Delay Bounds in a Network With Aggregate Scheduling

Anna Charny * Jean-Yves Le Boudec †

April 14, 2000

Abstract

A large number of products implementing aggregate buffering and scheduling mechanisms have been developed and deployed, and still more are under development. With the rapid increase in the demand for reliable end-to-end QoS solutions, it becomes increasingly important to understand the implications of aggregate scheduling on the resulting QoS capabilities. This paper studies the bounds on the worst case delay in a network implementing aggregate scheduling. We derive an upper bound on the queuing delay as a function of priority traffic utilization and the maximum hop count of any flow, and the shaping parameters at the network ingress. Our bound explodes at a certain utilization level which is a function of the hop count. We show that for a general network configuration and larger utilizaion utilization an upper bound on delay, if it exists, must be a function of the number of nodes and/or the number of flows in the network.

Keywords: delay, jitter, aggregate scheduling, FIFO, priority queueing, Guaranteed Service, Diffserv

1 Introduction and Background

In the last decade, the problem of providing end-to-end QoS guarantees in Integrated Services networks has received a lot of attention. One of the major challenges in providing hard end-to-end QoS guarantees is the problem of scalability. Traditional approaches to providing hard end-to-end QoS guarantees, which involve per flow signalling, buffering and scheduling, are difficult, if not impossible, to deploy in large ISP networks. As a result, methods based on traffic aggregation have recently become widespread. A notable example of methods based on aggregation has been developed in the Differentiated Services Working group of IETF. In particular, RFC 2598[1] defines Expedited Forwarding per Hop Behavior (EF PHB), the underlying principle of which is to ensure that at each hop the aggregate of traffic requiring EF PHB treatment receives service rate exceeding the total bandwidth requirements of all flows in the aggregate at this hop. Recently, a lot of practical implementations of EF PHB have emerged where all EF traffic is shaped and policed at the backbone

*Corresponding author: Cisco Systems, 300 Apollo Drive, Chelmsford, MA 01776, fax (978)-244-8126, acharny@cisco.com

†ICA-EPFL, INN, Ecublens, CH- 1015 Lausanne-EPFL, Switzerland, fax +41 21 693 6610, leboudec@epfl.ch

ingress, while in the core equipment all EF traffic shares a single priority FIFO or single high-weight queue in a Class-Based Fair Queuing scheduler. Since these implementations offer a very high degree of scalability at comparatively low price, they are naturally very attractive.

Sufficient bandwidth must be available on any link in the network to accommodate bandwidth demands of all individual flows requiring end-to-end QoS. One way of ensuring such bandwidth availability is by generously over provisioning the network capacity. Such over provisioning requires adequate methods for measuring and predicting traffic demands. While a lot of work is being done in this area, there remains a substantial concern that the absence of explicit bandwidth reservations may undermine the ability of the network to provide hard QoS guarantees. As a result, approaches based on explicit aggregate bandwidth reservations using RSVP aggregation are being proposed [2]. The use of RSVP aggregation allows the uncertainty in bandwidth availability to be overcome in a scalable manner. As a result, methods based on RSVP aggregation can provide hard end-to-end bandwidth guarantees to individual flows sharing the traffic aggregate with a particular aggregate RSVP reservation.

However, regardless of whether bandwidth availability is ensured by over provisioning, or by explicit bandwidth reservations, there remains a problem of whether it is possible to provide not only bandwidth guarantees but also end-to-end latency and jitter guarantees for traffic requiring such guarantees, in the case when only aggregate scheduling is implemented in the core.

An important example of traffic requiring very stringent delay guarantees is voice. A popular approach to supporting voice traffic is to use EF PHB and serve it at a high priority. The underlying intuition is that as long as the load of voice traffic is substantially smaller than the service rate of the voice queue, there will be no (or very little) queueing delay and hence very little latency and jitter. However, little work has been done in quantifying exactly how much voice traffic can actually be supported without violating stringent end-to-end delay budget that voice traffic requires.

Another example when a traffic aggregate may require a stringent latency and jitter guarantee is the so-called Virtual Leased Line (VLL) [1]. The underlying idea of the VLL is to provide a customer with an emulation of a dedicated line of a given bandwidth over the IP infrastructure. One attractive property of the real dedicated line is that a customer who has traffic with diverse QoS requirements can perform local scheduling based on those requirements and local bandwidth allocation policies, and the end-to-end QoS experienced by his traffic will be entirely under the customers control. One way to provide a close emulation of this service over IP infrastructure could be to allocate a queue per each VLL in a WFQ scheduler at each router in the network. However, since there could be a very large number of VLLs in the core, per-VLL queueing and scheduling presents a substantial challenge. As a result, it is very attractive to use aggregate queueing and scheduling for VLL traffic as well. However, aggregate buffering scheduling introduces substantially more jitter than per-VLL scheduling. This jitter results in substantial deviation from the hard pipe model of the dedicated link, and may cause substantial degradation of service as seen by delay-sensitive traffic inside a given VLL. Hence, understanding just how severe the degradation of service (as seen by delay sensitive traffic within a VLL) can be with aggregate scheduling is an important problem that has not been well understood.

One of the goals of this document is to quantify the service guarantees that can be achieved with aggregate scheduling. There are two approaches that can be taken in pursuit of this goal. One is to analyze statistical behavior of the system and to quantify ”statistical guarantees”. While a large amount of work has been done in this area, there is still an uncertainty about what the appropriate traffic model should be. In this respect, an interesting approach is presented by Brichet et. al. in [3], where it is conjectured that the statistical behavior of periodic traffic entering a network of deterministic servers is ”better” than the behavior of Poisson traffic. If this conjecture is true, then the implication would be that the queueing delay and resulting jitter of strictly periodic traffic would be no worse that the behavior of a tandem of M/D/1 servers. Another approach to quantifying delay guarantees it to analyze the deterministic worst case performance. Such analysis is of immediate importance in the context of providing end-to-end Guaranteed Service [4] across a network with aggregate scheduling, such as a Diffserv cloud implementing EF PHB. Since GS definition requires availability of a meaningful mathematical delay bounds, the ability to provide a deterministic delay bound across a Diffserv cloud would enable supporting end-to-end Guaranteed Service to a flow even if its path traverses a Diffserv domain.

Understanding worst case delay bounds in a Diffserv cloud is also important in the context of supporting VoIP applications over a Diffserv network. Encoded voice traffic is periodic by nature, and voice generated by devices such as VoIP PBXs using hardware coders synchronized to network SONET clocks can additionally be highly synchronized. The consequence of periodicity is that if a bad pattern occurs for one packet of a flow, it is likely to repeat itself periodically for packets of the same flow. Therefore, it is not just a random isolated packet that can be affected by a bad pattern, making it very unlikely that more than a very small number of packets of any given flow will be affected (and hence justifying the assumption that the users will never notice such a rare event) - rather, it is a random flow or set of flows that may see consistently bad performance for the duration of the call. As the amount of VoIP traffic increases, every now and then a bad case will happen, and even if does not happen that frequently, when it does occur, a set of users are most certainly likely to suffer potentially severe performance degradation. These considerations provide motivation to take a closer look at the worst case behavior. On the other hand, the temporary nature of voice traffic (e.g. conversations starting and ending) disrupts strict periodicity of voice traffic. Furthermore, in practice it should be expected that aperiodic traffic may be sharing the same aggregate queues with periodic traffic, (e.g. mission critical medical application sharing the same EF queue with voice flows). Hence, basing worst case results on the assumption of strict periodicity may yield erroneous results.

There is a widespread belief, based largely on intuition, that as long as the utilization on any link is kept small enough (such as less than 50%), the worst case delay through the network will be very small. Unfortunately, this intuition leads to erroneous conclusions. Prior analytical work on studying the delay bounds in a network with aggregate scheduling indicates that networks with aggregate scheduling may have unbounded queueing delay even with aggregate output shaping [10]. On the other side of the spectrum, several researchers demonstrate that the ability to provide good delay bounds may depend on complex global conditions [11], [12]. In particular, this work assumes that individual flows are shaped at the network entry in such a way that the spacing between packets is at least equal to the so-called route interference number (RIN)[1]. It is shown that in this case the end-to-end

[1]RIN is defined as the number of occurrences of a flow joining the path of some other flow.

result is bounded by the time to transmit a number of packets equal to the RIN.

The contribution of this paper is twofold. First we show that for sufficiently low enough utilization factors, deterministic delay bounds can be obtained as a function of the bound on utilization of any link and the maximum hop count of any flow. Second, we argue that for larger utilization, for any given values of delay, hop count and utilization it is possible to construct a network in which the queuing delay exceeds the chosen delay value. This implies that for large enough utilization, either the bound does not exist at all, or if it does exist it must be a function of the size and possibly topology of the network.

2 Model, Assumptions and Terminology

We assume that all nodes in the network are output-buffered devices implementing aggregate class-based strict Priority Scheduling[2]. Traffic enters the network at some nodes (referred to as *ingress edge*) and exits at some other nodes (referred to as *egress edge*). We consider at least 2 classes of edge-to-edge (e2e) *flows,* where a flow is some collection of packets with the common ingress and egress edge pair. We will refer to one of the classes as *priority* class. At each node in the network packets belonging to the priority class are queued in a separate *priority queue* which is served at strict non-preemptive priority over any other queue. The total traffic of all priority flows sharing a particular link is refereed to as a *priority aggregate.* We assume that each e2e priority flow F is shaped to conform to a leaky bucket with parameters (R_F, B_F) when it arrives at the ingress edge. Note that the flow can itself consist of a number of microflows sharing the same ingress-egress edge pair, but no assumption is made on how those microflows are shaped. Our model further *allows* that the aggregate of all flows entering at a given ingress point, regardless of their egress edge, can be additionally aggregately shaped, but our results do *not* depend on such aggregate shaping.

Let $S(l)$ denote the set of all priority flows constituting the priority aggregate on link l. We assume that the amount of priority traffic on any link does not exceed a certain ratio $0 < \alpha < 1$ of the capacity of any link. More specifically we require that for any link l in the network $\sum_{F \in S(l)} R_F \leq \alpha C(l)$ where $C(l)$ is the capacity of link l. We also require that we can bound the sum of the token bucket depths burst at any link relative to the link capacity, namely we can express some parameter τ such that for all links l $\sum_{F \in S(l)} B_F \leq \tau C(l)$. In many cases, the depth of the leaky bucket of a flow depends linearly on the rate of the flow, such that $B_F \leq \tau_0 R_F$ for every flow F and for some τ_0. In such cases we set $\tau = \alpha \tau_0$. This paper makes no assumptions about the exact mechanism ensuring the conditions above – it can be based on explicit reservations such as described in [2], or some reliable provisioning technique. In addition, we assume that the route of any flow in the network traverses at most h nodes (also referred to as hops). Finally, we assume that there exists a bound on the size of any packet in the network, denoted MTU.

[2] Our results easily extend to class-based WFQ schedulers as well.

3 A Delay Bound for A General Topology

We can abstract our node description by saying that the priority aggregate at link l is guaranteed a service curve equal to $\sigma_l(t) = (tC(l) - MTU)^+$ [5, 6, 7, 8]. Also call $\Gamma(l)$ a bound on the peak rate of all incoming high priority traffic at link l. If we have no information about this peak rate, then $\Gamma(l) = +\infty$. For a router with large internal speed and buffering only at the output, $\Gamma(l)$ is the sum of the bit rates of all incoming links. The delay bound is better for a smaller $\Gamma(l)$. Also, define $u(l) = \frac{\Gamma(l)-C(l)}{\Gamma(l)-\alpha C(l)}$. Note that $u(l)$ increases with $\Gamma(l)$, and if $\Gamma(l) = +\infty$, then $u(l) = 1$. Finally, let $u = \max_l u(l)$ and $\Delta = \max_l \frac{MTU}{C(l)}$.

Theorem 1 *If $\alpha < \min_l \frac{\Gamma(l)}{(\Gamma(l)-C(l))(h-1)+C(l)}$ then a bound on the end-to-end delay for high priority traffic is*

$$D = \frac{h}{1-(h-1)u\alpha}(\Delta + u\tau)$$

Proof The proof consists in showing separately that (1) if a finite bound exists, then the formula in Theorem 1 is true and (2) that a finite bound exists.

(Part 1) We assume that a finite bound exists. Call D' the worst case delay across any node. Let $a_F(t) = tR_F + B_F$ be the arrival curve for the fresh traffic of flow F at the network entry. Consider a buffer at some link l. An arrival curve for the aggregate traffic at the input of link l is

$$a(t) = \min\left(t\Gamma(l), \sum_{F \in S(l)} a_F(t + (h-1)D')\right)$$

The former part in the formula is because the total incoming bit rate is limited by $\Gamma(l)$; the latter is because any flow reaching that node has undergone a delay bounded by $(h-1)D'$. Thus

$$a(t) \le a'(t) = \min(t\Gamma(l), t\alpha C(l) + b')$$

with $b' = \tau C(l) + \alpha C(l)(h-1)D'$. A bound on the delay at our node is given by the horizontal deviation between the arrival curve $a'(t)$ and the service curve $\sigma_l(t) = (tC(l) - MTU)^+$ [5, 6, 7, 8]. After some algebra, this gives a bound on the delay as $\frac{MTU+u(l)b'}{C(l)}$. Now D' is the worst case delay; pick l to be a node where the worst case delay D' is attained. We must have $D' \le \frac{MTU+u(l)b'}{C(l)}$. Thus, we must have

$$D' \le \Delta + u\tau + (h-1)u\alpha D'$$

or equivalently

$$(1-(h-1)u\alpha)D' \le \Delta + u\tau \tag{1}$$

α	0.04	0.08	0.12	0.16	0.20
$D(\Gamma(l) = +\infty)$	16.88	74.29	$+\infty$	$+\infty$	$+\infty$
$D\ (\Gamma(l) = 2C(l))$	12.75	32.67	71.50	186.00	$+\infty$

Table 1: The bound D (in ms) in this paper for $h = 10$. $B_F = 100$B for all flows, $R_F = 32$kb/s for all flows, $C(l) = 149.760$Mb/s and $\Gamma(l) = +\infty$ or $\Gamma(l) = 2C(l)$.

The condition on α means that $\alpha < \frac{1}{(h-1)u}$. Equation (1) implies then that

$$D' \leq D_1 = \frac{\Delta + u\tau}{1 - (h-1)u\alpha}$$

The end-to-end delay is thus bounded by hD_1, which after some algebra provides the required formula.

(Part 2) We now prove that a finite bound exists. We use the time-stopping method in [13]. For any time $t > 0$, consider the virtual system made of the original network, where all sources are stopped at time t. This network satisfies the assumptions of part 1, since there is only a finite number of bits at the input. Call $D'(t)$ the worst case delay across all nodes for the virtual network indexed by t. From the above derivation we see that $D'(t) \leq D_1$ for all t. Letting t tend to $+\infty$ shows that the worst case delay at any node remains bounded by D_1 ∎.

Discussion: If $\Gamma(l) = +\infty$, namely if we have no information about the sum of the bit rates of all incoming links, then the condition becomes $\alpha < \frac{1}{h-1}$ and the bound is

$$D = h\frac{\Delta + \tau}{1 - (h-1)\alpha}$$

For finite values of $\Gamma(l)$, the bound is smaller. Table 1 illustrates the value of our bound on one example. Our bound explodes when α tends to $\min_l \frac{\Gamma(l)}{(\Gamma(l)-C(l))(h-1)+C(l)}$, which is very unfortunate. It is not clear whether the delay does become unbounded in all cases where our bound is infinite. Explosion of bounds for aggregate scheduling is not new, and has been encountered for example in [9, 10]. In [10], an example network is given where the worst case delay becomes unbounded for a value of α close to 1. Note that if $\Gamma(l)$ is close to $C(l)$ for all l, then u is close to 0. This is the case for a high speed add-drop ring network, where the bit rate is much higher for the ring than for the input links. In that case, $D = h(\Delta + u\tau) + o(u)$ and the explosion does not take place. It is also known that, in practice, if some restriction on the topology, routes or rates are imposed, better worst case delays for FIFO networks can be obtained. For example, it can be shown that for some topologies, such as multistage networks, the worst case delay is given by $\frac{(1+\alpha)^h - 1}{\alpha}(\frac{\alpha b_{\max}}{R_{\min}} + \Delta)$, where $b_{\max}$ and $R_{\min}$ are the upper bound on the token bucket depth and the lower bound on the rate of all ingress leaky bucket shapers [14]. In this case the explosion does not take place. It is shown in [11, 12], that if individual flows are shaped at network entry in such a way that the spacing between packets is at least equal to the route interference number (RIN) of the flow (see footnote 1 p.4), then the end-to-end delay is bounded by the time to transmit a number

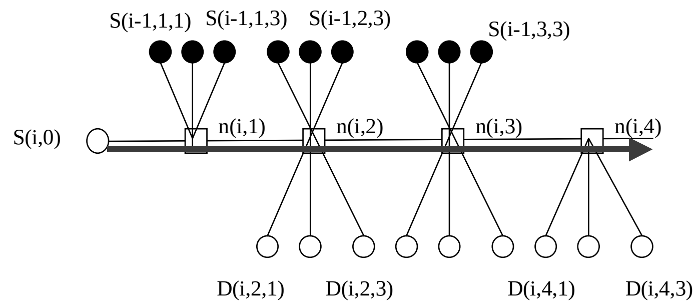

Figure 1: The iteratitive network topology of iteration i for $h = 5$. Black circles represent networks constructed at iteration $i - 1$. Only 3 out of kh black circles are shown per each node $n(i, j)$. The output of node $n(i, 4)$ in iteration $i - 1$ denoted by the arrow point is connected to the input of one of the black circles of iteration i.

of packets equal to the RIN. This usually results in much smaller bounds than the bounds here, albeit at the expense of more restrictions on the routes and the rates.

4 Delay With Larger Utilization

We will now argue that if utilization factor $\alpha > \frac{1}{h-1}$, then the worst case delay bound, if it exists, must depend on some network parameter describing the size of the network or its topology. More precisely, it will be shown that for any value of the delay D, and for any $\alpha > \frac{1}{h-1}\frac{kh+1}{kh-1}$, there exists a large enough network such that the delay of some packet can exceed D, even if the maximum hop count of any flow never exceeds h. Since k can be chosen to be arbitrarily large, this implies that for any $\alpha > \frac{1}{h-1}$, and for any value of delay D, there exists a network with maximum hop count h where the delay of some packet exceeds D.

To see how this network can be built, consider the following iterative construction, shown in Figure 1. The basic building block of the construction is a chain of $h - 1$ nodes. The nodes are indexed $n(i, j)$, where i is the iteration count and j is the count of the nodes in the chain. At any iteration i, there is a source $s(i, 0)$ which is attached directly to node $n(i, 1)$. Each of the nodes $n(i, j), j = 1, 2, ...h - 2$ also have kh input links each, and all nodes $n(i, j), j = 2, ...h - 1$ have hk output links. Each of the kh input links of iteration i are connected to node $n(i-1, h-1)$ of a network constructed at iteration $i-1$ (the networks constructed at iteration $i - 1$ are denoted $S(i - 1, j, m)$, where j is the index of the node in the chain, and m is the index of the input to node $n(i, j)$. Each of the kh output links corresponding to each node $n(i, j)$ is connected to kh destination nodes, denoted $D(i, j, m)$.

At iteration $i = 1$, each of the networks $S(0, j, m)$ (i.e. each of the black circles) is just a single source node. Assume that all packets are the same size, and that all the sources in the network are leaky bucket constrained with rates $r = \frac{\alpha C}{kh+1}$ and burst (depth) equal to one packet. Assume that a flow originating at source $S(0, j, m)$ enters at node $n(1, j)$ of the chain, traverses a single hop of the chain and exits at node $n(i, j+1)$ to destination $D(0, j+1, m)$. Assume further that a flow originating at source $s(1, 0)$ traverses the whole chain. Note that the utilization of any link in the chain does not exceed α . Assume that the propagation time of any link is negligible (it will become clear that this assumption does not lead to any loss of generality). Assume also that all links are of the same speed, and that the time to transmit the packet at link speed is chosen as unit of time.

At iteration $i = 1$ at time $t = 0$ we make a single packet fully arrive to node $n(1, 1)$ from each of the sources $S(1, 1, m)$. Let's color these packets blue. Assume also that at time zero a red packet started its transmission from node $S(1, 0)$ and fully arrived to $n(1, 1)$ at time $t_1 = 1$. By this time one blue packet of the hk blue packets which arrived at time $t = 0$ has departed from $n(i, 1)$, and therefore the red packet has found a queue of length $kh - 1$ blue packets ahead of it. By time $t_2 = t_1 + kh$ the red packet will be fully transmitted from node $n(1, 1)$. Assume that at time $t_2 - 1$ a single blue packet fully arrives from each of the nodes $S(1, 2, m)$, so that the red packet finds a queue of size $kh - 1$ at the second node in the chain as well (recall that all the blue packets from the previous node exit the network at node $n(1, 2)$.

Repeating this process at each of the nodes in the chain, it is easy to see that the red packet will suffer the total queueing delay of $(h - 1)(hk - 1)$ packet-times at link speed as it traverses the chain. Since the rate of the red flow is $r = \frac{\alpha C}{kh+1} > \frac{kh-1}{kh+1} \frac{1}{h-1} \frac{C}{(h-1)(kh-1)}$, it means that in $(h-1)(hk-1)$ packet times at link speed C at least one more red packet can be transmitted from the source $S(1, 0)$. Since there are no other blue packets arriving to the chain by construction[3], the second red packet will catch up with the first red packet, and hence the burst of two red packets will accumulate at the exit of node $n(1, h-1)$. Assume that the red flow at the first iteration just sends two packets and exits.

The next step of the iterative construction will take $(h-1)hk$ of the networks constructed in step 1 and connect the output of node $n(1, h-1)$ of each of these networks (node $n(1, 4)$ in Fig. 1) to one of the nodes $n(2, j)$ of the second iteration. That is, node $n(i-1, h-1)$ becomes the source $S(i-1, j, m)$ of the chain at the next iteration. Let $T(1)$ be the time it takes to accumulate the two-packet burst at the exit from node $n(1, h-1)$ at iteration 1. Let then $T(1)$ be also the time when the first bit of the two-packet burst arrives at node $n(2, 1)$ from each of the nodes $S(1, 1, m)$ at iteration 2. Let's recolor all those packets blue. If the source $S(2, 0)$ emits a red packet at time $t_1 = T(1) + 1$, then it fully arrives at node $n(2, 1)$ at time $t_2 = T(1) + 2$, and finds the queue equal to $2hk - 1$ blue packets (since $2hk$ packets have arrived and one has left in the previous time slot. That means that the last bit of the red packet will leave node $n(2, 1)$ at time $T(1) + 2 + 2kh$. Assume now that the level-1 networks connected to node $n(2, 2)$ are started at time $2kh$, instead of at time zero. This means that by the time the last bit of our red packet in the second iteration arrives to node $n(2, 2)$, a two-packet (blue) burst will arrive to node $n(2, 2)$ from each of the networks $S(1, 2, m)$, and so the red packet will again find the queue of size $2kh - 1$ blue packets. Repeating this process for all $h-1$ nodes in the chain at iteration 2, we make the red packet delayed by $(2kh - 1)(h - 1) > 2(kh - 1)(h - 1)$ packet times. During this time 2 more red packets will catch up, resulting in a 3-packet burst. Assume that at iteration 2 the red flow

[3]Note that the fact that blue sources are leaky bucket constrained with rate r does not prevent the possibility that they are sending at rate slower than r. In particular, each blue source can send just a single packet and then stop without violating any of the constraints.

sends 3 packets and exits.

Repeating this process as many times as necessary, with the red flow sending $i+1$ packets at iteration i and accumulating the burst of size $i+1$, we can grow a burst of red packets at least by one at each new iteration. Therefore, for any value of the delay D, we can always build a network of enough iterations, so that the burst B accumulated by the red flow over $h-1$ hops is such that $B/C > D$. Hence, taking one more iteration, it is easy to see that the red packet at the next iteration will be delayed by more than D at the first hop.

We emphasize that at each iteration the red flow traverses $h-1$ hop, and then one more hop as it is recolored blue in the next iteration. Hence, no flow in the network ever traverses more than h hops.

An interesting implications of this example is that the delay of a packet of a flow may be affected by flows which do not share any links in common with the given flow, and what is even more surprising, those flow had exited the network long before the flow to which the packet belongs has entered.

5 Discussion

The main difficulty in obtaining hard delay bounds for an arbitrary topology arises from the fact that in the case of aggregate scheduling, delay experienced by a single packet depends not only on the behavior of the flows that share at least one queue with the packet in question, but also on the behavior and arrival pattern of flows in the entire network, potentially much before the flow to which our packet belongs enters the network. As shown in the example in the previous section, a packet can encounter accumulated bursts of traffic that was initially ideally smooth at the entry, which in turn causes further burst accumulation of the considered flow. In turn, the new burst may contribute to further cascading jitter accumulation of other flows whose packets it encounters downstream. Intuitively, the lower the utilization factor and the shorter the route of the flow, the fewer chances a packet of that flow has in encountering other bursts, accumulating its own, and adversely affecting other traffic downstream. Our work has provided some quantification of this intuition.

The ability to provide hard delay bounds in a general topology network with aggregate scheduling must rely on cooperation of all devices in the cloud, as well as strict constraints on the traffic entering the cloud. It has been demonstrated that the global control of the following parameters global to the network is essential for the ability to provide strict queuing delay guarantees in such a network:

1. limited number of hops of any flow across the network
2. low (bounded) ratio of the load of priority traffic to the service rate of the priority queue on any link in the network
3. strict constraints on the quality and granularity of ingress shaping

While this has been intuitively understood for a long time, our contribution is to quantify exactly how these parameters affect the delay bound. In particular, barring new results on delay bounds, our results imply that in the absence of any other constraints on the topology or routes, priority traffic utilization must be much lower than previously assumed if strong delay and jitter guarantees are required. For the typical case of leaky bucket constrained flows our delay bound is linear in the bound on the ratio $\frac{B_F}{R_F}$ across all priority flows that are individually shaped at the ingress. Hence, there is an immediate implication that the smaller this ratio, the better the bounds can be provided. While B_F depends primarily on the *accuracy* of the shaper, R_F depends on the *granularity* of shaped flows. In the case of microflow shaping, the minimal rate of the microflow can be quite small, resulting in a large delay bound. There is a substantial advantage therefore in aggregating many small microflows into an aggregate and shaping the aggregate as a whole. While in principle there is a range of choices for aggregation, there are two natural choices: edge-to-edge aggregates or edge-to-everywhere aggregates. Edge-to-edge shaping is natural for explicit bandwidth reservation techniques. In this case R_F and B_F relate to the rate and token bucket depth of the edge-to-edge flow aggregates. As discussed above, shaping the total edge-to-edge aggregates allows reducing the delay bound linearly with the degree of aggregation. Edge-to-everywhere aggregate shaping is frequently assumed in conjunction with bandwidth provisioning.

The effect of this choice on delay bounds depends on exactly how provisioning is done. One possibility for provisioning the network is to estimate the edge-to-edge demand matrix for priority traffic and ensure that there is sufficient capacity to accommodate this demand, assuming that the traffic matrix is accurate enough. Another option is to make no assumption on the edge-to-edge priority traffic matrix, but rather to admit a certain amount of priority traffic at each ingress edge, regardless of the destination, and provision the network in such way that even if *all* traffic from *all* sources happens to pass through a single bottleneck link, the capacity of that link is sufficient to ensure that priority traffic utilization does not exceed α. Depending on which of the two choices for provisioning is chosen, shaping of the edge-to-everywhere aggregate has the opposite effect on the delay bound. In the case of "edge-to-edge provisioning", the bandwidth of any link may be sufficient to accommodate the *actual* load of priority traffic while remaining within the target utilization bound. Hence, it is the minimal rate and the maximum burst size of the *actual* edge-to-edge aggregates sharing any link that effect the delay bound. However, aggregate edge-to-all shaping may result in individual substream of the shaped edge-to-everywhere aggregate being shaped to a much higher rate than the expected rate of that substream. When the edge-to-everywhere aggregate splits inside the network into different substreams going to different destinations, each of those substreams may have in the worst case substantially larger token bucket depth than that of the aggregate edge-to-everywhere stream[4]. This results in substantial increase of the worst case delay over the edge-to-edge shaping model. Moreover, in this case the properties of ingress shapers do not provide sufficient information to bound the worst case delay, since it is the burst tolerance of the *substreams* inside the shaped aggregates that is needed, but is unknown.

In contrast, if the "worst case" provisioning is assumed, the network is provisioned in such a way that each link can accommodate all the traffic even if all edge-to-everywhere aggregates end up sharing this link. In this case the shaping parameters of the edge-to-everywhere aggregate are appropriate for the delay bound. Intuitively, in this case the aggregate burst tolerance on any link can only be smaller than in the worst case [5].

[4] This is because a low-rate substream of a high-rate shaped aggregate may be shaped to a much larger rate than its own.

[5] Note that the "worst case" provisioning model targeting a particular utilization bound results in sub-

6 Summary

We have discussed the delay bounds which can be obtained in a network with aggregate priority FIFO scheduling for priority traffic. Our main result is the delay bound for an arbitrary topology which is obtained as a function of priority traffic utilization and the maximum hop count. Unfortunately, this bound is valid only for reasonably small utilization. We have argued that for larger utilization the bound either does not exist, or must depend on some other constraints on the topology or routes. Our results imply that at least for small utilization and hop count, it is possible to provide strict delay guarantees even if only aggregate scheduling is employed in the network. We have argued that the delay bound can be improved by using appropriate aggregate shaping policies at the network ingress. It remains unclear whether the use of some additional network parameters characterizing the size of the network might yield reasonable delay bounds at larger utilization for a general topology. Understanding this issue appears to be an important area of future work both from the analytical and practical standpoints.

References

[1] V. Jacobson, K. Nichols, K. Poduri. An Expedited Forwarding PHB, *RFC 2598*, June 1999

[2] F. Baker, C. Iturralde, F. Le Faucheur, B. Davie. Aggregation of RSVP for IPv4 and IPv6 Reservations, *draft-ietf-issll-rsvp-aggr-00.txt* September 1999, work in progress.

[3] F Brichet, L. Massoulie, J. Roberts Stochastic Ordering and the Notion of Negligible CDV, *ITC15*, Washington, D.C. June 1997

[4] Schenker, S., Partridge, C., Guerin, R. . Specification of Guaranteed Quality of Service, *RFC 2212* September 1997

[5] R.L. Cruz. Quality of service guarantees in virtual circuit switched networks. *IEEE JSAC*, pages 1048–1056, August 1995.

[6] J.-Y. Le Boudec. Application of network calculus to guaranteed service networks. *IEEE Transactions on Information Theory*, 441087–1096, May 1998.

[7] C.S. Chang. On deterministic traffic regulation and service guarantee A systematic approach by filtering. *IEEE Transactions on Information Theory*, 441096–1107, August 1998.

[8] R. Agrawal, R. L. Cruz, C. Okino, and R. Rajan. Performance bounds for flow control protocols. *IEEE/ACM Transactions on Networking (7) 3*, pages 310–323, June 1999.

[9] Parekh A. K. and Gallager R. G. A generalized processor sharing approach to flow control in integrated services networks The multiple node case. *IEEE/ACM Trans. Networking, vol 2-2*, pages 137–150, April 1994.

stantially more overprovisioning than the "point-to-point" provisioning using an estimated traffic matrix,or explicit point-to point bandwidth allocation using signaled admission control.

[10] M. Andrews, B. Awerbuch, A. Fernandez, J. Kleinberg, T. Leighton, and Z. Liu. Universal stability results for greedy contention resolution protocols. In *37th Annual IEEE Symposium on Foundations of Computer Science (FOCS'96), Burlington VT*, October 1996.

[11] JY Le Boudec and G. Hebuterne. Comment on a deterministic approach to the end-to-end analysis of packet flows in connection oriented network. *IEEE/ACM Transactions on Networking*, February 2000.

[12] I. Chlamtac, A. Faragó, H. Zhang, and A. Fumagalli. A deterministic approach to the end-to-end analysis of packet flows in connection oriented networks. *IEEE/ACM transactions on networking*, (6)4422–431, 08 1998.

[13] C.S. Chang. *A filtering theory for Communication Networks*. Springer Verlag, 1999.

[14] A. Charny. "Delay Bounds in a Network with Aggregate Scheduling". Draft version. ftp://ftpeng.cisco.com/ftp/acharny/aggregate_delay_v4.ps, February 2000.