound for p_f^j

For the convenience of discussion, we first give a list of symbols in Table I.

Note that by using GR scheduling algorithms, we always assume that the guaranteed rate for a flow is greater than, or equal to, its average rate, i.e., $r_f \geq \rho_f$. Also, in the remainder of this paper, we omit the propagation delay $\tau^{i,i+1}$. Therefore, $\alpha^i = \beta^i$.

A. Fair Aggregator

Before deriving the delay at the aggregator, we first introduce an important concept—*fair aggregator*.

Definition 4 (Fair Aggregator): Let router S_i be an aggregator bundling flow f and others into aggregate flow A, which, in turn, is an input to router S_{i+1}. Then if $p_f^j = p_A^{j'}$, i.e., the j^{th} packet of flow f corresponds to the j'^{th} packet of aggregate flow A, S_i is said to be a *fair aggregator* if and only if S_i is a GR server and

$$GRC^{i+1}(p_A^{j'}) \leq GRC^i(p_f^j) + \gamma^i + \alpha^i, \qquad \forall j \geq 1, \tag{12}$$

where γ^i is a constant that depends on the scheduling algorithm, the router, and other flows in the same aggregate. We call it an *aggregation constant*.

The value $\gamma^i + \alpha^i$ can be considered as the *fairness index* of the aggregator, e.g., an aggregator is considered fairer than another if it has a smaller value of $\gamma^i + \alpha^i$. For an fair aggregator, both of its aggregation constant (γ^i) and scheduling constant (α^i) should be small. Also, our definition of *fair aggregator* is slightly different from that in [16], [17]. It is based on the relationship between a packet's GRC values at the aggregator and its next hop.

Lemma 3: Suppose S_i and S_{i+1} are two neighboring GR servers and S_i is an aggregator. S_i aggregates flow f and other

(N-1) flows into aggregate flow A, which, in turn, is an input to S_{i+1}. Suppose incoming flow k at S_i conforms to the token bucket model (σ_k, ρ_k) $(1 \le k \le N)$, and the guaranteed rate for flow k at S_i is r_k $(1 \le k \le N)$. Let $p_f^j = p_A^{j'}$, i.e., the j^{th} packet of flow f corresponds to the j'^{th} packet of aggregate flow A. If the outgoing flows at S_i conform to the token bucket model $(\tilde{\sigma}_k, \rho_k)$ $(1 \le k \le N)$, then for packet p_f^j,

$$GRC^{i+1}(p_A^{j'}) \le GRC^i(p_f^j) + \frac{\sum_{k\neq f}\tilde{\sigma}_k + \ell_f^{max}}{R} + \alpha^i, \quad j \ge 1, \quad (13)$$

where $R = \sum_{k=1}^{N} r_k$, which is the guaranteed rate for aggregate flow A at S_{i+1}.

Proof: Similar to the definition of busy period of a single flow [19], we define a *busy period of aggregate flow* A at router S_{i+1} as the maximum length of time period $(\tau_1, \tau_2]$, such that at any time $t \in (\tau_1, \tau_2]$, the total traffic of A arrived since the beginning of the interval is $A(\tau_1, t) \ge R \cdot (t - \tau_1)$. Without loss of generality, we consider only one busy period in the proof. Suppose the busy period starts at time t_0 when the first packet p_A^1 arrives, and in that busy period, time t_j is the instant when packet p_f^j arrives at S_{i+1}. Since $p_f^j = p_A^{j'}$, the total traffic arrival of aggregate A up to time t_j is

$$\begin{aligned}
\sum_{k=1}^{j'} \ell_A^k &= A(t_0, t_j) = \sum_{k=1}^{N} A_k(t_0, t_j) \\
&= A_f(t_0, t_j) + \sum_{k\neq f} A_k(t_0, t_j) \\
&\le \sum_{m=1}^{j} \ell_f^m + \sum_{k\neq f}(\tilde{\sigma}_k + r_k \cdot (t_j - t_0)) \\
&= \sum_{m=1}^{j} \ell_f^m + \sum_{k\neq f}\tilde{\sigma}_k + (R - r_f)(t_j - t_0). \qquad (14)
\end{aligned}$$

Since all the packets arrive in the same busy period of aggregate A, from the definition of GRC value, we have

$$\begin{aligned}
GRC^{i+1}(p_A^{j'}) &= \frac{\sum_{k=1}^{j'} \ell_A^k}{R} + t_0 \\
&\le \frac{\sum_{m=1}^{j} \ell_f^m + \sum_{k\neq f}\tilde{\sigma}_k + (R - r_f)(t_j - t_0)}{R} + t_0. \qquad (15)
\end{aligned}$$

Next, we prove that the theorem holds for both $j = 1$ and $j > 1$.

Case 1: $j = 1$.

$$\begin{aligned}
GRC^{i+1}(p_A^{j'}) &\le \frac{\ell_f^1 + \sum_{k\neq f}\tilde{\sigma}_k + (R - r_f)(t_1 - t_0)}{R} + t_0 \\
&= \frac{\sum_{k\neq f}\tilde{\sigma}_k + \ell_f^1}{R} + t_1 - \frac{r_f \cdot (t_1 - t_0)}{R} \\
&\le \frac{\sum_{k\neq f}\tilde{\sigma}_k + \ell_f^{max}}{R} + t_1.
\end{aligned}$$

Since S_i is a GR server for flow f, we have $t_1 \le GRC^i(p_f^1) + \alpha^i$. Therefore,

$$GRC^{i+1}(p_A^{j'}) \le GRC^i(p_f^1) + \frac{\sum_{k\neq f}\tilde{\sigma}_k + \ell_f^{max}}{R} + \alpha^i.$$

Case 2: $j > 1$. From the definition of GRC value, for flow f, we have

$$\begin{aligned}
GRC^i(p_f^2) &\ge GRC^i(p_f^1) + \frac{\ell_f^2}{r_f}, \\
GRC^i(p_f^3) &\ge GRC^i(p_f^2) + \frac{\ell_f^3}{r_f}, \\
&\vdots \\
GRC^i(p_f^j) &\ge GRC^i(p_f^{j-1}) + \frac{\ell_f^j}{r_f}.
\end{aligned}$$

Adding them up, we obtain

$$\begin{aligned}
&GRC^i(p_f^j) - GRC^i(p_f^1) \ge \frac{\sum_{m=2}^{j} \ell_f^m}{r_f} \\
\Longrightarrow &\sum_{m=1}^{j} \ell_f^m \le \ell_f^1 + r_f \cdot (GRC^i(p_f^j) - GRC^i(p_f^1)). \qquad (16)
\end{aligned}$$

Then, from Eqs. (15) and (16), and the fact $t_j \le GRC^i(p_f^j) + \alpha^i$, we obtain

$$\begin{aligned}
GRC^{i+1}(p_A^{j'}) &\le \frac{\ell_f^1 + r_f \cdot (GRC^i(p_f^j) - GRC^i(p_f^1))}{R} \\
&\quad + \frac{\sum_{k\neq f}\tilde{\sigma}_k + (R - r_f)(t_j - t_0)}{R} + t_0 \\
&\le \frac{\sum_{k\neq f}\tilde{\sigma}_k + \ell_f^1}{R} + \frac{r_f(GRC^i(p_f^j) - GRC^i(p_f^1))}{R} \\
&\quad + \frac{(R - r_f)(GRC^i(p_f^j) + \alpha^i - t_0) + R \cdot t_0}{R} \\
&= GRC^i(p_f^j) + \frac{\sum_{k\neq f}\tilde{\sigma}_k + \ell_f^1}{R} + \alpha^i \\
&\quad - \frac{r_f(GRC^i(p_f^1) + \alpha^i - t_0)}{R} \\
&\le GRC^i(p_f^j) + \frac{\sum_{k\neq f}\tilde{\sigma}_k + \ell_f^1}{R} + \alpha^i - \frac{r_f(t_1 - t_0)}{R} \\
&\le GRC^i(p_f^j) + \frac{\sum_{k\neq f}\tilde{\sigma}_k + \ell_f^{max}}{R} + \alpha^i.
\end{aligned}$$

The lemma is proved. ■

Next, we study the stand-alone aggregator first, which uses stand-alone GR scheduling algorithm to bundle flows into aggregates.

B. Delay Bound I: Stand-Alone Aggregator

Combining Lemmas 1 and 3, we obtain the following theorem on the relationship between the GRC values at a stand-alone aggregator and its next hop.

Theorem 4: Suppose S_i is an LR server, and both S_i and S_{i+1} are GR servers. As a stand-alone aggregator, S_i aggregates flow f and other (N-1) flows into aggregate A, which, in turn, is an input to router S_{i+1}. Suppose incoming flow k at S_i conforms to the token bucket model (σ_k, ρ_k) $(1 \le k \le N)$, and the guaranteed rate for flow k at router S_i is r_k $(1 \le k \le N)$. Let $p_f^j = p_A^{j'}$, i.e., the j^{th} packet of flow f corresponds to the j'^{th} packet of aggregate A. Then, S_i is a fair aggregator with $\gamma^i = \frac{\sum_{k\neq f}\sigma_k}{R} + \frac{\sum_{k\neq f}\theta_k^i \cdot \rho_k + \ell_f^{max}}{R}$. In other words,

$$\begin{aligned}
GRC^{i+1}(p_A^{j'}) &\le GRC^i(p_f^j) + \alpha^i + [\frac{\sum_{k\neq f}\sigma_k}{R} \\
&\quad + \frac{\sum_{k\neq f}\theta_k^i \cdot \rho_k + \ell_f^{max}}{R}], \quad j \ge 1. \qquad (17)
\end{aligned}$$

The proof is trivial.

Now, we are ready to derive the e2e delay bound for aggregate scheduling under the token bucket model and stand-alone aggregator.

Theorem 5: Suppose N flows share the same K hops of GR servers inside an aggregation region, and they are bundled into aggregate A at stand-alone aggregator S_1 and split back at S_K. Routers $S_2, \ldots, S_{K-1}$ schedule packets of aggregate A. If flow k conforms to the token bucket model (σ_k, ρ_k) and has the guaranteed rate r_k with $r_k \geq \rho_k$ $(1 \leq k \leq N)$ at S_1 and S_K, and A has the guaranteed rate $R = \sum_{k=1}^{N} r_k$ at $S_2, \ldots, S_{K-1}$, then for any flow f $(1 \leq f \leq N)$, the e2e delay of packet p_f^j, d_f^j, is bounded as follows:

$$d_f^j \leq \frac{\sigma_f}{r_f} + [\frac{\sum_{k \neq f} \sigma_k}{R} + \frac{\sum_{k \neq f} \theta_k^1 \cdot r_k + \ell_f^{max}}{R}] + (K-3)\frac{\ell_A^{max}}{R} + \frac{\ell_f^{max}}{r_f} + \sum_{i=1}^{K} \alpha^i, \quad j \geq 1. \tag{18}$$

Proof: Let $p_A^{j'}$ be the packet in the aggregate flow A corresponding to p_f^j in flow f. From Theorem 4, we have

$$GRC^2(p_A^{j'}) \leq GRC^1(p_f^j) + \gamma^1 + \alpha^1,$$

where $\gamma^1 = \frac{\sum_{k \neq f} \sigma_k}{R} + \frac{\sum_{k \neq f} \theta_k^1 \cdot \rho_k + \ell_f^{max}}{R}$. Since S_2, ..., S_{K-1} are all GR servers for aggregate flow A with guaranteed rate $R = \sum_{k=1}^{N} r_k$, from Lemma 2 we have

$$\begin{aligned} GRC^3(p_A^{j'}) &\leq GRC^2(p_A^{j'}) + \frac{\ell_A^{max}}{R} + \alpha^2, \\ GRC^4(p_A^{j'}) &\leq GRC^3(p_A^{j'}) + \frac{\ell_A^{max}}{R} + \alpha^3, \\ &\vdots \\ GRC^{K-1}(p_A^{j'}) &\leq GRC^{K-2}(p_A^{j'}) + \frac{\ell_A^{max}}{R} + \alpha^{K-2} \end{aligned}$$

Adding them all up, we obtain

$$GRC^{K-1}(p_A^{j'}) \leq GRC^1(p_f^j) + (K-3)\frac{\ell_A^{max}}{R} + \gamma^1 + \sum_{i=1}^{K-2} \alpha^i.$$

Since $p_f^j = p_A^{j'}$, for packet p_f^j, the departure time at S_{K-1} is

$$\begin{aligned} D^{K-1}(p_f^j) &= D^{K-1}(p_A^{j'}) \\ &\leq GRC^{K-1}(p_A^{j'}) + \alpha^{K-1} \\ &\leq GRC^1(p_f^j) + \gamma^1 + (K-3)\frac{\ell_A^{max}}{R} + \sum_{i=1}^{K-1} \alpha^i. \end{aligned}$$

From the definition of the GR server, the first (K-1) servers for flow f can be viewed as a virtual GR server with scheduling constant $\alpha^* = \gamma^1 + (K-3)\frac{\ell_A^{max}}{R} + \sum_{i=1}^{K-1} \alpha^i$. Since S_K is also a GR server for flow f with guaranteed rate r_f, from Lemma 2, we have

$$GRC^K(p_f^j) \leq GRC^1(p_f^j) + \frac{\ell_f^{max}}{r_f} + \alpha^*.$$

Therefore, the departure time of packet p_f^j from S_K is

$$\begin{aligned} D^K(p_f^j) &\leq GRC^K(p_f^j) + \alpha^K \\ &\leq GRC^1(p_f^j) + \frac{\ell_f^{max}}{r_f} + \alpha^* + \alpha^K \\ &= GRC^1(p_f^j) + [\frac{\sum_{k \neq f} \sigma_k}{R} + \frac{\sum_{k \neq f} \theta_k^1 \cdot r_k + \ell_f^{max}}{R}] \\ &\quad + (K-3)\frac{\ell_A^{max}}{R} + \frac{\ell_f^{max}}{r_f} + \sum_{i=1}^{K} \alpha^i. \end{aligned}$$

Then, the e2e delay d_f^j satisfies

$$\begin{aligned} d_f^j &= D^K(p_f^j) - A^1(p_f^j) \\ &\leq [GRC^1(p_f^j) - A^1(p_f^j)] + (K-3)\frac{\ell_A^{max}}{R} + \frac{\ell_f^{max}}{r_f} \\ &\quad + [\frac{\sum_{k \neq f} \sigma_k}{R} + \frac{\sum_{k \neq f} \theta_k^1 \cdot r_k + \ell_f^{max}}{R}] + \sum_{i=1}^{K} \alpha^i. \end{aligned} \tag{19}$$

From [18], for flow f conforming to the token bucket model (σ_f, ρ_f) and with reserved rate $r_f (r_f \geq \rho_f)$, $GRC^1(p_f^j) - A^1(p_f^j) \leq \frac{\sigma_f}{r_f}$. Thus the theorem is proven. ■

As in Section II, to further understand the physical meaning of Eq. (18), we consider the fluid traffic model, where packet size is infinitely small. Then, Eq. (18) can be simplified as (since in general the term θ_k^1 also relies on packet size and will become infinitely small):

$$d_f \leq \frac{\sigma_f}{r_f} + \frac{\sum_{k \neq f} \sigma_k}{R}. \tag{20}$$

Note that Eq. (20) provides the lower limit of the e2e delay bound. Comparing Eqs. (11) and (20), we can see that with aggregate scheduling, the delay bound of a flow is not only decided by the burstiness of the flow itself (term $\frac{\sigma_f}{r_f}$), but also strongly related to the burstiness of other flows that share the same aggregate with it (term $\frac{\sum_{k \neq f} \sigma_k}{R}$). Intuitively, this is easy to understand, since a packet from flow f has to wait behind not only the earlier packets from the same flow, but also those from other flows sharing the same aggregate. The result implies how aggregation should be done—a flow should not be aggregated with other flows with substantially larger burst ratios. This is similar to the conclusion in [22]. This issue will be further explored in Section V.

From Eq. (18), we can see that the delay bound is affected by the latency of the scheduling algorithm at the aggregator. To get a tighter bound, we should use a low-latency scheduling algorithm. For example, with PGPS and VC ($\theta_f^1 = \frac{\ell_f^{max}}{r_f} + \frac{L_{max}^1}{C^1}$) at the aggregator S_1, the e2e delay is bounded as follows:

$$d_f^j \leq \frac{\sigma_f}{r_f} + [\frac{\sum_{k \neq f} \sigma_k}{R} + \frac{\sum_{k=1}^{N} \ell_k^{max}}{R}] + (K-3)\frac{\ell_A^{max}}{R} + \frac{\ell_f^{max}}{r_f} + \frac{L_{max}^1}{C^1} + \sum_{i=1}^{K} \alpha^i. \tag{21}$$

Finally, comparing Eqs. (10) and (18), we note that, depending on the burst ratios of the constituent flows and the maximum packet size in the aggregate, the delay bound under aggregate scheduling can be tighter than that under per-flow

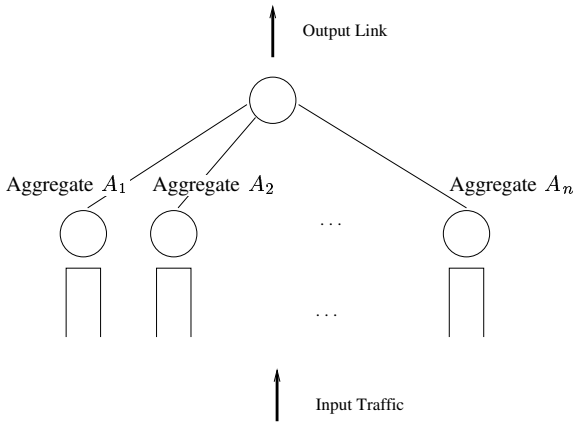


Fig. 2. Hierarchical Scheduling

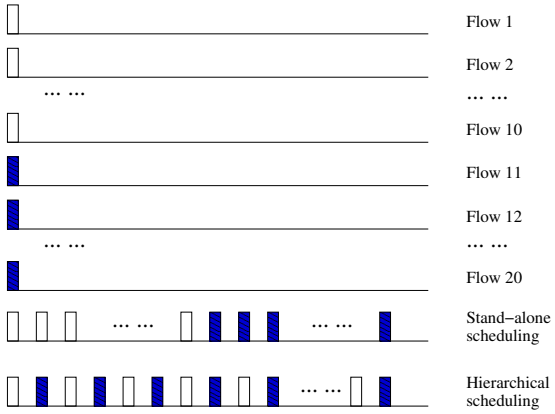


Fig. 3. Benefits of Hierarchical Scheduling

scheduling. We will compare the delay bounds numerically later in Section V.

In the next subsection, we further decrease the delay bound by using hierarchical GR scheduling algorithms at aggregators.

C. Delay Bound II: Hierarchical Aggregator

In the previous subsection, stand-alone GR scheduling algorithms were used at aggregators. Thus when computing the burstiness of an outgoing aggregate at aggregators, the burst size of each constituent flow in the aggregate was computed separately and then summed up. In this subsection, we use hierarchical GR algorithms at aggregators to improve the delay bound, as shown in Fig. 2. First, the flows that end up in the same aggregate are grouped together at the lower-level schedulers. Then, the aggregates are scheduled at the upper-level scheduler. By doing so, we can reduce the burstiness of the outgoing aggregate and the aggregation constant at aggregators, thus reducing the e2e delay.

How does hierarchical scheduler reduce the burstiness of the outgoing aggregates? Fig. 3 shows the intuition behind it. Suppose there are 20 flows coming into one aggregator, and all of them have the same reserved rate. The first 10 flows belong to one aggregate flow (e.g., A_1), and the last 10 belong to another aggregate flow (e.g., A_2). Suppose 20 packets with identical size arrive at the idle aggregator at exactly the same time, one from each flow. Then, with a stand-alone scheduler, the 20 packets will be scheduled in a random order. Thus, it is possible that all the 10 packets of A_1 are scheduled before all the packets of aggregate flow A_2. Therefore, the output of the aggregator will look like: 10 packets of aggregate A_1 comes out first, then 10 packets of aggregate A_2. For the next router which is handling aggregates, the burstiness of both A_1 and A_2 is very high. In contrast, with hierarchical scheduler, the upper-level scheduler will make sure the packets from A_1 and A_2 are scheduled alternately, making the traffic in both aggregate flows smoother.

The hierarchical scheduling algorithms under consideration are two-level *hierarchical packet fair queueing* (H-PFQ) algorithms [23]. As defined in [23], flow f arriving at a H-PFQ of height H has H ancestors in the tree, with $p(f)$ as its immediate parent node, and $p^h(f)$ is the parent node of $p^{h-1}(f)$ $(h = 1, \ldots, H)$. Here $p^0(f) = f$, $p^1(f) = p(f), \ldots$, and $p^H(f)$ = link capacity C. Since we only need two-level hierarchy, $H = 2$.

Before deriving the delay bounds, we first repeat the definition of *Bit Worst-case Fair Index* (B-WFI) below. Note that some notations in the definition are changed to keep the paper consistent.

Definition 5 (Bit Worst-case Fair Index (B-WFI)): A server node S_i is said to guarantee a Bit Worst-case Fair Index (B-WFI) of $\eta_{i,f}$ for flow f, if for any packet p_f^j the following holds

$$W_f^i(t, d_f^j) \geq \frac{\phi_f}{\Phi} W^i(t, d_f^j) - \eta_{i,f} \tag{22}$$

where d_f^j is the time p_f^j departs the server, t is any time instant such that $t < d_f^j$ and session f is continuously backlogged during $[t, d_f^j]$, and $\frac{\phi_f}{\Phi}$ is the service share guaranteed to flow f by server S_i.

Next, we show that H-PFQ also belongs to the GR class.

Fact 1: If a stand-alone packet fair queueing (PFQ) algorithm belongs to the GR class with the scheduling constant β_{PFQ}^i, then its corresponding H-PFQ of height H also belongs to the GR class with scheduling constant:

$$\beta_{HPFQ}^i \leq \beta_{PFQ}^i + \sum_{h=1}^{H-1} \frac{\eta_{p^h(f),f}}{r_{p^h(f)}}, \tag{23}$$

where $\eta_{p^h(f),f}$ and $r_{p^h(f)}$ are the *Bit Worst-case Fair Index* (B-WFI) [23] of the logical queue at node $p^h(f)$ and the guaranteed rate at node $p^h(f)$, respectively.

Proof: From Theorem 2 of [23], the delay bound for a flow f at H-PFQ is at most $\sum_{h=1}^{H-1} \frac{\eta_{p^h(f),f}}{r_{p^h(f)}}$ larger than that at the corresponding stand-alone PFQ server. Therefore, $\beta_{HPFQ}^i \leq \beta_{PFQ}^i + \sum_{h=1}^{H-1} \frac{\eta_{p^h(f),f}}{r_{p^h(f)}}$. ■

Now, we derive the burstiness of the outgoing traffic at aggregators. For any flow f, we derive $\sum_{k \neq f} \tilde{\sigma}_k$—the total outgoing burst size of all the other flows sharing the same aggregate with f. In the derivation, we make use of the corresponding fluid GPS (Generalized Processor Sharing) server of a PFQ (Packet Fair Queueing) server Let $W_k^i(\tau, t)$ $(W_{k,GPS}^i(\tau, t))$ be the service received by flow k at the PFQ server S_i (the corresponding GPS fluid server) during $(\tau, t]$,

and define λ_k^i and δ_k^i as follows

$$\lambda_k^i = \max_{t\geq 0}\{W_{k,GPS}^i(0,t) - W_k^i(0,t)\}, \quad (24)$$

$$\delta_k^i = \max_{t\geq 0}\{W_k^i(0,t) - W_{k,GPS}^i(0,t)\}. \quad (25)$$

Intuitively, λ_k^i and δ_k^i define the maximum difference between the amount of services flow k gets from the corresponding GPS server and the PFQ server S_i in any time period: λ_k^i (δ_k^i) defines how much more (less) service flow k can get from the corresponding GPS server than the PFQ server.

Theorem 6: Suppose S_i is an aggregator with a two-level H-PFQ scheduler and S_i aggregates flow f and other (N-1) flows into aggregate A. Also, suppose flow k at S_i conforms to the token bucket model (σ_k, ρ_k) $(1 \leq k \leq N)$, and the guaranteed rate for flow k at S_i is r_k $(1 \leq k \leq N)$. Then, we have

$$\sum_{k\neq f}\tilde{\sigma}_k \leq \sum_{k\neq f}\sigma_k + [\lambda_f^i + \delta_f^i] + (1 - \frac{r_f}{R})[\lambda_A^i + \delta_A^i]. \quad (26)$$

The proof is similar to the proof of Theorem 3 in [19]. We first prove the total backlog size of the (N-1) flows is upper bounded, then derive the burst size from the backlog size. The details of the proof can be found in Appendix II.

To get a small burst size, we should use PFQ with small λ_k^i and δ_k^i, especially small δ_k^i. From Theorem 1 of [24], WF^{2}Q (Worst-case Fair Weighted Fair Queueing) yields $\lambda_k^i = L_{max}^i$ and a very small $\delta_k^i = (1 - \frac{r_k}{C^i})\ell_k^{max}$ for flow k at server S_i with capacity C^i. Therefore, if we use H-WF^{2}Q at the aggregator, the burst size becomes

$$\begin{aligned}\sum_{k\neq f}\tilde{\sigma}_k &\leq \sum_{k\neq f}\sigma_k + [\ell_A^{max} + (1 - \frac{r_f}{R})\ell_f^{max}] \\ &\quad + (1 - \frac{r_f}{R})[L_{max}^i + (1 - \frac{R}{C^i})\ell_A^{max}] \\ &\leq \sum_{k\neq f}\sigma_k + L_{max}^i + \ell_f^{max} + 2\ell_A^{max}. \end{aligned} \quad (27)$$

Corollary 1: Using the H-WF^{2}Q algorithm at the aggregator, and under the same condition of Theorem 5, the e2e delay d_f^j is bounded as follows:

$$\begin{aligned} d_f^j &\leq \frac{\sigma_f}{r_f} + \frac{\sum_{k\neq f}\sigma_k + 2(\ell_f^{max} + L_{max}^1) + (K-1)\ell_A^{max}}{R} \\ &\quad + \frac{\ell_f^{max}}{r_f} + \sum_{i=1}^{K}\alpha^i. \end{aligned} \quad (28)$$

The proof is similar to that of Theorem 5. Refer to the Appendix I for details [25].

In general, the bound in Eq. (28) is tighter than that in Eq. (21), especially when the number of flows (N) is large. However, as can be seen, the bound in Eq. (28) is still affected by the average burst ratio of other flows in the same aggregate.

D. Multiple Aggregations

The derivation of the two delay bounds (using work-conserving aggregators) in this section relies on the fact that the incoming traffic conforms to the token-bucket model with known parameters. Inside an aggregation region, the traffic pattern (the burst size, in particular) will distort and become unpredictable. If further aggregation is done inside an aggregation region, this change of traffic pattern makes deriving delay bound very difficult (if not impossible). In fact, the two delay bounds in this section are derived under the condition that aggregation is done only once. Therefore, in the multiple aggregation case, they are not true e2e delay bounds, but rather delay bounds between two nodes (between which the traffic of interest experiences only one aggregation and split) along the e2e path. To use the two delay bounds to derive true e2e bounds, the traffic has to be reshaped before every aggregator to make it conform to token-bucket model. The shaping causes extra shaping delay, which is not considered in the paper, though.

IV. Non-Work-Conserving Aggregator

In the previous section, work-conserving scheduling algorithms were used at aggregators, and the derivation of the delay bounds required that the incoming traffic pattern at the aggregator be known. This requirement limits the ability to do multiple aggregations inside an aggregation region, since the traffic pattern inside the region is difficult to predict. The authors of [26] proved that reshaping a flow inside the network won't change its e2e delay bound. However, the proof was done for a single flow, and it is not clear if reshaping an aggregate flow will change the e2e delay bound of a constituent flow of the aggregate. We conjecture that the delay bound will be affected.

Cobb [16], [17] overcame this difficulty by using non-work-conserving scheduling algorithms at aggregators. As stated in [27], [28], non-work-conserving scheduling algorithms have the following features: (i) the burstiness of traffic can be controlled; (ii) the sum of the per-hop bounds can tightly bound the e2e delay and jitter; and (iii) since the non-work-conserving scheduler shapes the traffic at each hop, it is easy to bound the performance of heterogeneous networks. Both the *basic fair aggregator* and *greedy fair aggregator* in [17] use non-work-conserving scheduling algorithms at aggregators to shape the outgoing traffic aggregate. This way, very few packets in the same aggregate queue up at later hops, which, in turn, makes the queueing delay at those hops very small. However, the hidden assumption in [17] is that there is only one output aggregate from each aggregator, which is not the general case. In this section, we extend the result in [17] by allowing multiple outgoing aggregates from an aggregator and derive the e2e delay bound under the token bucket traffic model.

A. Delay Bound III: Non-Work-Conserving Aggregator

First, we define a new type of fair aggregator—*rate-controlled fair aggregator*.

Definition 6 (Rate-Controlled Fair Aggregator): Server S_i is said to be a *rate-controlled fair aggregator* if (i) it is a two-level hierarchical GR scheduler; (ii) each lower-level scheduler handles the flows belonging to one aggregate with capacity R, which is the sum of the guaranteed rates for all the constituent flows of the aggregate; and (iii) the upper level is the scheduler for all the aggregates.

Note that the hierarchical scheduling algorithm defined here is different from the one used in Section III-C: it is non-work-conserving, and the lower-level are constant-rate servers.

A packet at a lower-level scheduler won't be sent to the upper-level scheduler if the capacity of that scheduler does not permit it, even when the upper-level scheduler is idle. In contrast, in the ordinary H-PFQ, the lower-level schedulers are variable-rate servers [23]. Since the lower-level schedulers are constant-rate servers, in what follows, we view the lower-level schedulers and the upper-level scheduler as two virtual hops.

Lemma 4: Suppose server S_i is a rate-controlled fair aggregator, with one of the lower-level schedulers, $S_{i,l}$, serving N incoming flows (flow f is one of them). The output of $S_{i,l}$ is the aggregate A. $S_{i,h}$ is the upper-level scheduler dealing with all the aggregates. Suppose at $S_{i,l}$, the N flows have guaranteed rate r_k, $1 \le k \le N$, respectively, and at $S_{i,h}$ the guaranteed rate for A is $R = \sum_{k=1}^{N} r_k$. Let the j^{th} packet of A correspond to the j'^{th} packet of flow f (i.e., $p_A^j = p_f^{j'}$). Then, we have

$$GRC^{i,h}(p_A^j) \le GRC^{i,l}(p_f^{j'}) + \alpha^{i,l}, \tag{29}$$

where $\alpha^{i,l}$ is the scheduling constant at $S_{i,l}$ (with capacity $R = \sum_{k=1}^{N} r_k$).

Proof: Without loss of generality, we consider only one busy period of aggregate A at $S_{i,h}$. Since the lower-level server $S_{i,l}$ is rate-controlled with capacity R, the start times of the transmission of two consecutive packets (p_A^j and p_A^{j+1}) in the busy period are separate by a interval of $\frac{\ell_A^j}{R}$. Also, since $S_{i,l}$ and $S_{i,h}$ are two virtual nodes, the virtual link capacity between them is infinite, i.e., $C \to \infty$. Thus the transmission time of a packet is infinitely small. Therefore, we have

$$A^{i,h}(p_A^j) = \begin{cases} A^{i,l}(p_A^j), & j = 1 \\ A^{i,h}(p_A^{j-1}) + \frac{\ell_A^{j-1}}{R}, & j > 1. \end{cases} \tag{30}$$

Next, we prove that for all $j \ge 1$,

$$GRC^{i,h}(p_A^j) = A^{i,h}(p_A^j) + \frac{\ell_A^j}{R}. \tag{31}$$

Prove by induction on j.

Base Case: $j = 1$. Since it is the first packet of the busy period, we have

$$GRC^{i,h}(p_A^1) = A^{i,h}(p_A^1) + \frac{\ell_A^1}{R}$$

Induction Hypothesis: Suppose Eq. (31) holds for $1 \le j \le m$.

Induction Step: $j = m + 1$. From the definition of GRC value, we have

$$GRC^{i,h}(p_A^{m+1}) = \max\{A^{i,h}(p_A^{m+1}), GRC^{i,h}(p_A^m)\} + \frac{\ell_A^{m+1}}{R}.$$

However, from Eq. (30) and the *Induction Hypothesis* step, we know

$$A^{i,h}(p_A^{m+1}) = A^{i,h}(p_A^m) + \frac{\ell_A^m}{R} = GRC^{i,h}(p_A^m),$$

$$\Longrightarrow \quad GRC^{i,h}(p_A^{m+1}) = A^{i,h}(p_A^{m+1}) + \frac{\ell_A^{m+1}}{R}.$$

Also, since $S_{i,l}$ is a GR server, by definition we have

$$A^{i,h}(p_A^j) + \frac{\ell_A^j}{R} \le GRC^{i,l}(p_A^j) + \alpha^{i,l}, \quad \forall j \ge 1.$$

Since $p_A^j = p_f^{j'}$ and thus $\ell_A^j = \ell_f^{j'}$, from Eq. (31) we obtain

$$GRC^{i,h}(p_A^j) \le GRC^{i,l}(p_f^{j'}) + \alpha^{i,l}, \quad \forall j \ge 1.$$

■

From Lemma 4, one can easily see that the rate-controlled fair aggregator is a fair aggregator. Next, we derive the e2e delay bound and show that by using non-work-conserving scheduling algorithms at the aggregator, the e2e delay bound of aggregate scheduling can be decreased.

Theorem 7: Suppose flow f traverses K hops of GR servers $S_1, S_2, \ldots, S_K$, and S_1 is a rate-controlled fair aggregator, which bundles f and other (N-1) flows into aggregate flow A. S_K is the deaggregator for A. Suppose flow k has a guaranteed rate r_k ($1 \le k \le N$) at S_1 and S_K, and the guaranteed rate for aggregate A at $S_2, S_3, \ldots, S_{K-1}$ is $R = \sum_{k=1}^{N} r_k$. Then, for packet p_f^j, the e2e delay d_f^j satisfies:

$$d_f^j \le [GRC^{1,l}(p_f^j) - A^1(p_f^j)] + (K-2)\frac{\ell_A^{max}}{R} + \alpha^{1,l} + \frac{\ell_f^{max}}{r_f} + \sum_{i=1}^{K} \alpha^i. \tag{32}$$

The proof is similar to that of Theorem 5. Refer to Appendix III for a detailed proof.

Corollary 2: If flow f conforms to the token bucket model (σ_f, ρ_f) and $\rho_f \le r_f$, the e2e delay result in Theorem 7 becomes:

$$d_f^j \le \frac{\sigma_f}{r_f} + (K-2)\frac{\ell_A^{max}}{R} + \frac{\ell_f^{max}}{r_f} + \alpha^{1,l} + \sum_{i=1}^{K} \alpha^i. \tag{33}$$

Note that due to the rate-control mechanism at aggregators, the e2e bound does not depend on the burstiness of other flows in the same aggregate. Thus, the bound is tighter than those of the work-conserving cases. If scheduling algorithms, such as PGPS, VC, and WF^{2}Q, are used at the rate-controlled fair aggregator S_1, we have $\alpha^{1,l} = \frac{\ell_A^{max}}{R}$. Then Eqs. (32) and (33) can be simplified as

$$d_f^j \le [GRC^{1,l}(p_f^j) - A^1(p_f^j)] + (K-1)\frac{\ell_A^{max}}{R} + \frac{\ell_f^{max}}{r_f} + \sum_{i=1}^{K} \alpha^i \tag{34}$$

and

$$d_f^j \le \frac{\sigma_f}{r_f} + (K-1)\frac{\ell_A^{max}}{R} + \frac{\ell_f^{max}}{r_f} + \sum_{i=1}^{K} \alpha^i, \tag{35}$$

respectively.

In addition, the derivation of the bound in Eq. (33) does not require the knowledge of the traffic pattern of the incoming flows. This enables us to derive the delay bound for multiple aggregation cases. For simplicity, in the following discussion, we use the term $GRC^i(p_f^j)$ to represent $GRC^{i,l}(p_f^j)$. Also, we assume that scheduling algorithms such as PGPS, VC, and WF^{2}Q are used at the rate-controlled fair aggregators. Therefore, the delay bound for single aggregation is in the form of Eq. (34).

B. Multiple Aggregations

Theorem 8: Suppose flow f traverses K hops of GR servers. Any hop can be an aggregator and all the aggregators are rate-controlled fair aggregators. For each aggregator, there is a corresponding deaggregator at a later hop. Then, the e2e delay for packet p_f^j, d_f^j, satisfies:

$$d_f^j \le [GRC^1(p_f^j) - A^1(p_f^j)] + \sum_{i=2}^{K} \frac{\ell_{\hat{A}_i}^{max}}{\hat{R}_i} + \sum_{i=1}^{M} \frac{\ell_{A_i}^{max}}{R_i} + \sum_{n=1}^{K} \alpha^n, \tag{36}$$

where M is the number of aggregators along the path; $\hat{A}_i$ is the aggregate flow that contains flow f at S_i, $\ell_{\hat{A}_i}^{max}$ is

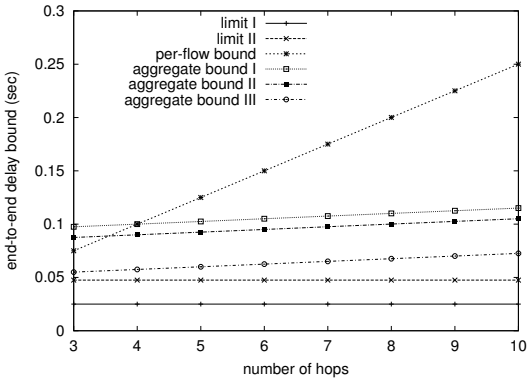


Fig. 4. Comparison of Deterministic Delay Bounds

the maximum packet size in aggregate flow $\hat{A}_i$, and $\hat{R}_i$ is the guaranteed rate for $\hat{A}_i$ at S_i; A_i is the i^{th} aggregate along the path that contains flow f, and $\ell_{A_i}^{max}$ and R_i are the maximum packet size in the aggregate and its guaranteed rate, respectively.

To prove the theorem, we first prove two lemmas that consider two basic cases of multiple aggregations: pure recursive aggregation and pure sequential aggregation. Refer to Appendix IV for a detailed proof.

Note that there are a total of (K - 1) + M terms related to packet size in the above delay bound. They can be understood as follows: the (K-1) terms in $\sum_{i=2}^{K} \frac{\ell_{\hat{A}_i}^{max}}{\hat{R}_i}$ correspond to the delays at all the hops (except the first hop). In addition, for each aggregation with guaranteed rate R_i, there is a term $\frac{\ell_{A_i}^{max}}{R_i}$ as overhead. Compared to the delay bound for per-flow scheduling in Eq. (10), Eq. (36) has M more terms due to aggregation. However, since the guaranteed rates for aggregate flows are much higher at the routers in the aggregation region, the total delay bound in Eq. (36) can be tighter than that in Eq. (10). Further, if there is only one aggregate (M=1), and S_1 and S_K are the aggregator and deaggregator, respectively, Eq. (36) is just Eq. (34).

V. Evaluation

In this section, we first use some sample data to numerically calculate the deterministic delay bounds derived so far, and compare them with that of per-flow scheduling. Then, we perform extensive simulations to compare the e2e delays of aggregate and per-flow scheduling.

First, the deterministic bounds are compared using the data in Table 1 of [5]: $\sigma_f = 100$B, $r_f = 32Kb/s$, $C^i = 150Mb/s$. All packets are of the same size, 100B, and there are 10 identical flows that make up an e2e aggregate. To see the effect of the number of hops, we vary K from 3 to 10. As shown in Fig. 4, all the bounds increase linearly with the number of hops, but those for aggregate scheduling increase much more slowly. When the number of hops is large, the delay bounds of aggregate scheduling are smaller than that of per-flow scheduling. The *per-flow bound* is computed from Eq. (10); the *aggregate bound* I, II and III are computed from Eqs. (21), (28) and (35), respectively; and the *limit* I and II

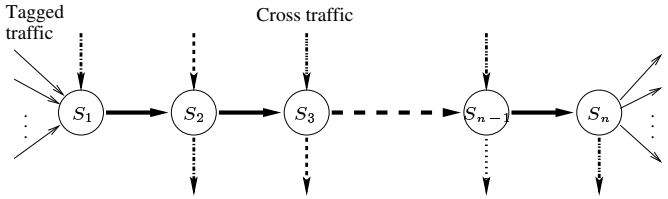


Fig. 5. Simulation Topology

are computed from $\frac{\sigma_f}{r_f}$ and $\frac{\sigma_f}{r_f} + \frac{\sum_{k \neq f} \sigma_k}{\sum_k r_k}$, respectively, which are independent of the number of hops.

From this example, one can see that, if the degree of burstiness of the flows is not very high, the delay bounds of aggregate scheduling can be tighter than that of per-flow scheduling. In addition, it shows that hierarchical aggregator and rate-controlled hierarchical aggregator can provide even tighter delay bounds. Moreover, packet size and hop count play a big role in the delay bound under per-flow scheduling, which is much larger than $\frac{\sigma_f}{r_f}$. In contrast, they make much less impact on the delay bounds of aggregate scheduling. The result also shows that aggregation is more advantageous if the number of hops is large, similar to the conclusion in [29].

Next, extensive simulation is conducted to compare the actual delay performance of aggregate and per-flow scheduling. The objective of our simulation is twofold: (i) to compare the deterministic delay bounds with the worst-case delay from the simulation and see how tight/loose the deterministic bounds are; and (ii) to compare the worst-case delays of aggregate and per-flow scheduling. We intend to find conditions under which aggregate scheduling outperforms per-flow scheduling in terms of e2e delay.

A. Simulation Setup

In the simulation, we used the *ns2* [30] simulator and the topology in Fig. 5 which is widely-used elsewhere, e.g., [31], [32]. In this topology, a number of 'tagged' flows enter the network at the ingress node S_1, and traverse all the other nodes until they reach the egress node S_n. The 'tagged' flows are those of interest to our study; their e2e delays are checked at the egress node. In order to simulate interference by cross-traffic, external traffic is injected at every node on the path. The cross-traffic at each node shares the path with the tagged traffic for only one hop before exiting the network at the next hop. For backbone links, we set the bandwidth to 160Mb and the propagation delay to 2ms, respectively, while for incoming and outgoing links, we set the bandwidth to 10Mb and the propagation delay to 10ms. The total number of tagged flows is fixed at 128, which is divided into multiple aggregates in the aggregate scheduling cases.

The tagged flows are generated by using a modified CBR model with varying packet and burst sizes. Each incoming tagged flow is shaped by a token bucket. The cross traffic is generated by using the Pareto On/Off distribution [33], [34], which can simulate long-range dependency and is known to be suitable for a large volume of traffic.

The *ns2* version of WFQ (Weighted Fair Queueing) was used as the GR scheduler at each backbone node for both

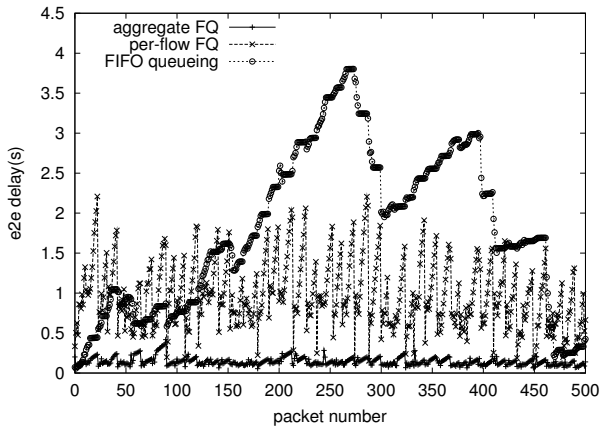


Fig. 6. End-to-End Delay Comparison

TABLE II
PARAMETERS AND THEIR VALUES IN ANOVA TEST

Parameters	Values
Tagged flow packet size	100B, 1000B
Cross traffic packet size	100B, 1000B
Tagged flow rate	3.2Kbps, 32Kbps
Hop count	5, 20
Burst size[a] of tagged flows	2, 10
Number of tagged flows in one aggregate	4, 128
Link utilization	5%, 50%
Shape[b] of cross traffic	1.1, 1.9

[a] *The burst size is a relative value: value k means the burst size is k times of the default packet size.*

[b] *The 'shape' parameter is used by the Pareto traffic model, which affects the burstiness of the traffic. We use it to vary the burstiness of the cross traffic.*

per-flow and aggregate scheduling. For aggregate scheduling, two versions of aggregator were used: work-conserving stand-alone aggregator and non-work-conserving rate-controlled (RC) aggregator. To support the non-work-conserving aggregator, a version of rate-controlled fair queueing was implemented.

- *Parameters*
  To run simulations under different scenarios, we vary several parameters, including:
  - packet size of the tagged flows
  - packet size of cross traffic
  - flow rate of the tagged flows
  - the number of hops the tagged flows travel
  - the number of flows in each aggregate
  - the link utilization
  - the burstiness of the tagged flows
  - the burstiness of the cross traffic

- *Metrics*
  The main performance metric is the worst-case e2e delay. For each scenario, 36 independent runs were conducted. All the results are plotted with the 95% confidence interval [35].

B. Simulation Results

1) A Typical Result: Fig. 6 shows a typical result of e2e delays under three different scheduling algorithms—per-flow FQ, aggregate FQ (which uses stand-alone work-conserving aggregator), and FIFO queueing. As can be seen, the e2e delay under aggregate FQ is most stable. Aggregate FQ yields not only the smallest worst-case delay but also very small delay variation. In contrast, per-flow FQ yields larger worst-case delays and delay variations; the delay under FIFO has the largest fluctuation and the worst performance. The above results were obtained by using: burst size 10000B; packet size 1000B; tagged flow rate 32Kb/s; hop count K = 15; the number of flows in the aggregate N = 16; and link utilization 55%. With these values, we can compute the deterministic bounds from Eqs. (10) and (21). The bounds turn out to be 6s for per-flow FQ and 5.53s for aggregate FQ. Both bounds are much larger than the worst-case delay found from the simulation, implying that they are rather pessimistic.

2) Homogeneous Case: To find the main factors that affect the delay performance of both aggregate and per-flow scheduling, we first used homogeneous flows in each aggregate. We used the $2^k \cdot r$ *factorial design* method [35] to evaluate the contribution of different parameters. Eight parameters ($k = 8$) were used, each with two different values. Each scenario was run 36 times ($r = 36$). The parameters and their values are summarized in Table II.

After collecting data, we used statistical tool ANOVA (ANalysis Of VAriance) [35] to analyze the significance of each parameter. Our focus was on the value $d_r = \frac{d_a}{d_f}$, where d_a (d_f) is the worst-case delay under aggregate (per-flow) scheduling. The intention was to find which parameters affect most the relative delay performance between aggregate and per-flow scheduling. The second parameter—packet size of cross traffic—turned out to have little effect on value d_r. This is easy to explain, since from the delay bounds in Eqs. (10), (21) and (35), the packet size of cross-traffic shows up only in the term $\frac{L^i_{max}}{C^i}$, which is usually negligible since C^i is very large. Therefore, we fixed its value at 1500B in all of the following simulations.

To illustrate the effect of the factors on the e2e delay, we ran multiple sets of simulation. For each set of simulation, we varied only one parameter with all the others fixed. All the default values of the parameters are summarized in Table III; the simulation results are summarized in Fig. 7. In Figs. 7 and 8, as well as the following discussion, 'aggregate FQ' stands for aggregate scheduling using stand-alone fair queueing; 'RC aggregate FQ' stands for aggregate scheduling using rate-controlled (RC) fair queueing.

Let us compare the performance of per-flow and aggregate FQ first. In Fig. 7 (a), as the burst size of the flows increases, the worst-case delay under per-flow FQ increases significantly faster than that under aggregate FQ. Surprisingly, the delays of aggregate FQ do not increase as fast as expected. This behavior can be attributed to the multiplexing gain of aggregate scheduling. In other words, although all flows become much burstier, the probability that all flows reach their peaks (of load) at the same time is very low. Instead, the peaks and

TABLE III
DEFAULT PARAMETER VALUES

Parameters	Values
Tagged flow packet size	400B
Tagged flow rate	32Kbps
Hop count	10
Burst size of tagged flows	2
Number of tagged flows in each aggregate	16
Link utilization	25%
Shape of cross traffic	1.5
Cross traffic packet size	1500B
Total number of tagged flows	128

valleys are more likely to be evened out.

In Fig. 7 (b), as the flow rate increases, the worst-case delay under per-flow FQ decreases faster than that under aggregate FQ. Thus aggregate FQ is shown to be more advantageous for lower-rate flows. In Fig. 7 (c) and (d), as the packet size (hop count) increases, the delay under per-flow FQ increases faster than that under aggregate FQ. Thus aggregate FQ is shown to be more advantageous when the packet size (hop count) is large. In Fig. 7 (e), as network link utilization increases, the delay under per-flow FQ increases dramatically faster than that under aggregate FQ, which shows that when the network utilization is high, aggregate FQ becomes more advantageous. In Fig. 7 (f), as the number of flows in an aggregate increases, the e2e delay under aggregate FQ decreases, while that under per-flow FQ remains unchanged. Aggregate FQ is shown to be more advantageous when the number of flows aggregated gets larger. However, as the number increases, the pace of increase becomes smaller since the margin of multiplexing gain decreases.

The effects of flow rate, packet size and hop count can be easily explained by the deterministic bounds. Since GR scheduling algorithms are rate-based, the delay of a flow is coupled with its reserved rate. Although aggregate scheduling has the same problem as per-flow scheduling, the reserved rate for an aggregate (R_A) is much larger than that of a single flow (r_f), thus the problem is not as severe as in per-flow scheduling. As for packet size, according to Eqs. (10) and (21), the delay bounds are proportional to the maximum packet size in a flow (aggregate). With aggregation, however, R_A is much larger than r_f. Thus the delay bound increases much faster under per-flow scheduling. The same holds true for hop count. The effect of link utilization is also related to the coupling problem of GR scheduling. When the link utilization becomes higher, there is less spare bandwidth left. Therefore, the rate for a flow (aggregate) is decreased to the reserved rate for it. Since R_A is much larger than r_f, aggregate scheduling has less increase in delay. Finally, since per-flow scheduling is independent of the number of flows in an aggregate (N), its delay should not change with N. For aggregate scheduling, from Eq. (21), two terms are related to N, the number of flows: $\frac{\sum_{k=1}^{N} \ell_k^{max}}{R}$ and $(K-3)\frac{\ell_A^{max}}{R}$. Since we are considering identical flows, the first term does not vary with the number of flows. However, R increases proportionally to N. Thus the second term decreases slightly, and then the total e2e delay decreases.

Now look at the results for RC aggregate FQ. In all the scenarios, the worst-case delays under RC aggregate FQ follow the same trend as those under aggregate FQ. However, since the default link utilization if only 25%, RC aggregate FQ performs worst than aggregate FQ; in most cases, its performance is even worst than that of per-flow FQ. This is mainly because RC aggregate FQ is non-work-conserving and does not take advantage of spare bandwidth. On the other hand, as Fig. 7 (e) shows, when the link utilization becomes larger, RC aggregate FQ still performs better than per-flow FQ, and the difference between RC aggregate FQ and aggregate FQ becomes smaller also.

We also ran simulations by varying the burstiness of cross-traffic. As the burstiness of cross traffic increases, the worst-case delay for per-flow scheduling increases faster than those under the two aggregate FQ cases, showing that aggregate scheduling is more robust to the burstiness of cross traffic than per-flow scheduling.

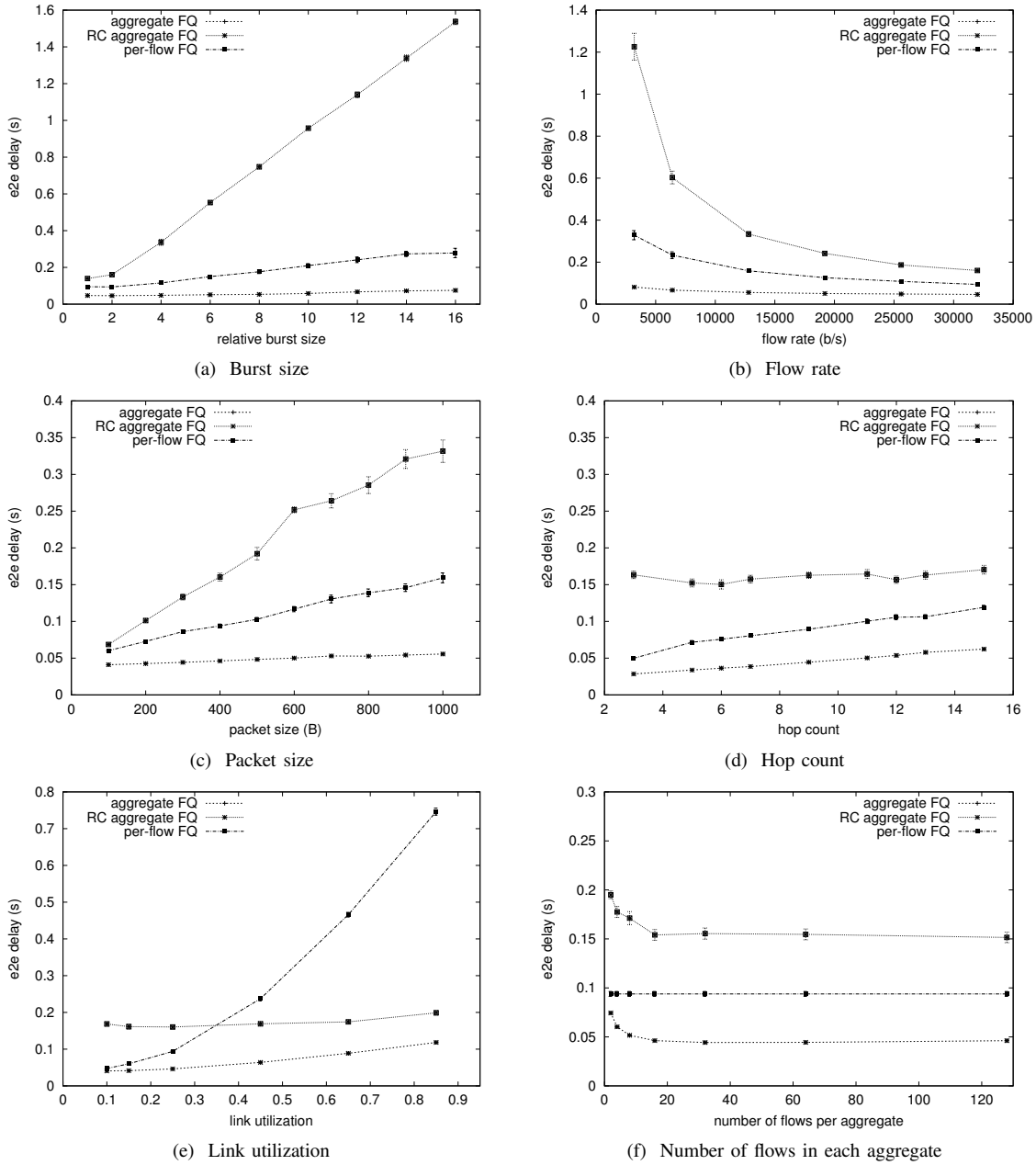
3) Heterogeneous Case: After studying the impact of different parameters by using homogeneous flows in a traffic aggregate, we mix heterogeneous flows (in terms of packet size, flow rate, and burst ratio) into the same aggregate to see what kind of flows are suitable to be aggregated together. In the following simulations, we aggregate 16 flows together: 15 of them are identical flows; only one flow is different from the rest. We monitor the worst-case delay of this flow when the parameters of other flows vary.

Fig. 8 (a) shows the result when the small-burst flow (with relative burst size 1) is aggregated with larger-burst flows. As the burst size of the other flows increases, the worst-case delay under aggregate FQ increases quickly. As expected, mixing a flow with other larger-burst flows will hurt its delay performance. Note that the performance of RC aggregate FQ is very stable since the aggregator controls the burstiness of the output aggregate.

Fig. 8 (b) plots the result when a high-rate flow (with rate 128Kbps) is aggregated with low-rate flows. As rate of other flows gets smaller, the delay under both aggregate FQ and RC aggregate FQ increases, with that under aggregate FQ increases faster and eventually becomes larger than the delay under per-flow FQ. From Eq. (21) we can see that as the rates of other flows decrease, R decreases, thus all three terms—$\frac{\sum_{k\neq f} \sigma_k}{R}$, $\frac{\sum_{k=1}^{N} \ell_k^{max}}{R}$ and $(K-3)\frac{\ell_A^{max}}{R}$—increase, and the total delay increases. Due to the rate-control at the aggregator, the delay increase under RC aggregate FQ is slower.

Fig. 8 (c) shows the result when a small-packet flow (with default size 100B) f_s is aggregated with large-packet flows. When the packet size of other flows is becomes larger, the delay performance of f_s is shown to suffer—leading to even larger delay than that under per-flow FQ and RC aggregate FQ. This is because when aggregating with large-packet flows, a small packet has to wait behind other large packets in the same aggregate. This implies that flows of similar packet size should be aggregated together.

For comparison, we also mix a flow with other flows with



(a) Burst size

(b) Flow rate

(c) Packet size

(d) Hop count

(e) Link utilization

(f) Number of flows in each aggregate

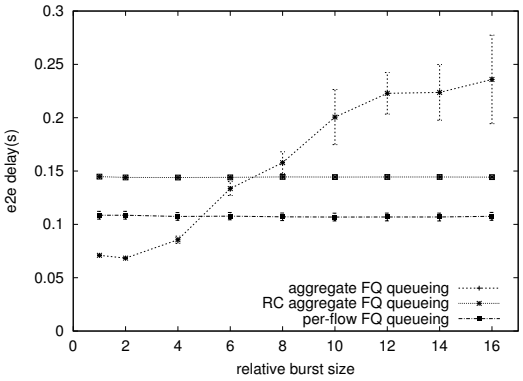
Fig. 7. Comparison of Aggregate and Per-flow Scheduling

smaller burst size, larger flow rate, and smaller packet size. The results show little or marginal change on the worst-case delay of aggregate scheduling. Similarly, the results can be explained by the delay bound in Eqs. (21) and (35).

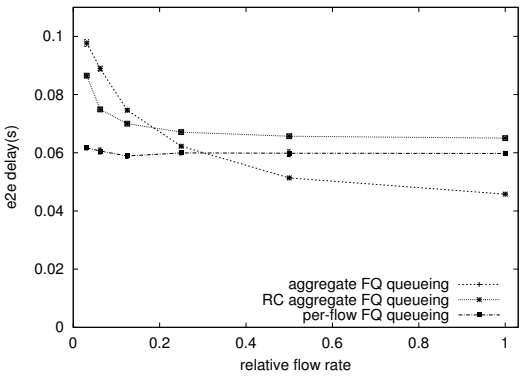
4) Case Study with MPEG Traces: To further compare the delay performance of aggregate FQ and per-flow FQ, we also used real MPEG-4 traces [36] in the simulation. Two video traces were used—high rate "Soccer" and low rate "Silence of the Lambs". The parameters of these two traces are shown in Table IV. They have very different burstiness, packet rate, packet rate variation. With the same simulation setup as before, we mixed 8 tagged flows driven by the "Soccer" trace and 8 other tagged flows driven by the "Silence of the Lambs" trace into an aggregate. The results are summarized in Fig. 9, which shows that for flows driven by both traces, aggregate FQ provides smaller maximum e2e delays; also, the improvement is much larger for burstier flows driven by the "Silence of the Lambs" trace.

VI. Related Work and Discussions

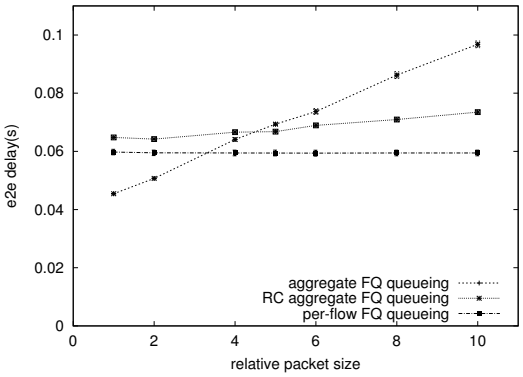
Aggregate scheduling has been studied extensively in the literature. In [37], [38], the authors proposed some grouping techniques to optimize the implementation of fair queueing. Certain flows (with similar throughput parameters) are grouped together. For example, the scheme in [38] is confined to ATM networks, in which the routers support only a fixed number of rates, and all the flows of the same rate are placed into a single group. It takes advantage of the fact that all the cells have the same size and all the flows in the same group have the same rate, thus simplifying the sorting of flows. However,



(a) Larger burst size



(b) Smaller flow rate



(c) Larger packet size

Fig. 8. Aggregate Heterogeneous Flows

it still uses per-flow-based scheduling. Although these two algorithms are sometimes called *aggregate scheduling*, they are different from the aggregate scheduling we are considering, because the core routers still recognize *each individual* flow. They are just efficient implementations of per-flow-based fair queueing. Other work on implementations of fair queueing include [39], [40], [41].

The authors of [42] studied QoS guarantees under aggregation (e2e aggregation was called *grouping*). Similar to our study, fair queueing is used to handle aggregates at core routers (or in aggregation regions). Based on some given e2e delay requirement, they derived the bandwidth and buffer requirements by using the IETF Guaranteed Service (GS) traffic specification (TSpec), and demonstrated the advantages in aggregating flows of equal or similar delay requirements.

TABLE IV
PARAMETERS OF THE VIDEO TRACES

Video Name	Frame Size (KB)		Bit Rate (Kbps)		Burst Size
	Mean	Peak	Mean	Peak	(Kb)
Soccer	5.53	17.93	1,106.6	3,585.4	140
Silence*	0.53	11.29	105.6	2,258.4	70

*Silence of the Lambs

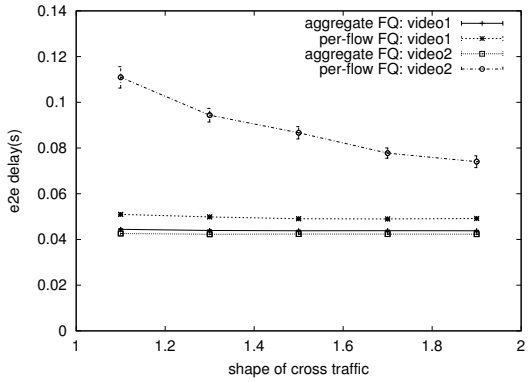


Fig. 9. Delay Results for Video Traces

By contrast, we derived the e2e delay bound under a given bandwidth guarantee without considering any buffer requirement.

Although the main focus of [22] is QoS routing, it discussed flow aggregation by defining the notion of '*burst ratio*.' For flows conforming to the token bucket model, the burst ratio r is the ratio of the burst size to the average rate, or σ/ρ. The authors suggested aggregation of flows of same or similar burst ratio, since flows with the same burst ratio can be merged and divided without changing the burst ratio of the resulting flows. This conclusion is the same as ours in the work-conserving case in Section IV. The authors of [22] analyzed some e2e delay for fluid traffic model, without considering any non-fluid traffic model. In contrast, our delay bound analysis is based on packet traffic model.

Cobb's work [16], [17] is closest to ours. He studied the delay bound problem of aggregate scheduling by using rate-based scheduling algorithms. The core routers treat each aggregate as a single flow, and handle all the aggregates by using rate-based fair queueing. He defined a type of fair aggregator and showed that, by using rate-based fair queueing, the e2e delay is bounded. Such aggregator can be used to aggregate the traffic recursively, and the e2e delay is still bounded. He also proposed two types of fair aggregator, called the *basic fair aggregator* and *greedy fair aggregator*. By using such aggregators, he showed that the e2e delay bound can be even smaller than the per-flow e2e delay bound.

We have several observations on the results in [16], [17]. First, the main idea of its *fair aggregator* is to control the burst of an aggregate. Such an aggregator works in a non-work-conserving way. Second, the hidden assumption of the *basic fair aggregator* and *greedy fair aggregator* is that only one aggregate goes out of an aggregator. In other words, all

the flows going through the same outgoing link belong to the same aggregate. In contrast, our work is more general. In Section III we showed that the delay bounds exist even for work-conserving aggregators. In Section IV, we extended the result in [17] by allowing multiple aggregates going out of the same link of an aggregator, and derived the e2e delay bound under token bucket traffic model.

DiffServ and Aggregate GR scheduling

The main difference between DiffServ and the proposed aggregate GR scheduling architecture is the way packets in the same class are scheduled. Although it is not specified in the DiffServ standard documents, FIFO queueing is usually used for packet scheduling within each class, thus resulting in coarse-grained control of delay performance.

In the context of DiffServ, the authors of [5] studied the worst-case delay bound of FIFO queueing, showing unsatisfactory performance of it. For a certain class of traffic, such as EF, the e2e delay bound exists only if the network utilization of that class is sufficiently low. In such a case, the delay bound is a function of the utilization of any link in the network, the maximum hop count of any flow, and the shaping parameters at the network ingress nodes. However, if the utilization is high, the e2e delay could be unbounded. Specifically, Charny [5] showed that for any given values of delay, hop count and utilization, it is possible to construct a network in which the e2e queueing delay experienced by some flow exceeds the chosen delay value. The main reason for this unbounded delay is that the e2e delay experienced by a packet under FIFO queueing is severely affected by the cross traffic. Moreover, the e2e delay depends not only on the behavior of the flows that directly share queues with the packet in question, but also on the behavior and arrival pattern of flows in the entire network (which may not directly share queues with that packet). Charny [5] showed that, for real-time traffic such as voice, achieving an acceptable delay bound under FIFO queueing requires the network utilization of such traffic to be rather low.

In aggregate GR scheduling architecture, the definition of traffic aggregates is more flexible and finer-grained than the DS traffic classes. Traffic within the same DS class can be grouped into multiple different aggregate flows that can be differentiated in aggregation regions by using the fair queueing algorithms of IntServ. Moreover, at aggregators, flows are bundled into aggregates by more intelligent scheduling schemes, instead of FIFO queueing.

Advantages and Disadvantages of Traffic Aggregation

With traffic aggregation, the core routers need to maintain fewer states, making packet classification and scheduling simpler. Flow aggregation is also beneficial to the flows with low bandwidth requirements, because, under GR scheduling algorithms—which couples bandwidth and delay—the flows with low bandwidth requirements will suffer long delays; but aggregation usually alleviates this problem. More importantly, as shown earlier, aggregate-based GR scheduling can provide guaranteed QoS, such as e2e delay bound.

However, aggregation also comes with its own disadvantages: the flows in the same aggregate cannot isolate from, and protect against, other flows. Therefore, the QoS guarantee of a flow will be affected by others in the same aggregate, and all the flows within an aggregate will receive roughly the same service. Due to the lack of isolation within an aggregate, bursty flows can 'steal' the bandwidth from well-behaving flows and unduly degrade their QoS. This problem can become worse in multi-service networks [29]. However, as pointed out in [22], the problem can be controlled if flows are selectively aggregated, i.e., only those flows that have some common characteristics are aggregated (which is similar to our conclusion; see the results in Section III). Moreover, 'fair' aggregation can also alleviate this problem. In this paper, we have shown this feature by using GR scheduling algorithms for aggregate flows.

VII. Conclusions and Future Work

In this paper, we first derived deterministic delay bounds for aggregates under the assumption that the incoming traffic at each aggregator conforms to the token bucket model and guaranteed-rate (GR) scheduling algorithms are used in each aggregation region. We considered three types of GR scheduling algorithms at an aggregator: stand-alone, two-level hierarchical, and rate-controlled two-level hierarchical GR algorithms. The delay bounds are shown to depend on several factors, such as the scheduling constant at each hop and the latency at the aggregator. We should, therefore, use the scheduling algorithms with a small scheduling constant at each hop, and those with small latency at aggregators. Among all the rate-based scheduling algorithms, PGPS, VC, and WF^{2}Q have the smallest scheduling constant ($\frac{L^i_{max}}{C^i}$) and latency ($\frac{\ell^{max}_f}{r_f} + \frac{L^i_{max}}{C^i}$ for flow f). The delay bounds also indicate that it is beneficial to aggregate flows with similar burst ratios. Aggregate scheduling provides better e2e delay bounds when a large number of hops use aggregate scheduling, because the overhead at the aggregators will be offset by the larger guaranteed rate for an aggregate. If the number of hops is small, the aggregation overhead becomes 'relatively' significant.

We also showed by simulation that aggregate scheduling is robust. By exploiting multiplexing gains, it can provide better worst-case delay performances than per-flow scheduling, as long as those flows aggregated together are not very diverse in terms of packet size, flow rate, and burst ratio. In addition, the simulation results also showed that in most scenarios non-work-conserving aggregator performs worse than work-conserving aggregator, since it does not take advantage of the spare bandwidth in the network. Also, the deterministic bounds are shown to be rather pessimistic. Although the simulation may not capture the worst-case e2e delay, it implies that the probability for the worst-case e2e delay to happen be very small.

Note that resource reservation and admission control are not covered in this paper, but techniques such as [11] can be used for this purpose. We also assume that the e2e path in an aggregation region can be set up by traffic engineering

mechanisms similar to the way the Label Switched Path (LSP) is set up in an MPLS network.

The large gap between the deterministic delay bound and the worst-case delay found from the simulations suggests the need for investigation into the stochastic behavior of aggregate scheduling. We are currently exploring ways to find statistical delay bounds for traffic aggregates. With such bounds, one can admit more flows and enhance network utilization.

ACKNOWLEDGMENT

The authors would like to thank the members of DiffServ Group of RTCL for their insightful comments on this work.

REFERENCES

[1] R. Braden, D. Clark, and S. Shenker, "Integrated services in the Internet architecture: an overview," RFC 1633, June 1994.
[2] H. Zhang, "Service disciplines for guaranteed performance service in packet-switching networks," *Proc. IEEE*, vol. 83, no. 10, pp. 1374–1396, Oct. 1995.
[3] S. Blake, D. L. Black, M. A. Carlson, *et al.*, "An architecture for differentiated services," RFC 2475, Dec. 1998.
[4] B. Davie, A. Charny, J. C. R. Bennett, *et al.*, "An expedited forwarding PHB (Per-Hop Behavior)," RFC 3246, Mar. 2002.
[5] A. Charny and J.-Y. L. Boudec", "Delay bounds in a network with aggregate scheduling," in *Proc. of QofIS'00*, Oct. 2000, pp. 1–13.
[6] F. Baker, C. Iturralde, F. L. Faucheur, *et al.*, "Aggregation of RSVP for IPv4 and IPv6 reservation," RFC 3175, Sept. 2001.
[7] J. Ehrensberger, "Resource demand of aggregated resource reservations," in *1st European Conf. on Universal Multiservice Networks (ECUMN)*, Oct. 2000, pp. 56–61.
[8] O. Schelén and S. Pink, "Aggregating resource reservation over multiple routing domains," in *Proc. of IWQoS'98*, May 1998, pp. 29–32.
[9] A. Terzis, L. Zhang, and E. L. Hahne, "Reservations for aggregate traffic: Experiences from an RSVP tunnels implementation," in *Proc. of IWQoS'98*, May 1998, pp. 23–25.
[10] B.-K. Choi, D. Xuan, R. Bettati, *et al.*, "Scalable QoS guaranteed communication services for real-time applications," in *Proc. of IEEE ICDCS'00*, Apr. 2000, pp. 180–187.
[11] H. Fu and E. W. Knightly, "Aggregation and scalable QoS: A performance study," in *Proc. of IWQoS'01*, June 2001, pp. 39–50.
[12] S. Berson and S. Vincent, "Aggregation of Internet integrated services state," in *Proc. of IWQoS'98*, May 1998, pp. 26–28.
[13] H. Zhang and S. Keshav, "Comparison of rate-based service disciplines," in *Proc. of ACM SIGCOMM'91*, Aug. 1991, pp. 113–121.
[14] T. Li and Y. Rekhter, "A provider architecture for differentiated services and traffic engineering (PASTE)," RFC 2430, Oct. 1998.
[15] D. Awduche, J. Malcolm, J. Agogbua, *et al.*, "Requirements for traffic engineering over MPLS," RFC 2702, Sept. 1999.
[16] J. A. Cobb, "Preserving quality of service guarantees in spite of flow aggregation," in *Proc. of IEEE ICNP'98*, Oct. 1998, pp. 90–97.
[17] ——, "Preserving quality of service guarantees in spite of flow aggregation," *IEEE/ACM Trans. Networking*, vol. 10, no. 1, pp. 43–53, Feb. 2002.
[18] P. Goyal, S. S. Lam, and H. M. Vin, "Determining end-to-end delay bounds in heterogeneous networks," in *Proc. of NOSSDAV'95*, Apr. 1995, pp. 287–298.
[19] D. Stiliadis and A. Varma, "Latency-rate servers: A general model for analysis of traffic scheduling algorithms," *IEEE/ACM Trans. Networking*, vol. 6, no. 5, pp. 611–624, Aug. 1998.
[20] A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: The single node case," *IEEE/ACM Trans. Networking*, vol. 1, no. 3, pp. 344–357, June 1993.
[21] L. Zhang, "Virtual Clock: A new traffic control algorithm for packet-switched networks," *ACM Trans. Computer Systems*, vol. 9, no. 2, pp. 101–124, May 1991.
[22] S. Vutukury and J. Garcia-Luna-Aceves, "A scalable architecture for providing deterministic guarantees," in *Proc. of IEEE IC3N'99*, Oct. 1999, pp. 534–539.
[23] J. C. R. Bennett and H. Zhang, "Hierarchical packet fair queueing algorithms," in *Proc. of ACM SIGCOMM'96*, Aug. 1996, pp. 143–156.
[24] ——, "WF^{2}Q: Worst-case fair weighted fair queueing," in *Proc. of IEEE INFOCOM'96*, Mar. 1996, pp. 120–128.
[25] W. Sun and K. G. Shin, "Delay bounds for end-to-end traffic aggregate under guaranteed rate scheduling algorithms," Dept. of EECS, Univ. of Michigan, Tech. Rep., 2002. [Online]. Available: http://www.eecs.umich.edu/~wsunz/publications/aggr-sched.ps
[26] J.-Y. L. Boudec", "Application of network calculus to guaranteed service networks," *IEEE Trans. Inform. Theory*, vol. 44, no. 3, pp. 1087–1096, May 1998.
[27] H. Zhang and D. Ferrari, "Rate-controlled service disciplines," *J. of High Speed Networks*, vol. 3, no. 4, pp. 389–412, 1994.
[28] S. Keshav, *An Engineering Approach to Computer Networking*. Addison Wesley, 1997.
[29] K. Dolzer, W. Payer, and M. Eberspächer, "A simulation study on traffic aggregation in multi-service networks," in *Proc. of IEEE Conf. on High Performance Switching and Routing*, June 2000, pp. 157–165.
[30] "ns2 simulator." [Online]. Available: http://www.isi.edu/nsnam/ns/
[31] R. A. Guérin and V. Pla, "Aggregation and conformance in differentiated service networks: A case study," *ACM Computer Communication Review*, vol. 31, no. 1, pp. 21–32, Jan. 2001.
[32] J. Sahni, P. Goyal, and H. M. Vin, "Scheduling CBR flows: FIFO or per-flow queueing?" in *Proc. of NOSSDAV'99*, June 1999.
[33] W. E. Leland, M. S. Taqqu, W. Willinger, *et al.*, "On the self-similar nature of Ethernet traffic (extended version)," *IEEE/ACM Trans. Networking*, vol. 2, no. 1, pp. 1–15, Feb. 1994.
[34] A. Popescu, "Traffic self-similarity," in *Proc. of IEEE Intl. Conf. on Telecommunications (ICT2001)*, June 2001.
[35] R. Jain, *The Art of Computer Systems Performance Analysis*. John Wiley & Sons, 1991.
[36] F. Fitzek and M. Reisslein, "MPEG-4 and H.263 video traces for network performance evaluation," *IEEE Network*, vol. 15, no. 6, pp. 40–54, Nov./Dec. 2001.
[37] J. L. Rexford, A. G. Greenberg, and F. G. Bonomi, "Hardware-efficient fair queueing architectures for high-speed networks," in *Proc. of IEEE INFOCOM'96*, Mar. 1996, pp. 638–646.
[38] J. C. R. Bennett, D. C. Stephens, and H. Zhang, "High speed, scalable, and accurate implementation of packet fair queueing algorithms in ATM networks," in *Proc. of IEEE ICNP'97*, Oct. 1997, pp. 7–14.
[39] D. C. Stephens and H. Zhang, "Implementing distributed packet fair queueing in a scalable switch architecture," in *Proc. of IEEE INFOCOM'98*, Mar. 1998, pp. 282–290.
[40] F. M. Chiussi and V. Sivaraman, "Implementing fair queueing in ATM switches. I. a practical methodology for the analysis of delay bounds," in *GLOBECOM'97*, Nov. 1997, pp. 509–518.
[41] F. M. Chiussi and A. Francini, "Implementing fair queueing in ATM switches: the discrete-rate approach," in *Proc. of IEEE INFOCOM'98*, Mar. 1998, pp. 272 –281.
[42] J. Schmitt, M. Karsten, L. Wolf, *et al.*, "Aggregation of guaranteed service flows," in *Proc. of IWQoS'99*, June 1999, pp. 147–155.

APPENDIX I
PROOF OF COROLLARY 1

Proof: Similar to Theorem 4, from Lemma 3 we have

$$GRC^{i+1}(p_A^{j'}) \le GRC^i(p_f^j) + \frac{\sum_{k\neq f}\tilde{\sigma}_k + \ell_f^{max}}{R} + \alpha^i.$$

Replacing $\sum_{k\neq f}\tilde{\sigma}_k$ by (27), we obtain

$$\begin{aligned} & GRC^{i+1}(p_A^{j'}) \\ \le\ & GRC^i(p_f^j) + \frac{\sum_{k\neq f}\sigma_k + L^i_{max} + 2\ell_f^{max} + 2\ell_A^{max}}{R} + \alpha^i. \end{aligned} \quad (37)$$

Thus for the two-level hierarchical H-WF^{2}Q aggregator S_i, it is a fair aggregator with aggregation constant $\gamma^i = \frac{\sum_{k\neq f}\sigma_k + L^i_{max} + 2\ell_f^{max} + 2l_A^{max}}{R}$.

Then, similar to the proof of Theorem 5, the e2e delay d_f^j satisfies

$$d_f^j \le \frac{\sigma_f}{r_f} + \gamma^1 + (K-3)\frac{\ell_A^{max}}{R} + \frac{\ell_f^{max}}{r_f} + \alpha_H^1 + \sum_{i=2}^{K}\alpha^i. \quad (38)$$

Also, since S_1 is a hierarchical aggregator, from Fact 1 we know that its scheduling constant is larger than that of the stand-alone server:

$$\alpha_H^1 \leq \alpha^1 + \frac{\ell_A^{max} + (L_{max}^1 - \ell_A^{max})\frac{R}{C^1}}{R} \leq \alpha^1 + \frac{L_{max}^1}{R}. \quad (39)$$

Rearranging the terms in Eq. (38), we obtain Eq. (28). ■

APPENDIX II
PROOF OF THEOREM 6

Proof: For the convenience of description, we regard the other (N-1) flows in aggregate flow A as a virtual flow $\bar{f}$, and prove the lemma for any time t in flow $\bar{f}$'s backlogged period.

Consider the two-level hierarchical H-PFQ scheduler S_i. At the upper-level scheduler, aggregate flow A is treated as a single flow conforming to the token bucket model (σ_A, ρ_A), where $\sigma_A = \sum_{k=1}^{N} \sigma_k$ and $\rho_A = \sum_{k=1}^{N} \rho_k$; at the lower-level variable-rate scheduler, N flows are aggregated into A. *As pointed out in [38], by defining virtual times in unit of bits instead of in unit of seconds, many properties of constant-rate packet schedulers still hold for variable-rate packet schedulers.*

Consider any backlogged period $(\tau, t]$ of flow $\bar{f}$ at S_i. For any flow k and time t, from Eqs. (24) and (25) we obtain:

$$W_{k,GPS}^i(0,t) - W_k^i(0,t) \leq \lambda_k^i,$$
$$W_k^i(0,t) - W_{k,GPS}^i(0,t) \leq \delta_k^i.$$

Thus, for flow f at the lower-level scheduler $S_{i,l}$,

$$\begin{aligned} W_f^i(\tau,t) &= W_f^i(0,t) - W_f^i(0,\tau) \\ &\leq [W_{f,GPS}^i(0,t) - W_{f,GPS}^i(0,\tau)] + \lambda_f^i + \delta_f^i \\ &= W_{f,GPS}^i(\tau,t) + \lambda_f^i + \delta_f^i. \end{aligned} \quad (40)$$

Let $W_l^i(\tau,t)$ be the total service provided by $S_{i,l}$, a variable-rate scheduler, during $(\tau,t]$. Since flow $\bar{f}$ is backlogged during this period, f at most gets its reserved share at the corresponding GPS server at $S_{i,l}$, and $W_l^i(\tau,t)$ reaches $S_{i,l}$'s capacity during the period. Thus,

$$W_{f,GPS}^i(\tau,t) \leq \frac{r_f}{R} \cdot W_l^i(\tau,t)$$

For $t = t_k$, the time when k^{th} ($k \geq 1$) packet in aggregate A is sent out, we have

$$\begin{aligned} W_l^i(\tau,t_k) &= W_A^i(\tau,t_k), \qquad k \geq 1 \\ \Longrightarrow W_{f,GPS}^i(\tau,t_k) &\leq \frac{r_f}{R} \cdot W_A^i(\tau,t_k), \quad k \geq 1. \end{aligned}$$

For $t_k \leq t \leq t_{k+1}$ ($k \geq 0$ and assume $t_0 = \tau$), since $W_{f,GPS}^i(\tau,t) = W_{f,GPS}^i(\tau,t_k)$ and $W_A^i(\tau,t) = W_A^i(\tau,t_k)$, we obtain,

$$W_{f,GPS}^i(\tau,t) \leq \frac{r_f}{R} \cdot W_A^i(\tau,t), \quad t_k \leq t \leq t_{k+1}, k \geq 0.$$

Therefore,

$$W_{f,GPS}^i(\tau,t) \leq \frac{r_f}{R} \cdot W_A^i(\tau,t), \quad t \geq \tau. \quad (41)$$

Similarly, for aggregate A at the upper-level scheduler, we have

$$\begin{aligned} W_A^i(\tau,t) &= W_A^i(0,t) - W_A^i(0,\tau) \\ &\geq [W_{A,GPS}^i(0,t) - W_{A,GPS}^i(0,\tau)] - \lambda_A^i - \delta_A^i \\ &= W_{A,GPS}^i(\tau,t) - \lambda_A^i - \delta_A^i. \end{aligned} \quad (42)$$

From Eqs. (40), (41), and (42) we obtain

$$\begin{aligned} W_{\bar{f}}^i(\tau,t) &= W_A^i(\tau,t) - W_f^i(\tau,t) \\ &\geq [W_A^i(\tau,t) - \frac{r_f}{R} W_A^i(\tau,t)] - [\lambda_f^i + \delta_f^i] \\ &\geq (1 - \frac{r_f}{R})[W_{A,GPS}^i(\tau,t) - \lambda_A^i - \delta_A^i] - [\lambda_f^i + \delta_f^i]. \end{aligned} \quad (43)$$

Since $\bar{f}$ is backlogged during $(\tau,t]$, so must A during this period, and hence

$$W_{A,GPS}^i(\tau,t) \geq R \cdot (t - \tau). \quad (44)$$

Since τ is the start time of the backlogged period, the backlog of $\bar{f}$ at τ is 0. Thus the backlog of $\bar{f}$ at time t, $Q_{\bar{f}}^i(t)$, is

$$\begin{aligned} Q_{\bar{f}}^i(t) &= \sum_{k \neq f} A_k(\tau,t) - W_{\bar{f}}^i(\tau,t) \\ &\leq \sum_{k \neq f} [\sigma_k + \rho_k(\tau,t)] - (1 - \frac{r_f}{R}) W_{A,GPS}^i(\tau,t) \\ &\quad + (1 - \frac{r_f}{R})[\lambda_A^i + \delta_A^i] + [\lambda_f^i + \delta_f^i] \\ &\leq [\sum_{k \neq f} \sigma_k + (R - r_f)(t - \tau)] - (1 - \frac{r_f}{R}) \cdot R \cdot (t - \tau) \\ &\quad + (1 - \frac{r_f}{R})[\lambda_A^i + \delta_A^i] + [\lambda_f^i + \delta_f^i] \\ &\leq \sum_{k \neq f} \sigma_k + (1 - \frac{r_f}{R})[\lambda_A^i + \delta_A^i] + [\lambda_f^i + \delta_f^i]. \end{aligned} \quad (45)$$

Note that Eq. (45) holds for any time t. Similar to the proof of Theorem 3 in [19], we obtain the total burstiness of the (N-1) outgoing flows as:

$$\sum_{k \neq f} \tilde{\sigma}_k \leq \sum_{k \neq f} \sigma_k + [\lambda_f^i + \delta_f^i] + (1 - \frac{r_f}{R})[\lambda_A^i + \delta_A^i].$$

■

APPENDIX III
PROOF OF THEOREM 7

Proof: Let $p_A^{j'}$ be the packet corresponding to packet p_f^j of flow f. From Lemma 4 we have

$$GRC^{1,h}(p_A^{j'}) \leq GRC^{1,l}(p_f^j) + \alpha^{1,l}.$$

Also, since virtual server $S_{1,h}$ and $S_2, \cdots, S_{K-1}$ are GR servers for aggregate flow A, from Lemma 2 we have

$$\begin{aligned} GRC^2(p_A^{j'}) &\leq GRC^{1,h}(p_A^{j'}) + \frac{\ell_A^{max}}{R} + \alpha^{1,h}, \\ GRC^3(p_A^{j'}) &\leq GRC^2(p_A^{j'}) + \frac{\ell_A^{max}}{R} + \alpha^2, \\ &\vdots \\ GRC^{K-1}(p_A^{j'}) &\leq GRC^{K-2}(p_A^{j'}) + \frac{\ell_A^{max}}{R} + \alpha^{K-2}, \end{aligned}$$

where $\alpha^{1,h} = \alpha^1$. Adding them up, and with $p_A^{j'} = p_f^j$, we obtain

$$\begin{aligned} GRC^{K-1}(p_f^j) &= GRC^{K-1}(p_A^{j'}) \\ &\leq GRC^{1,l}(p_f^j) + (K-2)\frac{\ell_A^{max}}{R} + \alpha^{1,l} + \sum_{i=1}^{K-2} \alpha^i. \end{aligned}$$

$$\begin{aligned} \Longrightarrow D^{K-1}(p_f^j) &\leq GRC^{K-2}(p_f^j) + \alpha^{K-1} \\ &\leq GRC^{1,l}(p_f^j) + (K-2)\frac{\ell_A^{max}}{R} + \alpha^{1,l} + \sum_{i=1}^{K-1} \alpha^i \end{aligned}$$

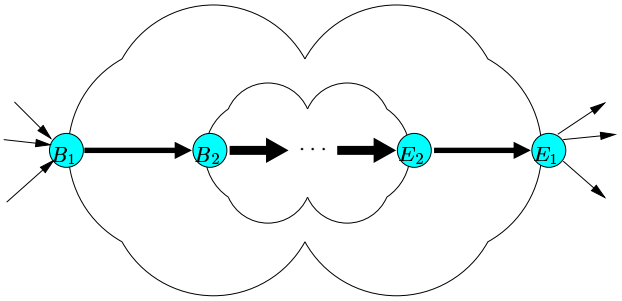


Fig. 10. Multiple Recursive Aggregation

Thus, from the definition of GR server, for flow f, the first (K-1) servers can be viewed as a virtual GR server with scheduling constant $\alpha^* = (K-2)\frac{\ell_A^{max}}{R} + \alpha^{1,l} + \sum_{i=1}^{K-1}\alpha^i$.

Then for the last hop, since S_K is also a GR server for flow f, we have

$$GRC^K(p_f^j) \le GRC^{1,l}(p_f^j) + \alpha^* + \frac{\ell_f^{max}}{r_f}.$$

Thus we obtain

$$\begin{aligned} D^K(p_f^j) &\le GRC^K(p_f^j) + \alpha^K \\ &\le GRC^{1,l}(p_f^j) + (K-2)\frac{\ell_A^{max}}{R} + \alpha^{1,l} + \frac{\ell_f^{max}}{r_f} + \sum_{i=1}^{K}\alpha^i \\ \Longrightarrow d_f^j &\le D^K(p_f^j) - A^1(p_f^j) \\ &\le [GRC^{1,l}(p_f^j) - A^1(p_f^j)] + (K-2)\frac{\ell_A^{max}}{R} \\ &\quad + \alpha^{1,l} + \frac{\ell_f^{max}}{r_f} + \sum_{i=1}^{K}\alpha^i. \end{aligned}$$

■

APPENDIX IV
PROOF OF THEOREM 8

We first prove two lemmas, which consider two basic cases of multiple aggregations. One is pure recursive aggregation, as shown in Fig. 10, and the other is pure sequential aggregation as shown in Fig. 11.

Lemma 5 (Recursive Aggregation): Suppose flow f takes K hops of GR servers, and there are M aggregators along the path. Suppose all the aggregators are rate-controlled fair aggregators. For each aggregator B_i, there is a corresponding deaggregator E_i at a later hop, $1 \le i \le M$. R_i is the GR for aggregate flow A_i which starts from B_i and ends at E_i. If $1 \le B_1 < B_2 < \cdots B_{N-1} < B_N < E_N < E_{N-1} \cdots < E_2 < E_1 \le K$ (i.e., aggregation is done recursively, as shown in Fig. 10), the e2e delay for packet p_f^j, d_f^j, satisfies:

$$\begin{aligned} d_f^j \le\ & [GRC^1(p_f^j) - A^1(p_f^j)] \\ & + [K-1-(E_1-B_1-1)]\frac{\ell_f^{max}}{r_f} \\ & + \sum_{i=1}^{M-1}[(E_i-B_i-1)-(E_{i+1}-B_{i+1}-1)]\frac{\ell_{A_i}^{max}}{R_i} \\ & + (E_M-B_M-1)\frac{\ell_{A_M}^{max}}{R_M} + \sum_{i=1}^{M}\frac{\ell_{A_i}^{max}}{R_i} + \sum_{n=1}^{K}\alpha^n. \end{aligned} \quad (46)$$

Proof: First, consider aggregate flow A_M. From Eq. (34) we know

$$\begin{aligned} D^{E_M}(p^j_{A_{M-1}}) \le\ & GRC^{B_M}(p^j_{A_{M-1}}) + (E_M - B_M)\frac{\ell_{A_M}^{max}}{R_M} \\ & + \frac{\ell_{A_{M-1}}^{max}}{R_{M-1}} + \sum_{n=B_M}^{E_M}\alpha^n. \end{aligned}$$

Thus, for aggregate flow A_{M-1}, the set of nodes from B_M to E_M can be viewed as a virtual GR server with scheduling constant $\alpha^{B_M} = (E_M - B_M)\frac{\ell_{A_M}^{max}}{R_M} + \frac{\ell_{A_{M-1}}^{max}}{R_{M-1}} + \sum_{n=B_M}^{E_M}\alpha^n$.

Then, recursively for aggregate flow A_i ($M-1 \ge i \ge 1$), we have

$$\begin{aligned} & D^{E_i}(p^j_{A_{i-1}}) \\ \le\ & GRC^{B_i}(p^j_{A_{i-1}}) + [E_i - B_i - (E_{i+1}-B_{i+1})]\frac{\ell_{A_i}^{max}}{R_i} \\ & + \frac{\ell_{A_{i-1}}^{max}}{R_{i-1}} + \sum_{n=B_i}^{B_{i+1}-1}\alpha^n + \sum_{n=E_{i+1}+1}^{E_i}\alpha^n + \alpha^{B_{i+1}}. \end{aligned}$$

Thus for aggregate flow A_{i-1} ($M-1 \ge i \ge 1$), the set of nodes from B_i to E_i can be viewed as a virtual GR server with scheduling constant

$$\begin{aligned} \alpha^{B_i} =\ & [E_i - B_i - (E_{i+1}-B_{i+1})]\frac{\ell_{A_i}^{max}}{R_i} + \frac{\ell_{A_{i-1}}^{max}}{R_{i-1}} \\ & + \sum_{n=B_i}^{B_{i+1}-1}\alpha^n + \sum_{n=E_{i+1}+1}^{E_i}\alpha^n + \alpha^{B_{i+1}}. \end{aligned}$$

Here we define $A_0 = f$ and $R_0 = r_f$.

Since S_1 and S_K are GR servers for flow f, from Theorem 3, we have

$$\begin{aligned} D^K(p_f^j) \le\ & GRC^1(p_f^j) + [K-1-(E_1-B_1)]\frac{\ell_f^{max}}{r_f} \\ & + \sum_{n=1}^{B_1-1}\alpha^n + \sum_{n=E_1+1}^{K}\alpha^n + \alpha^{B_1}. \end{aligned}$$

Thus, by replacing $\alpha^{B_i}(1 \le i \le M)$ recursively, we obtain

$$\begin{aligned} D^K(p_f^j) \le\ & GRC^1(p_f^j) + [K-1-(E_1-B_1)]\frac{\ell_f^{max}}{r_f} \\ & + \sum_{n=1}^{B_1-1}\alpha^n + \sum_{n=E_1+1}^{K}\alpha^n \\ & + [E_1 - B_1 - (E_2-B_2)]\frac{\ell_{A_1}^{max}}{R_1} + \frac{\ell_f^{max}}{r_f} \\ & + \sum_{n=B_1}^{B_2-1}\alpha^n + \sum_{n=E_2+1}^{E_1}\alpha^n \\ & \vdots \\ & + [E_{M-1} - B_{M-1} - (E_M - B_M)]\frac{\ell_{A_{M-1}}^{max}}{R_{M-1}} \\ & + \frac{\ell_{A_{M-2}}^{max}}{R_{M-2}} + \sum_{n=B_{M-1}}^{B_M-1}\alpha^n + \sum_{n=E_M+1}^{E_{M-1}}\alpha^n \\ & + [E_M - B_M]\frac{\ell_{A_M}^{max}}{R_M} + \frac{\ell_{A_{M-1}}^{max}}{R_{M-1}} + \sum_{n=B_M}^{E_M}\alpha^n. \end{aligned}$$

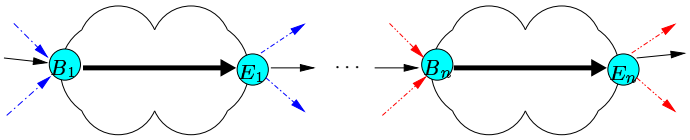


Fig. 11. Multiple Sequential Aggregation

Rearranging the terms, we obtain

$$\begin{aligned} D^K(p_f^j) \;\le\; & GRC^1(p_f^j) + [K-1-(E_1-B_1-1)]\frac{\ell_f^{max}}{r_f} \\ & + \sum_{i=1}^{M-1}[(E_i-B_i-1)-(E_{i+1}-B_{i+1}-1)]\frac{\ell_{A_i}^{max}}{R_i} \\ & + [E_M-B_M-1]\frac{\ell_{A_M}^{max}}{R_M} + \sum_{i=1}^{M}\frac{\ell_{A_i}^{max}}{R_i} + \sum_{n=1}^{K}\alpha^n \\ \Longrightarrow d_f^j \;\le\; & [GRC^1(p_f^j) - A^1(p_f^j)] \\ & + [K-1-(E_1-B_1-1)]\frac{\ell_f^{max}}{r_f} \\ & + \sum_{i=1}^{M-1}[(E_i-B_i-1)-(E_{i+1}-B_{i+1}-1)]\frac{\ell_{A_i}^{max}}{R_i} \\ & + (E_M-B_M-1)\frac{\ell_{A_M}^{max}}{R_M} + \sum_{i=1}^{M}\frac{\ell_{A_i}^{max}}{R_i} + \sum_{n=1}^{K}\alpha^n. \end{aligned}$$

■

Since B_i and E_i are, respectively, the aggregator and deaggregator for aggregate flow A_i, they schedule packets of aggregate flow A_{i-1}. Thus, the number of hops/servers that schedule packets of A_i is $(E_i-B_i-1)-(E_{i+1}-B_{i+1}-1)$ for $1 \le i < M$, or E_i-B_i-1 for $i=M$. Thus, Eq. (46) can be rewritten as Eq. (36).

Lemma 6 (Sequential Aggregation): Suppose flow f takes K hops of GR servers, and there are M aggregators along the path. Suppose all the aggregators are rate-controlled fair aggregators. For each aggregator B_i, there is a corresponding deaggregator E_i at a later hop, $1 \le i \le M$. R_i is the GR for aggregate A_i, which starts from B_i and ends at E_i. If $1 \le B_1 < E_1 \le B_2 < E_2 \le \cdots \le B_M < E_M \le K$ (i.e., aggregation is done sequentially, as shown in Fig. 11), then the e2e delay bound for packet p_f^j is:

$$\begin{aligned} d_f^j \;\le\; & [GRC^1(p_f^j) - A^1(p_f^j)] \\ & + [(K-1) - \sum_{i=1}^{M}(E_i-B_i-1)]\frac{\ell_f^{max}}{r_f} \\ & + \sum_{i=1}^{M}(E_i-B_i-1)\frac{\ell_{A_i}^{max}}{R_i} + \sum_{i=1}^{M}\frac{\ell_{A_i}^{max}}{R_i} + \sum_{n=1}^{K}\alpha^n. \end{aligned} \quad (47)$$

Proof: For each aggregate flow A_i $(1 \le i \le M)$, from Eq. (34), we have

$$\begin{aligned} & D^{E_i}(p_f^j) \\ \le\; & GRC^{B_i}(p_{A_i}^j) + (E_i-B_i)\frac{\ell_{A_i}^{max}}{R_i} + \frac{\ell_f^{max}}{r_f} + \sum_{n=B_i}^{E_i}\alpha^n \end{aligned}$$

Thus for flow f, the set of nodes from B_i to E_i can be view as a virtual GR server with scheduling constant $\alpha^{B_i} = (E_i - B_i)\frac{\ell_{A_i}^{max}}{R_i} + \frac{\ell_f^{max}}{r_f} + \sum_{n=B_i}^{E_i}\alpha^n$.

From Theorem 3, we obtain

$$\begin{aligned} D^K(p_f^j) \;\le\; & GRC^1(p_f^j) + [K-1-\sum_{i=1}^{M}(E_i-B_i)]\frac{\ell_f^{max}}{r_f} \\ & + \sum_{n=1}^{B_1-1}\alpha^n + \sum_{n=E_M+1}^{K}\alpha^n + \sum_{j=1}^{M-1}\sum_{n=E_j+1}^{B_{j+1}-1}\alpha^n + \sum_{i=1}^{M}\alpha^{B_i} \\ = \; & GRC^1(p_f^j) + [K-1-\sum_{i=1}^{M}(E_i-B_i)]\frac{\ell_f^{max}}{r_f} + \sum_{n=1}^{K}\alpha^n \\ & + \sum_{i=1}^{M}(E_i-B_i)\frac{\ell_{A_i}^{max}}{R_i} + \sum_{i=1}^{M}\frac{\ell_f^{max}}{r_f} \\ = \; & GRC^1(p_f^j) + [(K-1) - \sum_{i=1}^{M}(E_i-B_i-1)]\frac{\ell_f^{max}}{r_f} \\ & + \sum_{i=1}^{M}(E_i-B_i-1)\frac{\ell_{A_i}^{max}}{R_i} + \sum_{i=1}^{M}\frac{\ell_{A_i}^{max}}{R_i} + \sum_{n=1}^{K}\alpha^n. \end{aligned}$$

■

Since B_i and E_i are, respectively, the aggregator and deaggregator for aggregate flow A_i, they schedule packets of flow f. Thus, the number of hops that schedule packets of A_i is $E_i - B_i - 1$ $(1 \le i \le M)$. Thus, Eq. (47) can be rewritten as Eq. (36).

With these two lemmas, we can now prove Theorem 8.

Proof: The idea is to prove by induction on the levels of aggregation. Lemmas 5 and 6 have proved the basic cases for the theorem. We use them as two basic building blocks. We only need to show that if all the component regions of the e2e path satisfy the theorem, the whole aggregation region built based on these component regions satisfies the theorem as well. Again, we consider two cases:

Case 1: Recursive aggregation

Suppose flow f takes K hops of GR servers. Along the path, there is an aggregation region from B_1 to E_1 for aggregate flow A_1 with guaranteed rate R_1. This aggregation region contains another one from B_2 and E_2 for aggregate flow A_2, with guaranteed rate R_2. In other words, $1 \le B_1 < B_2 < E_2 < E_1 \le K$. If the region from B_2 to E_2 satisfies

$$\begin{aligned} D^{E_2}(p_{A_1}^j) \;\le\; & GRC^{B_2}(p_{A_1}^j) + \sum_{k=B_2+1}^{E_2}\frac{\ell_{\hat{A}_k}^{max}}{\hat{R}_k} \\ & + \sum_{k=2}^{M}\frac{\ell_{A_k}^{max}}{R_k} + \sum_{n=B_2}^{E_2}\alpha^n, \end{aligned}$$

then the e2e delay d_f^j satisfies

$$\begin{aligned} d_f^j \;\le\; & [GRC^1(p_f^j) - A^1(p_f^j)] + \sum_{i=2}^{K}\frac{\ell_{\hat{A}_i}^{max}}{\hat{R}_i} \\ & + \sum_{i=1}^{M}\frac{\ell_{A_i}^{max}}{R_i} + \sum_{n=1}^{K}\alpha^n. \end{aligned}$$

Case 2: Sequential aggregation

Suppose flow f takes K hops of GR servers and there are M aggregation regions along the path, each between B_i and E_i $(1 \le i \le M)$. The guaranteed rate for aggregate flow A_i is R_i, which starts from router B_i and ends at router E_i. If $1 \le B_1 < E_1 \le B_2 < E_2 \le \cdots \le B_M < E_M \le K$

(i.e., the aggregation regions are sequentially placed), and each aggregation region between B_i and E_i satisfies

$$D^{E_i}(p_f^j) \leq GRC^{B_i}(p_f^j) + \sum_{k=B_i+1}^{E_i} \frac{\ell_{\hat{A}_k}^{max}}{\hat{R}_k} + \sum_{k=M_i}^{N_i} \frac{\ell_{A_k}^{max}}{R_k} + \sum_{n=B_i}^{E_i} \alpha^n,$$

where M_i and N_i are the levels of aggregation in the i^{th} aggregation region, then the e2e delay d_f^j satisfies

$$d_f^j \leq [GRC^1(p_f^j) - A^1(p_f^j)] + \sum_{i=2}^{K} \frac{\ell_{\hat{A}_i}^{max}}{\hat{R}_i} + \sum_{i=1}^{M} \sum_{k=M_i}^{N_i} \frac{\ell_{A_k}^{max}}{R_k} + \sum_{n=1}^{K} \alpha^n.$$

The proof of the two cases are similar to those of Lemmas 5 and 6, respectively. Note that the aggregator and deaggregator for aggregate flow A_i do not schedule packets on A_i. They schedule packets of the constituent flows of A_i instead. With these two cases, the theorem is proven, since any kind of multiple aggregations is built by combining the recursive and sequential aggregations. ■