An In-Depth Look at Flow Aggregation for Efficient Quality of Service

Jorge A. Cobb
Department of Computer Science
The University of Texas at Dallas
Richardson, TX 75083-0688

Abstract

We investigate the preservation of quality of service guarantees to a flow of packets in the presence of flow aggregation. For efficiency, multiple flows, known as the constituent flows, are merged together resulting in a single aggregate flow. Packet schedulers located after the network point where the aggregation occurred are aware of the aggregate flow, but are unaware of its constituent flows. In spite of this, we show that quality of service guarantees may still be offered to the constituent flows provided the aggregation is performed fairly. In earlier work, we showed that the end-to-end delay is preserved (and in some cases improved) under flow aggregation, when the packet delay is coupled with the reserved rate of the flow. We go beyond these results by showing that, even when the delay is de-coupled from the reserved rate, the end-to-end delay is preserved under flow aggregation.

1. Introduction

Consider the problem of delivering packets from a real-time application across a computer network. This problem has been studied extensively, resulting in the development of *guaranteed-rate schedulers*. That is, scheduling protocols which guarantee a minimum bandwidth and a maximum packet delay to each application. Examples of these scheduling protocols can be found in [14][15].

In guaranteed-rate schedulers, resources, such as bandwidth and buffer space, are reserved for each application along the entire path to its destination. If not enough resources are available, the application's request for a network connection is denied. Due to the reservation of resources, the network can provide service guarantees to each application, provided the rate of the application does not exceed the agreed-upon rate. These service guarantees are of paramount importance to real-time applications, such as interactive audio and video [7].

Let us denote by *flow* the sequence of packets generated by a single application. In this paper, we investigate the effects of aggregating multiple flows, known as the constituent flows, into a single aggregate flow. Once the aggregation is done, the remaining schedulers along the path have no knowledge of the constituent flows of the aggregate flow. They simply treat the aggregate flow as a single flow whose rate is the sum of the reserved rates of the constituent flows.

One of many possible implementations of flow aggregation is virtual paths in virtual circuit networks. Multiple virtual circuits may be combined into a single virtual path. Schedulers are aware of the virtual path, but are unaware of the virtual circuits constituting the virtual path. Thus, a virtual circuit may be viewed as a constituent flow, and a virtual path may be viewed as an aggregate flow.

The purpose of flow aggregation is to improve the efficiency of the schedulers and to simplify the management of flows. For example, buffer management is simplified, because only one queue per aggregate flow is required, rather than one queue per constituent flow. Scheduling efficiency is improved, because the number of flows is reduced. Finally, rerouting an aggregate flow in the event of a failed channel along its path is much more efficient than rerouting each of the constituent flows individually.

In this paper, we examine if it is possible to provide quality of service guarantees to the constituent flows, even though schedulers are not aware of them. In particular, we consider end-to-end delay guarantees. In an earlier work [1], we investigated flow aggregation in a network where the per-hop delay of a flow is coupled with the flow's reserved rate. We showed that, if the aggregation of flows is performed fairly, then the same (and sometimes a better) upper bound on end-to-end delay is obtained with flow aggregation as compared to no flow aggregation. In this paper, we investigate end-to-end delays under flow aggregation, where the per-hop delay is not coupled with the reserved rate of the flow. We show that, even when the delay is de-coupled from the reserved rate, the end-to-end delay is preserved under flow aggregation.

Due to space restrictions, only selected proofs are presented. The remaining may be found at [2].

2. Network Model

A *network* consists of a set of computers connected via point-to-point communication channels. A *flow* in a computer network is a sequence of packets generated by a single application.

Each output channel of a computer is equipped with a scheduler. From the input channels, the scheduler receives packets from flows whose next hop to the destination is the output channel of the scheduler. The scheduler maintains a first-in-first-out queue for each flow. Whenever its output channel becomes idle, the scheduler chooses a received packet and forwards the packet to the output channel. The rate at which the scheduler forwards the packets of a flow must be at least the reserved rate of the flow.

We say a packet is *forwarded to the output channel* when the first bit of the packet is being transmitted by the channel. A packet *exits the scheduler* when the last bit of the packet is transmitted by the output channel. A packet is *in the output channel* if it has been forwarded to the output channel but has not yet exited the output channel. The *path* of flow f is the sequence of schedulers traversed by f from its source to its destination.

We adopt the following notation for each flow f and each scheduler s along the path of f:

- R.f forwarding rate (bytes/sec.) reserved for f.
- p.f.i ith packet from flow f, i ≥ 1.
- L.f.i packet length (bytes) of packet p.f.i.
- A.s.f.i arrival time to scheduler s of packet p.f.i.
- E.s.f.i exit time of packet p.f.i from scheduler s.
- $L_{max}.f$ upper bound on packet length for flow f.
- $L_{min}.f$ lower bound on packet length for flow f.
- $L_{max}.s$ upper bound on packet length for all flows at scheduler s.
- C.s capacity in bytes/sec. of the output channel of scheduler s.

3. Measuring Packet Delay

Before discussing flow aggregation, we begin by defining packet delays at a scheduler. In this section, we present how packet delays are assigned in our model, and the conditions under which the calculated delays can be guaranteed to each flow.

3.1. Start-Time and Finish-Time

Consider a scheduler s, an input flow f of s, and packet p.f.i. We define the *start-time* S.s.f.i and *finish-time* F.s.f.i as follows. Assume s were to forward the packets of f at *exactly* the rate R.f, as if f were its only input flow and C.s were equal to R.f. Then, S.s.f.i is the time at which p.f.i is forwarded to the output channel of s, and F.s.f.i is the time at which p.f.i exits scheduler s.

Definition 1

Let f be an input flow of scheduler s.

a) S.s.f.1 = A.s.f.1.

b) S.s.f.i = max(A.s.f.i, F.s.f.(i-1)), for every i, i > 1.

c) F.s.f.i = S.s.f.i + L.f.i/R.f, for every i, i ≥ 1.

♦

Since flow f reserves a rate R.f from scheduler s, one way to define delay is for s to forward each packet p.f.i no later than F.s.f.i plus a small constant $\alpha.s$ (usually $\alpha.s = L_{max}.f/C.s$). That is, s forwards the packets of f at least as fast as a constant-rate server. We refer to this delay as *rate-proportional delay*.

With rate-proportional delay, the per-hop delay of each packet of f is $L_{max}.f/R.f + \alpha.s = L_{max}.f/R.f + L_{max}.f/C.s$ [3][8]. This delay is coupled with the reserved rate of the flow. To decrease the per-hop delay, a greater rate must be reserved by the flow.

Another approach of assigning delays to packets is the real-time channel model [16]. Here, each flow is assigned a pair of values, ($\delta.f$, T). Packets from flow f are assumed to arrive with an inter-packet separation time of at least T seconds, and the scheduler guarantees their per-hop delay to be at most $\delta.f$. This delay is flexible, since each flow chooses its $\delta.f$ value, as opposed to rate-proportional delay, where the delay is coupled with the reserved rate. However, a schedulability test must performed to determine if the scheduler can satisfy the delay chosen by each flow, and the packet size is fixed. Necessary and sufficient conditions for the schedulability of this model may be found in [16].

In this paper, we choose a more flexible packet deadline based on the packet's start-time (similar to the delay chosen in [6]). That is, the exit time of a packet p.f.i from a scheduler s should be at most $S.s.f.i + \delta.s.f.i$ for some constant $\delta.s.f.i$.

Definition 2

A scheduler s is a *start-time* scheduler iff, for every input flow f of s and every i, i ≥ 1,

$$E.s.f.i \le S.s.f.i + \delta.s.f.i$$

for some constant $\delta.s.f.i$.

♦

We always assume schedulers are start-time schedulers.

Start-time delay is flexible enough to represent both rate-proportional delay and real-time channel delay. If we choose $\delta.s.f.i = L_{max}.f + \alpha.s$, start-time delay equals rate-proportional delay. Real-time channel delay is obtained from start-time delay by making packets of constant size, setting T = L.f/R.f, and preventing packet p.f.i from being considered for scheduling until clock ≥ S.s.f.i.

Since the start-time of a packet determines its exit time from a scheduler, we must examine how the start-time of a packet changes from one scheduler to the next.

Theorem 1

Let s and t be two consecutive schedulers in the path of flow f. For all i,

$$S.t.f.i \le S.s.f.i + \Delta.s.f.i$$

where $\Delta.s.f.i = \max_{0 \le x \le i} \{\delta.s.f.x\}$.

♦

A theorem similar to Theorem 1 was proven in [6]. From this bound on the start-time increase, we obtain a bound on end-to-end packet delay.

Theorem 2

Let t_1, t_2, … , t_k be a sequence of start-time schedulers traversed by flow f. For all i,

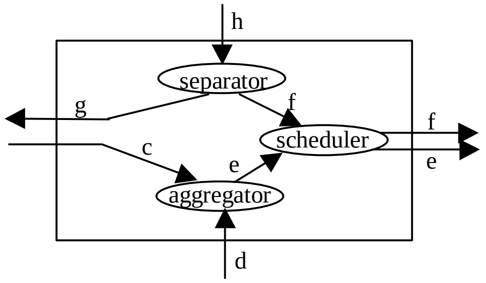

Figure 1 Example of flow aggregation

$$S.t_k.f.i \le S.t_1.f.i + \sum_{x=0}^{k-1} \Delta.t_x f.i$$

$$E.t_k.f.i \le S.t_k.f.i + \delta.t_k.f.i$$

♦

The first part of the Theorem follows from induction on the length k of the sequence of schedulers. The second part follows from the definition of a start-time scheduler.

Since rate-proportional delay can be represented using start-time delay, the end-to-end theorems using rate-proportional delay reported in [3] and [8] are a corollary of Theorem 2.

3.2. Schedulability Tests

With start-time delay, each flow chooses its desired per-hop delay δ.s.f.i at each scheduler s. Obviously, if all flows choose a very small delay, the scheduler will not be able to satisfy these delays. Therefore, a schedulability test is required to determine if it is possible to satisfy the deadlines required by each flow.

For the case of rate-proportional delay, where $\delta.s.f.i = L_{max}.f/R.f + L_{max}.f/C.s$, a necessary and sufficient condition for schedulability is the sum of the rates of the input flows must be at most the capacity of the output channel [12].

For start-time delay in general, we assume that δ.s.f.i is equal for all packets of f, so we refer to it simply as δ.s.f. Otherwise, a schedulability condition is difficult (if not impossible) to find.

We begin with the case in which all packets from the same flow are of the same size, i.e., $L_{min}.f = L_{max}.f$ for all f. The theorem and its proof is very similar to the real-time channel proof of [16], but not identical, since we assume no rate control on the input flows.

Theorem 3

Let $f_1, f_2, \dots, f_n$ be the input flows of a scheduler s. Also, let $0 < L_{min}.f_x = L_{max}.f_x$, $1 \le x \le n$. Flows $f_1, f_2, \dots, f_n$ are schedulable iff, for all t, t > 0,

$$(\Sigma\, x, \delta.s.f_x \le t, \left(\left\lfloor \frac{(t - \delta.s.f_x) \cdot R.f_x}{L_{max}.f_x} \right\rfloor + 1 \right) \cdot L_{max}.f_x)$$

♦

In Theorem 3, we test for all values of t, t > 0, for an unbounded number of tests. However, a test with a finite number of values of t can be obtained in the same way as done in [16].

We next consider the case of variable packet size, under the assumption that the maximum packet size of each flow is at least twice its minimum packet size.

Theorem 4

Let $f_1, f_2, \dots, f_n$ be the input flows of a scheduler s, and the sum of the rates of these flows is at most C.s. Also, let $0 < 2 \cdot L_{min}.f_x \le L_{max}.f_x$, $1 \le x \le n$. Finally, let $S_1 = \{\delta.s.f_1, \delta.s.f_2, \dots, \delta.s.f_n\}$, and $S_2 = \{\delta.s.f_1 + L_{min}.f_1, \delta.s.f_2 + L_{min}.f_2, \dots, \delta.s.f_n + L_{min}.f_n\}$. Flows $f_1, f_2, \dots, f_n$ are schedulable iff, for all t, $t \in S_1 \cup S_2$,

$$(\Sigma\, x, \delta.s.f_x \le t,$$

$$L_{max}.f_x + \begin{cases} (t - \delta.s.f_x) \cdot R.f_x & \text{if } t \ge \delta.s.f_x + \dfrac{L_{min}.f_x}{R.f_x} \\ 0 & \text{if } t < \delta.s.f_x + \dfrac{L_{min}.f_x}{R.f_x} \end{cases}$$

$$)$$

$$\le C.s \cdot t - \frac{L_{max}.s}{C.s}$$

♦

4. Flow Aggregation

We next describe the aggregation of multiple flows into a single flow. In the next section, we show the end-to-end delay achievable under flow aggregation.

A *simple flow* is a potentially infinite sequence of packets generated by an application. That is, the flows considered in earlier sections were simple flows.

A flow f is a *constituent* of flow g if the packets of f are a subset of the packets of g.

A scheduler that receives as inputs a set of flows $f_0, f_1, \dots, f_n$, and produces as output a *single* aggregate flow g, by merging the packets of the input flows, is called an *aggregator.* Flows $f_0, f_1, \dots, f_n$ are said to be the *immediate constituents* of flow g. Note that any constituents of flows $f_0, f_1, \dots, f_n$ are also constituents of g. The *reserved rate*, R.g, of aggregate flow g is the sum of the reserved rates of the immediate constituent flows of g.

A scheduler whose set of input flows is the same as its set output flows is called a *non-aggregating* scheduler.

For each input flow g of a scheduler, the scheduler is unaware of the constituent flows contained by g (or simply chooses to ignore them). It thus schedules the packets of g as if g were a simple flow with reserved rate R.g, i.e., the start and finishing times of each packet of g are defined as if g were a simple flow with rate R.g.

We assume all aggregators are start-time schedulers. That is, every input packet p.f.i will exit an aggregator s no later than time S.s.f.i + δ.s.f.i.

A *separator* is a process that receives as input an aggregate flow, and produces as output the set of immediate constituents of the input flow. We assume a separator causes no packet delay. Separators are the only processes aware of the immediate constituents of a flow.

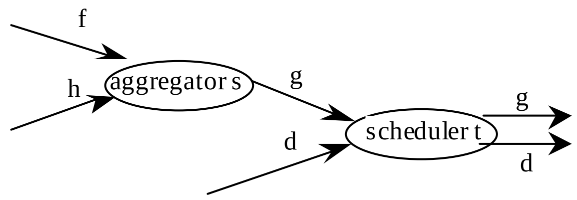

Figure 2: Fair Aggregators

We say that flow g is the *root* of flow f at some point in the network if f is a constituent of g, and g is not a constituent of any other flow at this network point.

Consider for example the computer with four input/output channels depicted in Figure 1. Here, flows f and g are the immediate constituents of h, and they are separated from h through a separator. Input flows c and d are aggregated together to form flow e. I.e., c and d are the immediate constituents of e. Flows e and f are scheduled by a non-aggregating scheduler and sent to an output channel without being aggregated together.

Note that the root of a flow may change as it traverses the network The root of flow f before entering the computer is h, and after exiting the computer is f itself.

Also, note that the aggregator and the scheduler are in the same computer, and hence, the aggregator may forward packets to the scheduler without delay. In this case, we view the output channel of the aggregator as having infinite capacity.

5. Fair Flow Aggregation

In this section, we investigate the end-to-end delay of a flow f as it traverses a network containing flow aggregators and separators. In order to guarantee a low end-to-end delay to each flow, the behavior of flow aggregators must be restricted. We begin by exploring this restriction.

5.1. Fair Aggregators

Consider Figure 2. Scheduler t is not aware that its input flow g is the aggregation of two flows, f and h. Hence, scheduler t only guarantees the exit time of each packet p.g.j to be at most $S.t.g.j + \delta.t.g.j$, and makes no guarantees about the exit times of packets from f and h.

Since the exit time at scheduler t depends on the start-time of each packet, we would like to bound the start-time of flow g at scheduler t as if no aggregation had occurred. For example, assume instead that s is a non-aggregating scheduler, and thus, f, h, and d are individual input flows of scheduler t. Then, from Theorem 1, and s being a start-time scheduler,

$$S.t.f.i \le S.s.f.i + \Delta.s.f.i$$

Let p.g.j = p.f.i, i.e., the j^{th} packet from g is the i^{th} packet from f. We would like to obtain a similar relationship between S.t.g.j and S.s.f.i as the one above between S.t.f.i and S.s.f.i. This will bound the exit time from t of p.f.i as a function of S.s.f.i.

We choose the bound as follows.

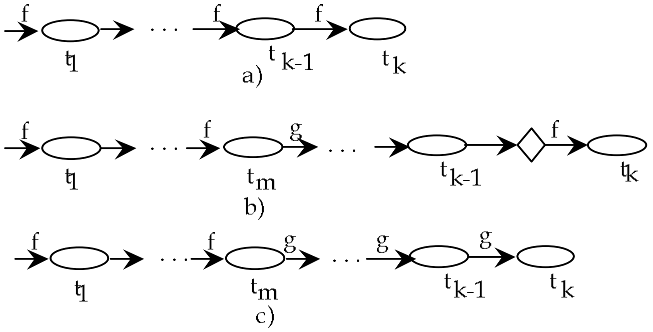

Figure 3: Induction Scenarios

(i) $$S.t.g.j \le S.s.f.i + \beta.s.f.i$$

for some constant $\beta.s.f.i$, $\beta.s.f.i \ge 0$. From (i), t being a start-time scheduler, and p.f.i = p.g.j, we conclude,

$$E.t.f.i = E.t.g.j \le S.t.g.j + \delta.t.g.j \le S.s.f.i + \beta.s.f.i + \delta.t.g.j$$
$$= S.s.f.i + \beta.s.f.i + \delta.t.f.i$$

Therefore, the bound of (i) above can be used to compute the exit time of p.f.i at scheduler t even though scheduler t is not aware of flow f. We formalize this definition next.

Definition 3

1. An aggregator s is *fair*, iff, for every immediate constituent f of its output flow g, there exists a constant $\beta.s.f.i$ such that
$$S.t.g.j \le S.s.f.i + \beta.s.f.i$$
where t is the next scheduler of g, and p.g.j = p.f.i.
2. A non-aggregating scheduler s is *fair*, iff, for every input flow f, there exists a constant $\beta.s.f.i$, such that
$$S.t.f.i \le S.s.f.i + \beta.s.f.i$$
where t is the next scheduler of f.
3. For a scheduler s (aggregating or non-aggregating), we define
$$B.s.f.i = \max_{k \le i} \{\beta.s.f.k\}$$

♦

Note that for a non-aggregating scheduler s, if s is a start-time scheduler, then from Theorem 1, $\Delta.s.f.i = \beta.s.f.i$. In Section 5.4, we show how fair aggregators can be constructed.

As mentioned in Section 3, all aggregators are assumed to be start-time schedulers. However, this does not imply that the aggregator is fair, as we show next.

Consider Figure 2, and assume that both f and h are generating packets at a rate greater than their reserved rate. The aggregator forwards packets from f at exactly the rate R.f, but it forwards the packets from h at a rate greater than R.h. Notice that the aggregator satisfies the definition of a start-time scheduler, since the packets of f will exit the aggregator close to their start-times.

However, it is possible that the queue of g will grow without bound at scheduler t, since packets of g arrive at a rate greater than R.g = R.f + R.h. Thus, the packets of f will be delayed excessively at scheduler t, because in the queue of g there is an excessive number of packets of h in between any two packets of f.

To prevent the above, the aggregator should be fair. That is, it should forward packets from f and h in a manner that ensures relation (i) holds, and also ensures a similar relation between h and g. We will see below that, by restricting all aggregators to be fair, an upper bound on the end-to-end delay of each flow, similar to the bound of Theorem 2, holds in the presence of flow aggregation.

5.2. End-to-End Delay

We next examine the end-to-end delay of a flow in the presence of aggregation.

Theorem 5

Let t_1, t_2, … , t_k be the path of fair schedulers traversed by flow f. Define rp.(s,f,i) to be the triplet s.g.j, where g is the root flow of f at scheduler s, and p.g.j = p.f.i. Then,

$$S.rp(t_k.f.i) \le S.t_1.f.i + \sum_{x=0}^{k-1} B.rp(t_x,f,i)$$

$$E.t_k.f.i \le S.rp(t_k.f.i) + \delta.rp(t_k,f,i)$$

Proof

Let depth.f be the maximum number of aggregations f goes through to obtain any of f's roots along the path of f. Let length.f be the number of schedulers in the path traversed by f. The proof is based on induction over the pair (depth.f, length.f).

For the base case, we consider pair (d, 1), i.e., a flow of any depth d through only one scheduler. In this case, Theorem 5 reduces simply to $S.t_1.f.i \le S.t_1.f.i$ and $E.t_1.f.i \le S.t_1.f.i + \delta.t_1.f.i$, which follows from the definition of a start-time scheduler.

For the induction case, we assume that Theorem 5 holds for all flows of depth less than depth.f, and for all flows at depth depth.f (including f) for the first k-1 schedulers along the path of the flow.

Consider Figure 3, which gives us three scenarios for the induction step. In scenario (a), the input to t_{k-1} is f, and the input to t_k is also f. From the induction hypothesis,

$$S.t_{k-1}.f.i \le S.t_1.f.i + \sum_{x=0}^{k-2} B.rp(t_x,f,i)$$

Since t_{k-1} is a fair scheduler, then

$$S.t_k.f.i \le S.t_{k-1}.f.i + B.t_{k-1}.f.i$$

Combining the above two relations, and noting that $rp(t_{k-1},f.i) = f.i$, we obtain the desired result. The value of $E.t_k.f.i$ follows from t_k being a start-time scheduler.

Consider now case (b). Here, the input to t_{k-1} is a flow whose depth is less than the depth of f (i.e., it contains f as a constituent flow). Let t_m be the last aggregator in the path of f whose input is f. Thus, g goes through t_{m+1} up to t_{k-1}. Let p.g.j = p.f.i. Note that the root flow of g is the same as that of f. Since the length of the path of g is less than k, then by the induction hypothesis,

$$\text{(ii)} \qquad S.rp(t_{k-1}.g.j) \le S.t_{m+1}.g.j + \sum_{x=m+1}^{k-2} B.rp(t_x,g,j)$$

$$= S.t_{m+1}.g.j + \sum_{x=m+1}^{k-2} B.rp(t_x,f,i)$$

Also, from the induction hypothesis,

$$\text{(iii)} \qquad S.t_m.f.i = S.rp(t_m.f.i) \le S.t_1.f.i + \sum_{x=0}^{m-1} B.rp(t_x,f,i)$$

Since t_m is a fair aggregator,

$$\text{(iv)} \qquad S.t_{m+1}.g.j \le S.t_m.f.i + B.t_m.f.i$$

Since t_{k-1} is a start-time scheduler, we have from (ii)

$$E.rp(t_{k-1}.g.j) \le S.t_{m+1}.g.j + \sum_{x=m+1}^{k-2} B.rp(t_x,f,i) + \delta.rp(t_{k-1},g,j)$$

Combining this with relation (iv), and $E.t_{k-1}.f.i = E.t_{k-1}.g.j$,

$$E.rp(t_{k-1}.f.i) \le S.t_m.f.i + B.t_m.f.i + \sum_{x=m+1}^{k-2} B.rp(t_x,f,i) + \delta.rp(t_{k-1},g,j)$$

Note that $rp(t_{k-1},g,j) = rp(t_{k-1},f,i)$ and $rp(t_m,f,i) = f.i$. Thus,

$$E.t_{k-1}.f.i \le S.t_m.f.i + \sum_{x=m}^{k-2} B.rp(t_x,f,i) + \delta.rp(t_{k-1},f,i)$$

Thus, the whole path from t_m up to t_{k-1} can be viewed as a single start-time scheduler P, whose δ.P.f.i value equals,

$$\sum_{x=m}^{k-2} B.rp(t_x,f,i) + \delta.rp(t_{k-1},f,i)$$

Note that B is an increasing function with each successive packet, and thus, the maximum of δ.P.f over all packets of f up to and including i, is at most

$$\sum_{x=m}^{k-2} B.rp(t_x,f,i) + B.rp(t_{k-1},f,i)$$

Hence, from Theorem 1

$$S.t_k.f.i \le S.t_m.f.i + \sum_{x=m}^{k-1} B.rp(t_x,f,i)$$

Combining the above with (iii),

$$S.rp(t_k.f.i) = S.t_k.f.i \le S.t_1.f.i + \sum_{x=0}^{k-1} B.rp(t_x,f,i)$$

which is the desired result. The value of $E.t_k.f.i$ follows from t_k being a start-time scheduler.

Case (c) in Figure 3 is similar (actually, simpler) than case (b), and is not presented due to space restrictions.
♦

By comparing Theorem 2 and Theorem 5, we see that the end-to-end delay obtained via flow aggregation is similar to the end-to-end delay obtained without flow aggregation. The following points about this theorem must be stressed.

Consider a pair of flows f and h that are aggregated to form flow g. Since each scheduler cannot distinguish between f and h, both flows will experience the same per-hop delay throughout the entire path of g. Therefore, f and h should be aggregated together only if they have similar delay requirements. This restriction is not too rigid, since it is unlikely that the per-hop delay requirements of flows will be diverse.

In [1], we presented a theorem for the end-to-end delay of a flow in the presence of aggregation. Here, the schedulers were rate-proportional schedulers. That, is, each packet p.g.j of an input flow g would exit scheduler s no later than F.s.g.j + α.s, where α.s is a constant, usually equal to L_{max}.s/C.s. Examples of this type of schedulers are Virtual Clock [12][13], PGPS [9], and Time-Shift Scheduling [4][10]. The per-hop delay at scheduler s is L_{max}.g/R.g + L_{max}.s/C.s. Notice that if f is a constituent flow of g, then R.f < R.g, and this delay with aggregation is smaller than the delay without aggregation, i.e., smaller than L_{max}.f/R.f + L_{max}.s/C.s, provided $L_{max}.g \approx L_{max}.f$. Thus, for rate-proportional schedulers, aggregation not only improves efficiency, but also decreases end-to-end delay.

The admission control test for rate-proportional schedulers is simply that the sum of the rates of the input flows must be at most the rate of the output channel. Hence, if the flows are schedulable without flow aggregation, then they are also schedulable under flow aggregation.

In the case start-time schedulers, as shown in previous sections, a more complex schedulability test is required to check if the requested deadline bounds can be satisfied. Note that Theorem 5 requires each scheduler to be fair, so we assume that each scheduler has passed its schedulability test. However, the question that must be answered is the following. Assume each flow has a per-hop delay requirement, and assume that all the flows are schedulable in a network without aggregation. Then, if we perform aggregation by aggregating together flows with the same per-hop delay requirement, would the schedulability test still be satisfied? We address this question in the next section.

5.3. Schedulability Tests

In this section, we examine whether the schedulability of flows is hindered by flow aggregation. We begin with the case when all packets of the same flow have the same packet size.

Theorem 6

Let $f_1, \dots, f_n, h_1, \dots, h_m$ be the input flows to a scheduler s. Let each flow have a constant packet size. Furthermore, let the packet sizes of flows $f_1, \dots, f_n$ be equal. We refer to the packet size of these flows as L.f. Let $\delta.s.f_x$ be also equal for every f_x, $1 \le x \le n$. If flows $f_1, \dots f_n, h_1, \dots, h_m$ are schedulable at s, then so are flows g, $h_1, \dots, h_m$, where $R.g = \sum_{x=1}^{n} R.f_x$, L.g = L.f, and δ.s.g = δ.s.f.

Corollary 1

Let $g_1, \dots, g_n$, be a set of aggregate flows, where all the constituent flows of g_x, $1 \le x \le n$, have the same constant packet size as g_x, and the same per-hop delay as g_x. Then, if the constituent flows of flows $g_1, \dots, g_n$ are schedulable at a scheduler s, then flows $g_1, \dots, g_n$ are also schedulable at s.

Proof

We have to show that for each time interval [0, t] of size t, the number of bytes from g whose start time is at least 0, and deadline is at most t, is at most the number of bytes from $f_1, \dots, f_n$. Since we will not be referring to the $h_1, \dots, h_m$ flows, and the deadlines and packet sizes of $f_1, \dots, f_n$, and g are the same, we drop the subscripts and refer to them as L and δ.

From the proof of Theorem 3 [2], the number of bytes with the deadline at most t from g are at most

$$\left(\left\lfloor \frac{(t-\delta)\cdot R.g}{L.} \right\rfloor + 1 \right)\cdot L$$

The number of bytes with deadline at most t from all flows f_x is at most

$$\sum_{x=1}^{n} \left(\left\lfloor \frac{(t-\delta)\cdot R.f_x}{L} \right\rfloor + 1 \right)\cdot L$$

Define $(t - \delta)\cdot R.f_x$ to be X_x, and define

$$\overline{X} = \sum_{x=1}^{n} X_x$$

Then the number of bytes from g becomes

$$\left(\left\lfloor \frac{\overline{X}}{L} \right\rfloor + 1 \right)\cdot L$$

and the number of bytes from each f_x becomes

$$\left(\left\lfloor \frac{X_x}{L} \right\rfloor + 1 \right)\cdot L$$

Therefore, we simply have to show that

$$\left(\left\lfloor \frac{\overline{X}}{L} \right\rfloor + 1 \right)\cdot L \quad \le \quad n\cdot L + \sum_{x=1}^{n} \left\lfloor \frac{X_x}{L} \right\rfloor \cdot L$$

Simplifying, we need to show that

$$\text{(v)} \qquad \left\lfloor \frac{\overline{X}}{L} \right\rfloor \le n - 1 + \sum_{x=1}^{n} \left\lfloor \frac{X_x}{L} \right\rfloor$$

In general,

$$\text{(vi)} \qquad \left\lfloor \frac{\overline{X}}{L} \right\rfloor \le \sum_{x=1}^{n} \left\lfloor \frac{X_x}{L} \right\rfloor + \left\lfloor \frac{\sum_{x=1}^{n} \left(X_x - \left\lfloor \frac{X_x}{L} \right\rfloor \cdot L \right)}{L} \right\rfloor$$

That is, the number of times L fits into $\overline{X}$ is at most the number of times it fits into each of the X_x values, plus the number of times L fits into the sum of the "leftovers" from each of the X_x.

Note also that,

$$\sum_{x=1}^{n}\left(X_x - \left\lfloor \frac{X_x}{L} \right\rfloor \cdot L\right) < n \cdot L$$

That is, each term in the sum is at least zero and less than L. Hence,

$$\left\lfloor \frac{\sum_{x=1}^{n}\left(X_x - \left\lfloor \frac{X_x}{L} \right\rfloor \cdot L\right)}{L} \right\rfloor \le n - 1$$

Thus, (v) follows from above and (vi). Corollary 1 follows from a simple induction over the aggregation level.

♦

Not only does flow aggregation improve the efficiency of scheduling and signaling in a network, but from Theorem 6, we are still capable of obtaining the same per-hop delay as in the case with no flow aggregation. However, we do have the restriction that only flows having the same per-hop delay can be aggregated together, and furthermore, all flows aggregated together must have the same packet size.

Currently, we are working on an analysis of the case where each individual flow has a constant packet size, but flows with different packet sizes are aggregated together.

We next consider the case when the packet size is not constant for each flow.

Theorem 7

Let $f_1, \ldots, f_n, h_1, \ldots, h_m$, be the input flows to a scheduler s. Let each of $f_1, \ldots, f_n$, have the same maximum and minimum packet size, denoted $L_{max}.f$ and $L_{min}.f$, respectively, where $L_{max}.f \ge 2 \cdot L_{min}.f > 0$. Furthermore, the per-hop delay of each of $f_1, \ldots, f_n$, at s is equal, and denoted by $\delta.s.f$. Lastly, $L_{max}.h_x \ge 2 \cdot L_{min}.h_x > 0$, for every x, $1 \le x \le n$. Then, if flows $f_1, \ldots, f_n, h_1, \ldots, h_m$ are schedulable at s, then so are flows $g, h_1, \ldots, h_m$, where $R.g = \sum_{x=1}^{n} R.f_x$, $L_{max}.g = L_{max}.f$, $L_{min}.g = L_{min}.f$, and $\delta.s.g = \delta.s.f$.

Corollary 2

Let $g_1, \ldots, g_n$, be a set of aggregate flows, where all the constituent flows of g_x, $1 \le x \le n$, have the same maximum and minimum packet size as g_x, and also the same per-hop delay at s as g_x. Also, $L_{max}.g_x \ge 2 \cdot L_{min}.g_x > 0$, for every x, $1 \le x \le n$. Then, if the constituent flows of flows $g_1, \ldots, g_n$ are schedulable at scheduler s, then flows $g_1, \ldots, g_n$ are also schedulable at s.

Proof

We have to show that for each time interval [0, t] of size t, the number of bytes from g whose start time is at least 0, and deadline is at most t, is at most the number of bytes from $f_1, \ldots, f_n$. Since each f flow and g have the same per-hop delay $\delta.s.f$, we only need to consider values of t at least $\delta.s.f$.

From the proof of Theorem 4, the number of bytes from flow g are at most

$$\text{(vii)}\ L_{max}.g + \begin{cases} (t - \delta.s.g) \cdot R.g & \text{if } t \ge \delta.s.g + \dfrac{L_{min}.g}{R.g} \\ 0 & \text{if } t < \delta.s.g + \dfrac{L_{min}.g}{R.g} \end{cases}$$

Recall the each flow f and flow g have the same packet sizes and same per-hop delay. Thus, the number of bytes from a flow f_x are at most

$$\text{(viii)}\quad L_{max}.g + \begin{cases} (t - \delta.s.g) \cdot R.f_x & \text{if } t \ge \delta.s.g + \dfrac{L_{min}.g}{R.f_x} \\ 0 & \text{if } t < \delta.s.g + \dfrac{L_{min}.g}{R.f_x} \end{cases}$$

Thus, our objective is to show that (vii) is at most the sum over all x of (viii).

Note that the left-hand-side of (vii) is $L_{max}.g$, and left-hand-side of (viii), over all x, is $n \cdot L_{max}.g$. Therefore, the left-hand-side of (viii) is at least $2 \cdot (n-1) \cdot L_{min}.g$ greater than the left-hand-side of (vii). Note that $2 \cdot (n-1) \cdot L_{min}.g$ is at least $n \cdot L_{min}.g$ for all n, $n \ge 2$.

Consider any f_x. Note that the maximum value of the right-hand-side of (viii) equals $(t - \delta.s.g) \cdot R.f_x$. Then, the sum of these over all x equals $(t - \delta.s.g) \cdot R.g$, which in turn is the maximum of the right-hand-side of (viii).

However, consider an f_x such that the right-hand-side of (viii) is 0. In this case, $t < \delta.s.g + L_{min}.g/R.f_x$, which implies that $(t - \delta.s.g) \cdot R.f_x$ is less than $L_{min}.g$. Thus, summing over all n, the right-hand-side of (viii) is at most $n \cdot L_{min}.g$ smaller than the right-hand-side of (vii). However, as discussed two paragraphs above, the left-hand-side of (viii) is at least $n \cdot L_{min}.g$ greater than the left-hand-side of (vii). Hence, (vii) is at most (viii), as desired.

The Corollary follows by a straightforward induction over the aggregation level.

♦

Theorem 7 is similar to Theorem 6, but for the case when packet sizes are not constant. Thus, as long as the constituents of a flow have the same upper and lower bounds on message sizes, and also the same per-hop delay requirements, then aggregating flows does not hinder the schedulability of packets.

5.4. Implementing Fair Aggregators

Thus far, we have considered the end-to-end packet delay under flow aggregation, and the schedulability tests required to ensure this delay. However, we have not addressed how a fair aggregator may be implemented.

Let g be the output flow of an aggregator s, and the channel capacity of the aggregator be C.s. A simple technique to construct an aggregator is the following. Consider a fictitious start-time scheduler v whose output channel has capacity R.g and has the same input flows as aggregator s. Aggregator s forwards the packets in the same order as the fictitious scheduler v. However, after the aggregator forwards a packet of length L, the aggregator does not forward another packet until L/R.g seconds later, even though the packet takes only L/C.s seconds to transmit.

We call a fair aggregator constructed from the above technique a *basic fair-aggregator.* Note that the above actually defines a whole family of basic fair aggregators, one family member for each possible type of start-time scheduler emulated.

Theorem 8

Let s be a basic fair-aggregator, f be one of its input flows, and g its output flow. Furthermore, let v be the fictitious scheduler emulated by the basic fair-aggregator. Then,

$$S.t.g.j \leq S.s.f.i + \delta.v.f.i - L.f.i/R.g + L.f.i/C.s + L_{max}.g/C.s$$

where t is the scheduler after s, and p.g.j = p.f.i.

Proof

First, it is easy to show by induction that for all j, $A.t.g.j \geq F.t.g.(j\text{-}1) - L_{max}.g/C.s$ (a proof may be found in [1]). Thus, from Definition 1, $S.t.g.j \leq A.t.g.j + L_{max}.g/C.s$.

Furthermore, since v is a start-time scheduler, and the output channel rate of s is C.s, then $A.t.g.j \leq S.s.f.i + \delta.v.f.i - L.f.i/R.g + L.f.i/C.s$. Hence,

$$S.t.g.j \leq S.s.f.i + \delta.v.f.i - L.f.i/R.g + L.f.i/C.s + L_{max}.g/C.s$$

♦

Thus, we can construct a fair aggregator from a start-time scheduler using the technique above. Notice that if the start-time scheduler v being emulated is work-conserving, then an input flow f of the aggregator s will be allowed to forward packets at a rate greater than R.f, but not at a rate greater than R.g. Thus, if the scheduler v distributes unused bandwidth in a fair manner among its input flows, then aggregator s will also distribute unused bandwidth (up to rate R.g) among its input flows.

6. Concluding Remarks

In this paper, we have defined the aggregation of multiple flows into a single flow, and how the end-to-end delay bound is preserved in spite of flow aggregation. The advantages of the scheme is simplified scheduling and signaling due to the reduction in the number of input flows to a scheduler. The disadvantage is that flow aggregation has to be performed by a non-work conserving scheduler.

In [5], a different approach is taken for the aggregation of flows. The delay-jitter of a flow f that is aggregated with other flows to form flow g is set to zero. Thus, the characteristics of f once it is separated from g are identical to its characteristics when it was aggregated into g (modulo an equal delay applied to all packets). The disadvantage of this approach is that f cannot take advantage of unused bandwidth and temporarily exceed its reserved rate R.f. In our approach, a flow f can exceed its reserved rate R.f up to the reserved rate R.g of its parent flow, allowing for a better use of network bandwidth.

In [10], a rate-proportional protocol is presented for a network core without per-flow state. This is advantageous, since no flow state needs to be maintained. However, the packet delay is the same as with per-flow state techniques, and hence higher than the delay with flow aggregation.

References

[1] Cobb J, "Preserving Quality of Service Guarantees In-Spite of Flow Aggregation", *IEEE ICNP* 1998

[2] Cobb J., "An In-Depth Look at Flow Aggregation", available from www.utdallas.edu/~jcobb.

[3] Cobb J., Gouda M., "Flow Theory", *IEEE/ACM Transactions on Communications,* January 1998.

[4] Cobb J., Gouda M., El-Nahas A., "Time-Shift Scheduling: Fair Scheduling of Flows in High-Speed Networks", *IEEE/ACM Transactions on Communications,* June 1998.

[5] Cruz R. L., "SCED+: Efficient Management of Quality of Service Guarantees", *INFOCOM 1998.*

[6] Figueira N., Pasquale J., "Leave-in-Time: A New Service Discipline for Real-Time Communications in a Packet-Switching Data Network", *SIGCOMM 1995.*

[7] Gall D., "A Video Compression Standard for Multimedia Applications", *Com. of the ACM,* 34(4), April 1991.

[8] Goyal P, Lam S., Vin H., "Determining End-to-End Delay Bounds in Heterogeneous Networks", *NOSSDAV,* 1995.

[9] Parekh A. K. J., Gallager R., "A generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single Node Case", *IEEE/ACM Transactions on Networking,* 1(3):344-357, June 1993.

[10] Stoica I, Zhang H, Providing Guaranteed Services without Per-Flow Management, *ACM SIGCOMM '99.*

[11] Suri S., Varghese G., "Leap-Forward Virtual Clock: A New Fair Queuing Scheme with Guaranteed Delays and Throughput Fairness", *INFOCOM 1997.*

[12] Xie G., Lam S., "Delay Guarantee of Virtual Clock Server", *IEEE/ACM Trans. on Networking,* Dec. 1995.

[13] Zhang L., "Virtual Clock: A New Traffic Control Algorithm for Packet-Switched Networks", *ACM Transactions on Computer Systems*, Vol. 9, No. 2, May 1991.

[14] Zhang H., Keshav S., "Comparison of Rate-Based Service Disciplines", *ACM SIGCOMM Conference*, 1991.

[15] Zhang H., "Service Disciplines for Guaranteed Performance Service in Packet-Switching Networks", *Proceedings of the IEEE,* Vol. 83, No. 10, Oct. 1995.

[16] Zheng Q., Shin K.G., "On the Ability of Establishing Real-Time Channels in Point-to-Point Packet-Switched Networks", *IEEE Trans. on Comm.,* Vol 42, No. 2/3/4, 1994.