

Work-Conserving Fair-Aggregation with Rate-Independent Delay

Jorge A. Cobb

Department of Computer Science
The University of Texas at Dallas
Richardson, TX 75083-0688
Email: {cobb}@utdallas.edu

Abstract—Flow aggregation has been proposed as a technique to improve the scalability of QoS scheduling in the core of the Internet, by reducing the amount of per-flow state necessary at a router. This is accomplished by aggregating into a single flow multiple flows that share part of the same path to the destination. In its simplest form, flow aggregation is inherently non-work-conserving, that is, it may temporarily leave the output channel idle even though there are still packets to be forwarded.

Work-conserving variations of flow aggregation are more desirable, because they allow flows to temporarily exceed their reserved rate and take advantage of bandwidth unused by other flows. This, however, requires a per-hop delay proportional to the rate reserved for the aggregate flow. Thus, if a lower delay is desired, the aggregate flow must reserve from the network a data rate that is greater than its true rate.

In this paper, we explore the conditions under which work-conserving flow aggregation can be performed with rate-independent delay. That is, the aggregate flow is able to choose a per-hop delay that is independent of the data rate it reserved from the network. The price to be paid for this flexibility, however, is a restriction in the choice of packet size and data rates of its constituent flows.

I. INTRODUCTION

Currently, the Internet only provides best-effort service to its datagram traffic. This is sufficient for traditional applications, such as file transfer, email, web browsing, etc., that do not require real-time Quality of Service (QoS) guarantees. On the other hand, QoS guarantees are required for emerging applications such as audio and video conferencing, video on demand, IP telephony, etc.. In order to provide QoS to these new applications, the network must reserve resources, such as data rate and buffer space, for each individual flow.

To provide QoS, each network link implements a real-time scheduling algorithm such as Virtual Clock (VC) [1], [2] and Weighted Fair Queuing (WFQ) [3]. Zhang provided an excellent review of such schedulers in [4]. Schedulers such as VC and WFQ provide a per-hop delay that is inversely related to the data rate reserved for the flow. This is known as *rate-dependent delay*. Thus, if a flow requires an even smaller delay, the rate reserved for the flow will be larger than necessary. On the other hand, scheduling protocols such as EDD [5] and Real-Time-Channels [6] provide a per-hop delay that is chosen by the individual flow, and thus independent of the rate reserved for the flow. Thus, we refer to this approach as *rate-independent delay*.

A real-time scheduling protocol needs to maintain information about each flow that traverses its link in order to provide the required QoS to each flow. While the number of flows is manageable in an access network, routers in the core network do not have the necessary resources to manage each individual flow. This limits the scalability of these protocols, and has led to the introduction of flow aggregation [7] and dynamic packet state [8], [9] as means to reduce the per-flow state required at each router.

In flow aggregation, multiple flows that share the same path are aggregated together into a single aggregate flow. Routers after the point of aggregation are aware only of the aggregate flow and not of the individual flows that constitute it. This reduces the amount of per-flow state required in the core routers, provided flow aggregation is performed in the access networks. On the other hand, in dynamic packet state, access networks add to each packet header enough information so that the packet's deadline may be computed at each hop. Thus, no per-flow state of any kind is required at the core routers.

Although flow aggregation requires more state to be maintained at the core routers, it has its advantages over dynamic packet state. If rate-dependent delay is used, then flow aggregation provides a lower per-hop delay, since the reserved rate of the aggregate flow is much larger than that of the individual flows. If rate-independent delay is used, we have shown in [10] that flow aggregation allows a greater number of flows to satisfy the admission-control test required for rate-independent delay. Thus, in this paper, we focus on flow aggregation.

In its simplest form, flow aggregation is inherently non-work-conserving, that is, it may temporarily leave the output channel idle even though there are still packets to be forwarded. Work-conservation is more desirable, because it allows flows to temporarily exceed their reserved bandwidth and take advantage of bandwidth unused by other flows.

Recently, Work-Conserving Flow Aggregation (WCFA) has been proposed [11]. However, it is based on rate-dependent delay. In this paper, we explore the conditions under which work-conserving flow aggregation can be performed with rate-independent delay. That is, the aggregate flow will be able to choose a per-hop delay that is independent of the data rate it reserved from the network. The price to be paid for this flexibility in delay, however, is a restriction in the choice of packet size and data rates of its constituent flows.

The rest of this paper is organized as follows. We begin in Section II, where we present our QoS model. We then continue in Section III with an overview of the aggregation of multiple flows into a single flow. We show how flow aggregation was originally designed to be non-work-conserving, and then present recently-proposed work-conserving extensions. In Section IV, we present the obstacles to our objective: providing rate-independent delay in work-conserving flow aggregation. In Section V we show how these obstacles can be avoided by restricting the choices of packet size and reserved rate of the constituent flows. Finally, in Section VI, we present the scheduling protocol that must be implemented at each router. Future work is discussed in Section VII.

II. QUALITY OF SERVICE MODEL

Our network model is based on the models of [12] and [13]. A *network* is a set of computers connected via point-to-point communication channels. A *flow* is a sequence of packets, all of which originate at the same source computer and are destined to the same computer. All packets of a flow must traverse the network along a fixed path.

Each flow is characterized by its data rate and an upper bound on its end-to-end delay. Before a source introduces a new flow to the network, enough network resources are reserved to ensure the flow's delay bound is not violated. If enough resources are not available, the flow is rejected.

Each output channel of a computer is equipped with a scheduler. A scheduler receives packets from flows whose path traverses its output channel. Whenever its output channel becomes idle, the scheduler chooses a received packet and forwards it over the output channel.

A packet *exits* a scheduler when the last bit of the packet is transmitted by its output channel. We adopt the following notation for each flow f and each scheduler s along its path.

R_f	data rate reserved for flow f .
$p_{f,i}$	i^{th} packet of f , $i \geq 1$.
$L_{f,i}$	length of packet $p_{f,i}$.
L_f^{max}	maximum packet size of f .
L_s^{max}	maximum packet length at s .
$A_{s,f,i}$	arrival time of $p_{f,i}$ at scheduler s .
$E_{s,f,i}$	exit time of $p_{f,i}$ from s .
C_s	bandwidth of the output channel of s .
π_s	propagation delay of channel s .

We define the *virtual start-time* $S_{s,f,i}$ of packet $p_{f,i}$ at scheduler s as follows [12], [13]. Assume s were to forward the packets of f at *exactly* R_f bits/sec.. Then, $S_{s,f,i}$ is the time at which the first bit of $p_{f,i}$ is forwarded by s . Similarly, the *virtual finishing time* $F_{s,f,i}$ is the time at which the last bit of $p_{f,i}$ is forwarded by s .

More formally, let f be an input flow of scheduler s . Then,

$$\begin{aligned} S_{s,f,1} &= A_{s,f,1} \\ F_{s,f,1} &= S_{s,f,1} + \frac{L_{f,1}}{R_f} \\ S_{s,f,i} &= \max(A_{s,f,i}, F_{s,f,i-1}) \quad i > 1 \\ F_{s,f,i} &= S_{s,f,i} + \frac{L_{s,f,i}}{R_f} \quad i > 1 \end{aligned}$$

Assume that scheduler s forwards the packets of an input flow f at a rate of at least R_f . Then, each packet $p_{f,i}$ exits scheduler s not much later than its start time, $S_{s,f,i}$. We refer to these schedulers as *rate-guaranteed schedulers* [12], [13], [14]. More formally, a scheduler s is a rate-guaranteed scheduler iff, for every input flow f of s and every i , $i \geq 1$,

$$E_{s,f,i} \leq S_{s,f,i} + \delta_{s,f,i}$$

for some constant $\delta_{s,f,i}$. We refer to $\delta_{s,f,i}$ as the *delay* of packet $p_{f,i}$ at scheduler s , and we refer to $S_{s,f,i} + \delta_{s,f,i}$ as the *deadline* of $p_{f,i}$ at s .

For example, by choosing $\delta_{s,f,i} = L_f^{\text{max}}/R_f + L_s^{\text{max}}/C_s$, $\delta_{s,f,i}$ becomes the *rate-dependent delay* of the Virtual Clock [1], [2] and Weighted Fair Queuing [3] protocols, since the delay is related to the rate of the flow¹. Otherwise, if $\delta_{s,f,i}$ is chosen by flow f to be unrelated to R_f , then we say the delay is *rate-independent*.

We assume all schedulers are rate-guaranteed schedulers that provide rate-independent delay. Also, as commonly done in the literature [12], [13], [6], we assume the delay is fixed for all packets of the same flow, i.e., each flow f has a delay constant $\delta_{s,f}$ at scheduler s .

The delay of a packet across a sequence of rate-guaranteed schedulers is well known [12], [13], [14]. Since the start-time of a packet determines its exit time from a scheduler, then a bounded end-to-end delay requires a bounded per-hop increase in the start-time of the packet. This bound is as follows.

Let t_1, t_2, \dots, t_k be a sequence of k rate-guaranteed schedulers traversed by flow f . For all i ,

$$\begin{aligned} S_{t_k,f,i} &\leq S_{t_1,f,i} + \sum_{x=1}^{k-1} \delta_{t_x,f} + \sum_{x=1}^{k-1} \pi_{t_x} \\ E_{t_k,f,i} &\leq S_{t_k,f,i} + \delta_{t_k,f} \end{aligned} \quad (1)$$

To ensure packets exit by their deadline, a scheduling test must be performed for each scheduler to determine if enough resources are available. For rate-dependent delay, the test is simple: the sum of the rates of all flows must be at most the capacity of the output channel. For rate-independent delay, however, scheduling tests are somewhat more involved due to the flexibility in choosing deadlines. The interested reader is referred to [6] for optimum scheduling tests with rate-independent delay.

¹Note that the last term, L_s^{max}/C_s is very small, it is just the transmission time of a single packet.

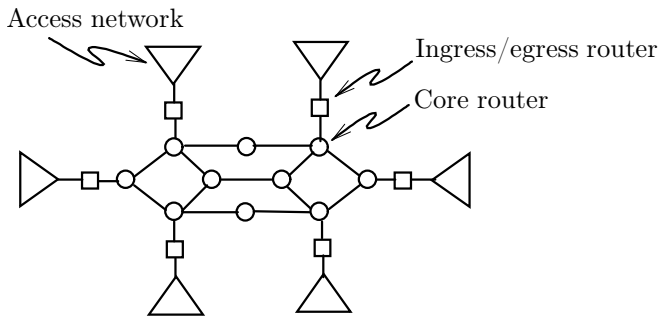


Fig. 1. Core Network

III. OVERVIEW OF FLOW AGGREGATION

A. Non-Work-Conserving Flow Aggregation

To reduce the amount of state managed by each router, multiple flows can be combined together to form a single aggregate flow [7], [10], [15], [16].

An *aggregate* flow g is obtained by merging, at a single point in the network, the packets of multiple flows f^1, f^2, \dots, f^n . In this case, f^1, f^2, \dots, f^n are said to be the *constituents* of g . The reserved rate, R_g , of aggregate flow g is at least the sum of the reserved rates of its constituent flows. A scheduler that receives as inputs a set of flows f^1, f^2, \dots, f^n , and produces as output a single aggregate flow g is called an *aggregator*.

Schedulers after the aggregation point are not aware of the constituents of an aggregate flow. At a later point in the network, the aggregate flow is separated again into its constituent flows. A *separator* is a process that receives as input an aggregate flow, and produces as output the set of *constituents* of the input flow. We assume a separator causes no packet delay.

We consider aggregation over a core network model, shown in Figure 1. It consists of a network of core routers surrounded by access networks².

Ingress routers in the access networks manage the input of flows into the core network. To reduce the amount of state, they aggregate together flows that have the same exit point from the core network. Egress routers in the access networks receive the aggregate flows that crossed the core network and separate each of these into their constituent flows, before the flows exit into the access networks.

Flow aggregation is advantageous regardless of whether rate-dependent delay or rate-independent delay is chosen. For rate-dependent delay, the per-hop delay of an aggregate flow g at a scheduler t would be

$$\frac{L_g^{max}}{R_g} + \frac{L_t^{max}}{C_t} \quad (2)$$

as opposed to the delay of a simple flow f without aggregation which is

$$\frac{L_f^{max}}{R_f} + \frac{L_t^{max}}{C_t}$$

²This network is similar to a SCORE network in [8], [9].

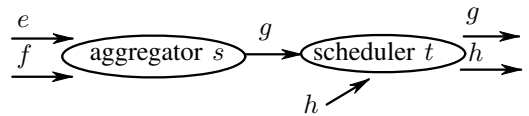


Fig. 2. Need for fair aggregation

Note that in general $R_f \ll R_g$ and $L_f^{max} \approx L_g^{max}$, and thus the delay is much smaller with flow aggregation. If rate-independent delay is used, we have shown earlier in [10] that a larger number of flows can be supported in the network and still satisfy the (per-node) optimal scheduling tests of [17]. Thus, a greater network utilization is possible.

B. Work-Conserving Flow Aggregation

Even if an aggregator is a rate-guaranteed scheduler, it is not sufficient to ensure a bounded end-to-end delay to its input flows. E.g., consider Figure 2, where two flows, e and f , are input to an aggregating scheduler s , whose aggregate output is g , and the next scheduler after s is t . Assume e generates packets at a rate greater than R_g , i.e., greater than $R_e + R_f$, f is generating few packets, if any, and aggregator s does not delay packets. Since scheduler t forwards the packets of g at a rate of R_g , the queue of g may grow arbitrarily large at t . Therefore, the next packet of f encounters a large number of packets ahead of it in the queue of g at t , and suffers an excessive delay.

Flow aggregation was originally non-work-conserving [7] as follows. Aggregators were chosen to be rate-guaranteed schedulers whose output was always restricted to the rate R_g of the aggregate flow. That is, after forwarding a packet of L bytes, the next packet could not be forwarded until L/R_g seconds later, even though the capacity of the output channel of the aggregator is significantly greater than R_g . Note that this prevents the queue buildup at scheduler t in Figure 2.

The first approach to introduce work-conservation into flow aggregation was presented by Sun and Shin [18]. Consider again Figure 2. If there is a large queue at g when packet $f.i$ arrives, the delay of $f.i$ could be kept small if the queue of g were not served in FIFO order. That is, if $f.i$ could be served first before the other packets of flow e .

To implement the above, aggregator s assigns a tag $T_{f,i}$ to each input packet $p_{f,i}$. The tag is chosen to be equal to the *virtual finishing time of the packet* at s , i.e., $T_{f,i} = F_{f,i}^s$.

Scheduler t , and each subsequent scheduler in the path of g , sorts the queue of g by tag value. Note, however, that these schedulers, although they are aware of flow g , they are not aware of g 's constituent flow f .

Although work-conserving, [18] required flows to be filtered via leaky-buckets, and the end-to-end delay depended on the leaky-bucket parameters. Thus, flows were unable to take advantage of unused bandwidth, which is the principal objective of a work-conserving system.

In [11] we presented a work-conserving aggregation method, based on rate-dependent delay, whose end-to-end

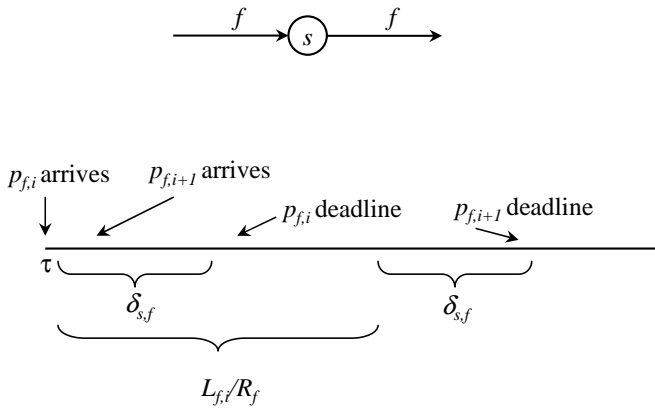


Fig. 3. Separation of packet deadlines

delay is similar to Relation (1), except that the per-hop delay of aggregate flow g at scheduler t is

$$2 \cdot \frac{L_g^{max}}{R_g} + \frac{L_t^{max}}{C_t} \quad (3)$$

and is thus independent of the leaky-bucket parameters of other flows. The same assignment of tags to packets as above is used, however, schedulers are restricted to have a high-degree of fairness, such as W²FQ [19] and the whole family of schedulers defined in [20]. Note that work-conserving aggregation has the price of a delay twice that of non-work conserving aggregation (compare (2) and (3) above).

IV. OBSTACLES FOR RATE-INDEPENDENT DELAY IN WORK-CONSERVING FLOW AGGREGATION

We next consider the obstacles to be overcome in adding rate-independent delay to work-conserving flow aggregation. Consider a single scheduler s with several input flows, one of which is f . Scheduler s has to provide rate-independent delay to each input flow (see Figure 3). Assume that at time τ , a packet $p_{f,i}$ arrives, and $S_{s,f,i} = \tau$. According to the definition of a rate-guaranteed scheduler, the packet will exit by time

$$S_{s,f,i} + \delta_{s,f}$$

Assume, however, that packet $p_{f,i+1}$ arrives immediately after $p_{f,i}$. In this case, by definition, the deadline of the packet will be $S_{s,f,i+1} + \delta_{s,f}$, that is,

$$S_{s,f,i} + \frac{L_{f,i}}{R_f} + \delta_{s,f} \quad (4)$$

This is proper behavior because the deadlines of packets from the same flow are separated in general by at least L_f/R_f .

Assume instead that s has an aggregate flow g as input, g has multiple constituent flows f_1, \dots, f_n , and packets are tagged by an aggregator as discussed earlier. Scheduler s has to sort the packets of g before forwarding them.

Note that the desired per-hop delay of each constituent flow f_x should be $\delta_{s,g}$. Consider, however, the following scenario. A packet from flow f_x arrives at time $\tau + \epsilon$, where ϵ is small,

and a packet from flow f_y arrives at time τ . Assume the tag of f_y is much greater than that of f_x (which is possible because aggregators don't delay packets in any way, they just tag them).

In this case, scheduler s is unaware of the difference between f_x and f_y , and it might simply send out the first packet of g (i.e. of f_y) at time τ , because it assumes it has a deadline of $S_{s,g} + \delta_{s,g} = \tau + \delta_{s,g}$. Then, similar to the above case in (4), s will be busy forwarding packets of other flows other than g , and may not consider scheduling the next packet of g (i.e., f_x) until time $\tau + L_{f_y}/R_g$. Thus, the exit time of the packet from f_x would be at most

$$\tau + L_{f_y}/R_g + \delta_{s,g}$$

Note that this is dependent on the rate of g , and is thus undesirable. The reason for this problem is that packets cannot be preempted; once the packet from f_y is chosen for transmission, it cannot be stopped. This forces the next packet of f_x to wait an additional L_{f_y}/R_g seconds before being considered for transmission.

Furthermore, a similar scenario could occur at every hop along the path of g . And hence, the per-hop delay of g would be

$$L_g/R_g + \delta_{s,g}$$

as opposed to simply $\delta_{s,g}$.

It is also interesting to note that, for rate-dependent delay, where $\delta_{s,g} \approx L_g/R_g$, a similar scenario happens, which is why we mentioned above that work-conserving flow aggregation has a per-hop delay that is twice that of non-work-conserving aggregation.

V. ELIMINATING THE NEED FOR PACKET PREEMPTION

As discussed above, the reason that the delay increases in work-conserving flow aggregation is that packets cannot be preempted once they begin to be forwarded by the scheduler. However, we show that if we can restrict the rates and packet sizes of the constituent flows, then the need for packet preemption is eliminated. We begin with some fictitious scheduler s , and see how the need for preemption can be eliminated. In the next section, we describe how this would apply to our aggregators at the ingress routers and at the schedulers along the core routers.

Consider a scheduler s that receives multiple input flows f_1, \dots, f_n . Assume the output channel capacity of the scheduler is R_g bits/second. Assume s forwards the packets in order of their virtual finishing times $F_{s,f,i}$ ³. Then, it is well known that each packet $p_{f,i}$ will exit by time

$$F_{s,f,i} + \frac{L_s^{max}}{R_g} \quad (5)$$

The term $\frac{L_s^{max}}{R_g}$ arises because of the lack of packet preemption.

Assume instead that all packets of g have the same length, L_g , and that the virtual starting time $S_{s,f,i}$ of each packet

³This, in a sense, is what the virtual-clock protocol performs.

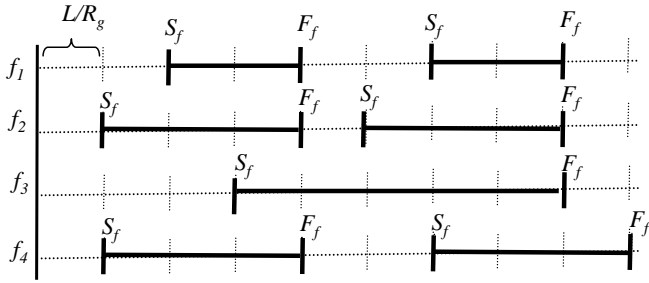


Fig. 4. Restrictions on virtual start times.

$p_{f,i}$ is a multiple of L_g/R_g . That is, for each packet $p_{f,i}$, $S_{s,f,i} = k \cdot L_g/R_g$ for some integer k . This is shown in Figure 4. Note that scheduler s takes L_g/R_g seconds to transmit a packet of length L_g .

From Figure 4, we can think of time as being slotted, with slots of width L_g/R_g . Assume then that scheduler s only forwards a packet to the output channel at the beginning of each slot. Then, it can be shown that each packet $p_{f,i}$ exits no later than time $F_{s,f,i}$.

The intuitive reason behind the removal of the L_g/R_g term from the exit bound in (5) is the following. Because time is slotted, it is not possible for a packet with larger virtual finishing time to begin transmission a brief moment before the virtual starting time of a packet with smaller virtual finishing time. By the time the next slot begins, both of these packets will be available at the scheduler, who will then choose the one with smallest virtual finishing time.

We can actually relax somewhat when the scheduler can transmit a packet. This is given by the theorem below.

Theorem 1: Consider a scheduler s that receives multiple input flows f_1, \dots, f_n and whose output channel capacity is at least R_g bits/second. All packets have a constant size L_g . Assume that for every packet $p_{f_x,i}$, there are two values defined, $\tilde{S}_{s,f_x,i}$ and $\tilde{F}_{s,f_x,i}$, such that

- $p_{f_x,i}$ arrives no later than time $\tilde{S}_{s,f_x,i}$.
- $\tilde{S}_{s,f_x,i} = k \cdot L_g/R_g$ for some integer k .
- $\tilde{S}_{s,f_x,i} + L_g/R_g = \tilde{F}_{s,f_x,i} \leq \tilde{S}_{s,f_x,i+1}$.

Assume s forwards packets in order of $\tilde{F}_{s,f_x,i}$, and it may not be work-conserving. However, it has the restriction that, at the beginning of every timeslot, it must forward a packet. Then, every packet $p_{f,i}$ exits s no later than time $\tilde{F}_{s,f,i}$. ■

VI. WORK-CONSERVING RID SCHEDULERS AND AGGREGATORS

The result from the previous section lays the foundation for how schedulers in the core network are to behave. In a sense, because the aggregators and schedulers are work-conserving, the scheduling and sorting of packets that occurs in Theorem 1 occurs in a distributed manner at each hop along the path of the flow. The restricted choice of packet size and \tilde{S} values ensure that the packet-preemption problem does not occur.

A. RID Aggregators

Flow aggregators for rate-independent delay (RID) at an ingress router will be similar to those described in Section III-B. We assume they are internal to the ingress router, and hence, their output rate is essentially infinity, and introduce no packet delay. Their aggregate flows are then given as input to an RID scheduler which then forwards them over an output channel to a core router. Aggregators attach a pair of labels to each packet before forwarding it. The labels of packet $p_{f,i}$ at aggregator s are $\tilde{S}_{s,f,i}$ and $\tilde{F}_{s,f,i}$. We assume the restrictions on these values as given in Theorem 1.

Note that, when a packet $p_{f,i}$ arrives at time $A_{s,f,i}$, and there is no multiple of L_g/R_g equal to $A_{s,f,i}$, the packet need not be delayed in the aggregator; it is simply given the next value of $\tilde{S}_{s,f}$ that is a multiple of L_g/R_g and forwarded. Thus, as mentioned above, aggregators introduce no packet delay.

Note also that the restriction on $\tilde{S}_{s,f}$, in effect, forces the rate of every constituent flow f to be such that L_g/R_f is a multiple of L_g/R_g . Otherwise, it would be impossible to satisfy the rate of R_f while at the same time ensuring that $\tilde{S}_{s,f}$ is a multiple of L_g/R_g .

We assume that all flows that enter and exit the core network at the same points and that wish to have the same per-hop delay δ_g are aggregated into a single flow g . We also assume that the scheduling condition for rate-independent delay given in [6] is satisfied at each hop in the core network. Finally, because core routers are aware of the aggregate flow g (but not its constituents), we expect them to be aware of R_g and δ_g . Thus, these values need not be included in the packet.

B. RID Schedulers

Consider now a scheduler in the core of the network that receives aggregate flows as input. We do not consider a specific rate-independent-delay (RID) scheduler. Instead, we present the general properties required from it. Most RID schedulers, such as [5] and [6], can be easily modified to meet our requirements. We first briefly overview the behavior of a typical RID scheduler, then we present our required modifications.

Assume the first packet of an input flow g arrives at the scheduler at time Γ . At this point, the scheduling protocol ensures that the next packet of g is forwarded by time

$$\Gamma + \delta_g.$$

Assume the next packet of g arrives at time Γ' . If $\Gamma' > \Gamma + L_g/R_g$, then the scheduler is required to forward the packet no later than time $\Gamma' + \delta_g$. On the other hand, if $\Gamma' \leq \Gamma + L_g/R_g$, then the scheduler will forward the second packet no later than time

$$\Gamma + L_g/R_g + \delta_g.$$

Our requirements for a scheduler consist of adding a few modifications, as follows.

First, schedulers in the network core must maintain the packets of each flow sorted by \tilde{F} value. Because all packets of a flow are of the same size, they take the same amount of

time to transmit over the output channel. Thus, if the head of some flow's queue changes due to the arrival of a packet, it makes no difference to the scheduler.

Second, instead of using the true arrival time of packets, the scheduler assumes that packets of g arrive only at timeslot boundaries, i.e., every L_g/R_g seconds. If a timeslot begins at Γ , then within the interval $[\Gamma, \Gamma + \delta_g]$, the scheduler will forward the next packet of g . The next timeslot begins at time $\Gamma' = \Gamma + L_g/R_g$, and the scheduler must forward another packet of g during the interval $[\Gamma', \Gamma' + \delta_g]$, etc..

We must address how a scheduler determines the starting point of each timeslot. This is implied by the tag values in each packet, because \tilde{S} must begin at a timeslot boundary. Note that this requires clocks to be synchronized between nodes, but this is a common assumption in core networks [21][22].

The last point is that flow g is delayed $\delta_g + \pi$ seconds at every hop, where π is the propagation delay. Thus, its \tilde{S} and \tilde{F} tags need to be updated for the next core router. Before a packet of g is forwarded, these values are increased by $\delta_g + \pi$.

We say a flow g is eligible for transmission at the beginning of each slot, and becomes ineligible once its next packet is forwarded. Since we assume the RID scheduler is work-conserving, if at a moment there are no eligible flows, the scheduler can forward the next packet from the queue of any flow, presumably with some fairness among flows.

We may now present the main result.

Theorem 2: Let s be a RID aggregator with all the properties described above, f be an input flow of s , and g be the output aggregate flow of g . Let g traverse k RID schedulers, t_1, \dots, t_k each with the properties described above, and the scheduling condition of [6] is satisfied at each of the k schedulers.

Then, the exit time of packet $p_{f,i}$ from the last scheduler is

$$E_{t_k, f, i} \leq \tilde{F}_{s, f, i} + \frac{L_g}{R_g} + k \cdot \delta_g + \sum_{x=1}^{k-1} \pi_x.$$

■

The first term in Theorems 2 and Theorem 1 is the same, because the packets of g are sorted in a distributed manner as they traverse the core routers. The additional L_g/R_g term arises because when a packet from f arrives at g , the next legal value of \tilde{S}_f may be L_g/R_g seconds greater than its arrival time. Finally, the per-hop delay is as expected.

VII. FUTURE WORK

In order to take advantage of work-conservation, it is possible that constituent flows exceed their reserved rate temporarily to take advantage of bandwidth unused by other flows. In this case, the flow that exceeds its rate will have values for \tilde{S} that are much greater than real time. If the other flows which were inactive become active and transmit at their full rate or higher, it is possible that the flow that originally exceeded its rate is temporarily denied service from the network until the \tilde{S} values of the other flows "catch up" to that of the original flow. Note, however, that the bounds of Theorem 2 still hold.

To mitigate the above, a technique known as timestamp-reuse can be added [9], [23]. With this technique, if the network is not fully loaded, it is possible that a flow may reuse earlier values of \tilde{S} in an effort to prevent/mitigate the above-mentioned denial of service. This will be considered in future work.

REFERENCES

- [1] X. G. and L. S., "Delay guarantee of the virtual clock server," *IEEE/ACM Transactions on Networking*, pp. 683–689, Dec. 1995.
- [2] L. Zhang, "Virtual clock: A new traffic control algorithm for packet-switched networks," *ACM Transactions on Computer Systems*, vol. 9, no. 2, pp. 101–124, May 1991.
- [3] A. K. J. Parekh and R. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: The single node case," *IEEE/ACM Transactions on Networking*, vol. 1, no. 3, Jun. 1993.
- [4] H. Zhang, "Service disciplines for guaranteed performance service in packet-switching networks," *Proceedings of the IEEE*, vol. 93, no. 10, Oct. 1995.
- [5] D. Ferrari and D. Verma, "A scheme for real-time channel establishment in wide-area networks," *IEEE Journal on Selected Areas in Communications*, vol. 8, no. 3, 1990.
- [6] S. K. Zheng Q., "On the ability of establishing real-time channels in point-to-point packet-switched networks," *IEEE Transactions on Communications*, vol. 42, no. 2-4, 1994.
- [7] J. Cobb, "Preserving quality of service guarantees in spite of flow aggregation," *IEEE/ACM Transactions on Networking*, vol. 10, no. 1, pp. 43–53, Feb. 2002.
- [8] I. Stoica and H. Zhang, "Providing guaranteed services without per-flow management," in *Proc. of the ACM SIGCOMM Conference*, 1999.
- [9] J. Kaur and H. M. Vin, "Core-stateless guaranteed rate scheduling algorithms," in *Proc. of the IEEE INFOCOM Conf.*, 2001.
- [10] J. Cobb, "Scalable quality of service across multiple domains," *Computer Communications*, vol. 28, no. 18, pp. 1997–2008, Nov. 2005, Elsevier.
- [11] J. Cobb and Z. Xu, "Maintaining flow isolation in work-conserving flow aggregation," in *Proc. IEEE GLOBECOM Conference*, 2005.
- [12] J. Cobb and M. Gouda, "Flow theory," *IEEE/ACM Transactions on Networking*, vol. 5, no. 5, pp. 661–674, Oct. 1997.
- [13] F. N. and P. J., "Leave-in-time: A new service discipline for real-time communications in a packet-switching data network," in *Proc. of the ACM SIGCOMM Conference*, 1995.
- [14] P. Goyal, S. Lam, and H. Vin, "Determining end-to-end delay bounds in heterogeneous networks," in *Proc. of the NOSSDAV Workshop*, 1995.
- [15] H. Fu and E. W. Knightly, "A simple model of real-time flow aggregation," *IEEE/ACM Transactions on Networking*, vol. 11, no. 3, Jun. 2003.
- [16] J. Qiu and E. W. Knightly, "Measurement-based admission control with aggregate traffic envelopes," *IEEE/ACM Transactions on Networking*, vol. 9, no. 2, Apr. 2001.
- [17] Q. Zheng and K. G. Shin, "On the ability of establishing real-time channels in point-to-point packet-switched networks," *IEEE Transaction on Communications*, vol. 42, no. 3, Mar. 1994.
- [18] W. Sun and K. G. Shin, "Coordinated aggregate scheduling for improving end-to-end delay performance," in *Proc. of the IEEE Workshop on Quality of Service (IWQoS)*, 2004.
- [19] J. C. Bennett and H. Zhang, "WF2Q: worst-case fair weighted fair queueing," in *IEEE INFOCOM Conference*, 1996.
- [20] D. Stiliadis and A. Varma, "Rate proportional servers: A design methodology for fair queuing algorithms," *IEEE/ACM Transactions on Networking*, Apr. 1998.
- [21] J. Kaur and H. M. Vin, "Core stateless guaranteed throughput networks," in *Proc. of the IEEE INFOCOM Conf.*, 2003.
- [22] —, "Providing deterministic end-to-end fairness guarantees in corestateless networks," in *Proc. IEEE International Workshop on Quality of Service (IWQoS)*, 2003.
- [23] J. Cobb and Z. Xu, "Guaranteed throughput in work-conserving flow aggregation through deadline reuse," in *Proceedings of the ICCCN Conference*, Oct. 2006.