Quality-of-Service in Packet Networks Basic Mechanisms and Directions

R. Guérin and V. Peris

***Abstract*— In this paper, we review the basic mechanisms used in packet networks to support Quality-of-Service (QoS) guarantees. We outline the various approaches that have been proposed, and discuss some of the trade-offs they involve. Specifically, the paper starts by introducing the different scheduling and buffer management mechanisms that can be used to provide service differentiation in packet networks. The aim is not to provide an exhaustive review of existing mechanisms, but instead to give the reader a perspective on the range of options available and the associated trade-off between performance, functionality, and complexity. This is then followed by a discussion on the use of such mechanisms to provide specific end-to-end performance guarantees. The emphasis of this second part is on the need for adapting mechanisms to the different environments where they are to be deployed. In particular, fine grain buffer management and scheduling mechanisms may be neither necessary nor cost effective in high speed backbones, where "aggregate" solutions are more appropriate. The paper discusses issues and possible approaches to allow coexistence of different mechanisms in delivering end-to-end guarantees.**

***Keywords*— QoS Mechanisms, Real-time Traffic, Scheduling, Buffer Management, Aggregation**

1 Introduction

The Internet is a continuously and rapidly evolving entity, that is being used by an increasingly large number of applications with diverse requirements. IP telephony is one of many new applications driving the need for substantial changes in the current Internet infrastructure. Specifically, the emergence of applications with very different throughput, loss, or delay requirements is calling for a network capable of supporting different levels of services, as opposed to a single, best-effort level of service which is the rule in today's Internet.

Providing different levels of service in the network, however, introduces a number of new requirements, that can be classified along two major axes: control path and data path. New data path mechanisms are the means through which different services will be enforced. They are responsible for classifying and mapping user packets to their intended service class, and controlling the amount of network resources that each service class can consume. New control path mechanisms are needed to allow the users and the network to agree on service definitions, identify which users are entitled to a given service, and let the network appropriately allocate resources to each service.

Data path mechanisms are the basic building blocks on which network QoS is built. They implement the actions that the network needs to be able to take on user packets, in order to enforce different levels of service. As a result, the paper's first goal is to provide a broad overview of the generic data path approaches that have been developed to control access to resources in packet networks.

Resources that networks need to manage in order to support service differentiation primarily include buffers and bandwidth. The corresponding mechanisms consist of buffer management schemes and scheduling algorithms, respectively. Buffer management schemes decide which packets can be stored as they wait for transmission, while scheduling mechanisms control the actual transmissions of packets. The two are obviously related, e.g., scheduling frequent packet transmissions for a given flow, i.e., allocating it more bandwidth, can help reduce its need for buffers, and conversely limiting the amount of buffer space a flow can use impacts the amount of bandwidth it is able to consume.

This paper does not attempt an exhaustive review of all possible scheduling and buffer management schemes. Instead, its aim is to identify the basic approaches that have been proposed and classify them according to the design trade-off they represent. In particular, it is important to understand how different schemes fare in terms of fairness (access to excess capacity), isolation (protection from excess traffic from other users), efficiency (number of flows that can be accommodated for a given level of service), and complexity (both in terms of implementation and control overhead).

For example, certain buffer management schemes maintain per flow buffer counts and use that information to determine if a new packet should be accommodated. Alternatively, other schemes base such decisions on more global information such as buffer content thresholds and service type indicators in packet headers. The finer granularity of per flow information has benefits in terms of fairness and efficiency, but comes at the cost of greater complexity. Scheduling algorithms face similar trade-offs. For example, schedulers based on the weighted fair queueing algorithm [14, 47] can be used to give rate and delay guarantees to individual flows, while class based mechanisms, e.g., CBQ [19], provide aggregated service guarantees to the set of flows mapped into the same class. Here, the trade-off is again between fairness and efficiency, and complexity.

In Sections 2 and 3, we illustrate through a selected set of scheduling and buffer management schemes, the many alternatives available to network designers to control how

R. Guérin and V. Peris are with IBM, T.J. Watson Research Center, P.O. Box 704, Yorktown Heights, NY 10598, USA.

user packets access network resources. As mentioned above, the emphasis is just as much on reviewing the basic methods that have been developed as on highlighting trade-offs.

Trade-offs are not the prerogative of data path mechanisms, and very similar issues exist on the control path. This applies to both the type of services available and how they can be requested. For example, services can vary greatly in the type of guarantees they provide. Service guarantees can be quantitative and range from simple throughput guarantees to hard bounds on end-to-end delay and lossless transmissions. Alternatively, some proposed service guarantees are qualitative in nature, and only stipulate "better" treatment for some packets, e.g., they'll go through a higher priority queue and, therefore, experience lower delay, or will be given a lower discard threshold in case of congestion. Similarly, requests for service can be dynamic and extend all the way to applications, e.g., using the RSVP protocol [6], or may be handled only through static configuration information that defines bandwidth provisioning between specific subnets.

In general, control path trade-offs are along two dimensions: the processing required to support the service, and the amount of information that needs to be maintained. For example, the complexity of a call admission decision to determine if a new request can be accommodated, depends on both the service definition itself and how it maps onto the underlying scheduling and buffer management mechanisms. Similarly, a service such as the Guaranteed Service [54] involves a number of parameters used to compute the end-to-end delay bound and buffer size that the service mandates. Examples illustrating some of these issues and the associated trade-off are briefly reviewed in Section 4, that also describes and contrasts the two services currently being standardized [54, 60] to provide QoS in IP networks.

In general, which trade-off is desirable or even feasible is a function of the scalability requirements of each environment, i.e., the total amount of information and processing that can be accommodated. This is a decision that involves **both** the data path and the control path. For example, per flow scheduling, buffer management, and state maintenance and processing may be appropriate in a local campus, but can create scalability problems in the core of a backbone where the number of individual flows will be orders of magnitude larger. In such an environment, scalability is of prime concern and while per flow service mechanisms remain desirable, approaches that rely on service aggregation may be needed. We review such issues in Section 5, where we highlight some of the approaches that have been proposed so far and point to some of the challenges being faced. In particular, we discuss the need for interoperability between fine grain and coarse grain solutions, and outline possible solutions.

2 Scheduling Mechanisms

The nature of the scheduling mechanism employed on network links greatly impacts the QoS guarantees that can be provided by the network. The basic function of the scheduler is to arbitrate between the packets that are ready for transmission on the link. Based on the algorithm used for scheduling packets, as well as the traffic characteristics of the flows multiplexed on the link, certain performance measures can be computed. These can then be used by the network to provide end-to-end QoS guarantees. First, we describe how the user traffic is characterized at the ingress into the network.

2.1 Traffic Specification

Both ATM and IP networks are similar in their characterization of user traffic. Rather than describe both we concentrate on IP networks, where the specification of user traffic is currently done through a TSpec [55], which consists of the following parameters[1]: a token bucket plus a peak rate (p), a minimum policed unit (m), and a maximum datagram size (M). The token bucket has a bucket depth, b, and a bucket rate, r. The token bucket, the peak rate, and the maximum datagram size, together define the *conformance* test that identifies the user packets eligible for service guarantees. The conformance test defines the maximum amount of traffic that the user can inject in the network, and for which it can expect to receive the service guarantees it has contracted. This maximum amount of traffic is expressed in terms of a traffic envelope $A(t)$, that specifies an upper bound on the amount of traffic generated in any time interval of duration t:

$$A(t) \leq \min(M + pt, b + rt) \qquad (1)$$

Equation (1) simply states that the user can send up to b bytes of data at its full peak rate of p, but must then lower its rate down to r. The presence of the factor M in the first term is because of the packet nature of transmissions, where up to one maximum size packet worth of data can be sent at link speed. The above model is essentially that of a traditional *"leaky bucket"*, that lets users specify a long term transmission rate (r) while preserving their ability to burst at a higher speed (p) for limited periods of time (up to $b/(p-r)$). The full TSpec further includes a minimum policed unit m, which counts any packet of size less than m as being of size m. This allows for accounting of per packet processing or transmission overhead, e.g., time needed to schedule packet transmissions.

2.2 Scheduling policies

We now turn our attention to the different scheduling policies that can be used to select packets for transmission on a link. There are several aspects that are to be considered when choosing a scheduling algorithm. Some of the main ones are

- The nature of the performance measures guaranteed by the scheduling algorithm. For instance, does it enable the network to provide a per-flow rate guarantee,

[1] Rates are in units of bytes/sec, while packet sizes and bucket depth are in bytes.

or does it provide a simple priority structure so that one class of traffic receives priority over another class?

- The efficiency of the algorithm in enforcing service guarantees. Basically, how many calls with given characteristics can be packed on the link. This is often measured in terms of call admission or schedulability region of the algorithm.
- The complexity of the algorithm. Given the high link speeds that are in use today and the even higher ones waiting in the wings, the amount of time available to schedule a single packet is continuously decreasing. Therefore, it is very important for the algorithm to require a small number of operations per packet.
- The basic mechanism and parameters used by the algorithm to make scheduling decisions. Of specific interest is whether the algorithm supports separate delay and rate guarantees, e.g., [11, 24, 62], or combines the two e.g., [29, 46, 58]. Separate rate and delay guarantees allow greater flexibility.
- The flexibility of the algorithm in handling traffic in excess of the amount for which the service guarantees have been requested. Some algorithms easily handle excess traffic while others require additional mechanisms. The flexibility of an algorithm in handling excess traffic is often described in terms of a *fairness* index, that attempts to measure how equally the algorithm redistribute available resources across flows (see [3, 4, 28] for discussions on this issue).

We try to cover each of these aspects as we review different scheduling policies.

2.3 First Come First Served

This is one of the simplest scheduling policies whose operation is as its name suggests, i.e., packets are served in the order in which they are received. While this may seem like a fairly poor way of providing differentiated service, it is quite simple to implement. In particular, insertion and deletion from the queue of waiting packets is a constant time operation and does not require any per-flow state to be maintained. Not surprisingly, the First Come First Served (FCFS) policy is one of the most commonly implemented scheduling policies.

The FCFS policy does not lend itself readily to providing delay or rate guarantees. One way to provide a delay bound is to limit the size of the queue of waiting packets. That way, once a packet is queued up for transmission it is guaranteed to be sent out in the time it takes to serve a full queue of packets. However, packets that arrive when the queue is full have to be dropped. To ensure that the drop probability is below a certain threshold only a limited number of flows should be accepted. A simple mechanism for performing this decision is to compute an *effective bandwidth* that quantifies the amount of link capacity that is required for each flow [25, 31, 40].

Given that the FCFS policy is one of the least sophisticated in its operation, it does not explicitly provide any mechanism for fair sharing of link resources. However, with the help of some buffer management mechanisms it is possible to control the sharing of bandwidth and we describe some of these mechanisms in Section 3.

2.4 Fixed Priority Scheduler

This policy offers some ability to provide different grades of service with a rather coarse granularity. Traffic is classified as belonging to one of a fixed number of static priorities. The link multiplexer maintains a separate queue for each priority and serves a packet in a lower priority queue only if all the higher priority queues are empty. Each queue is served in a FCFS manner and so this scheduling mechanism is almost as simple as the FCFS policy with the added complexity of having to maintain a few queues. Selecting a packet for transmission is a fixed cost operation that only depends on the number of priority levels and is independent of the number of flows that are being multiplexed.

While the priority scheduler does offer a certain amount of service differentiation capability, it still does not readily allow end-to-end performance guarantees to be provided on a per-flow basis. Rather, it only provides for one class of traffic to receive better service than another class of traffic at a single link. As with FCFS, the provision of end-to-end delay guarantees with a fixed priority scheduler can achieved by limiting buffer sizes. However, one important difference, that has to be accounted for, is the fact that a lower priority packet will be served only after all the packets from the higher priority queues are transmitted. This is bound to affect the variability in the delay that is observed by the lower priority packets. In general both the FCFS and fixed priority schedulers are not ideal candidates when it comes to providing guaranteed end-to-end delay bounds.

Another problem with the priority scheduler is that different switches may have different levels of priority and matching these levels from an end-to-end perspective is no easy feat. In Section 5 we touch upon some aspects of priority levels in the context of the Type of Service (TOS) bit semantics, which are currently being discussed at the IETF.

2.5 Fair Queueing schedulers

The Weighted Fair Queueing (WFQ) service discipline and its many variants have been very popular in the current decade. Part of its popularity stems from the fact that it overcomes some of the limitations of the FCFS and priority schedulers by allowing for a fine grain control over the service received by individual flows. This allows the network to provide end-to-end delay guarantees on a per-flow basis. In addition, WFQ serves excess traffic in a fair manner, where fairness is measured relative to the amount of resources that are reserved for each flow [26].

Most variants of the WFQ discipline are compared to the Generalized Processor Sharing (GPS) scheduler which is a theoretical construct based on a form of processor sharing [47]. The operation of a GPS scheduler can be elegantly defined for a fluid model of traffic and tight end-to-end delay bounds can be computed for a flow that traverses multiple links which are all served by GPS schedulers. The basic idea behind GPS is as follows: a weight ϕ_i is associated with flow $i, i = 1, \ldots, N$, and the link capacity is shared among the active flows in direct proportion to their weights. In other words, if γ were to denote the link speed, then flow i is guaranteed to obtain a minimum service rate of $\frac{\phi_i}{\sum_{j=1}^{N} \phi_j}\gamma$, $i = 1, \ldots, N$. However, at any given time it is quite likely that some flows do not have a backlog of traffic waiting to be transmitted on the link. This will translate into some unutilized link capacity that can be shared among the back-logged (active) flows. The GPS scheduler shares this excess capacity among the back-logged flows in proportion to their respective weights. If $B(t)$ denotes the set of flows that are back-logged at time $t \geq 0$, then flow i is guaranteed to receive a minimum service rate of $r_i(t)$ given by

$$r_i(t) = \begin{cases} \frac{\phi_i}{\sum_{j \in B(t)} \phi_j}\gamma & i \in B(t), \\ 0 & \text{otherwise.} \end{cases}$$

In reality we don't have fluid flows, so the main use of GPS is as a reference model. By simulating this reference model it is possible to define service disciplines that closely follow the GPS policy. This is precisely the way in which the Packetized Generalized Processor Sharing (PGPS) [47] – also referred to as Weighted Fair Queueing (WFQ) [14] – is defined. A packet arrival in the real system is modeled as the arrival of a certain volume of fluid in the reference system. A packet is considered to be transmitted in the reference system when the fluid volume corresponding to that packet has been completely served by the reference GPS scheduler. The PGPS policy simulates the reference GPS scheduler using the fluid arrival process as described above and computes the departure times of the packets in the reference system. The PGPS scheduler then selects for transmission, the packet that would have departed the earliest in the reference GPS system.

One of the reasons for the popularity of the GPS policy is its fair handling of excess traffic. If two flows are back-logged during any interval of time they receive service in direct proportion to their weights regardless of the amount of excess traffic that any of them may be generating. Let $W_i(t_1, t_2)$ denote the amount of flow i traffic served in the interval (t_1, t_2), $i = 1, \ldots, N$. The GPS policy ensures that for any two flows i, j back-logged during the interval (t_1, t_2), the following relation holds:

$$\frac{W_i(t_1, t_2)}{\phi_i} = \frac{W_j(t_1, t_2)}{\phi_j}$$

Indeed, it is this very relation that is one of the measures used to quantify fairness among the many approximations of GPS. A bound on $|\frac{W_i(t_1,t_2)}{\phi_i} - \frac{W_j(t_1,t_2)}{\phi_j}|$ for any two flows that are back-logged in the interval (t_1, t_2) can be used to compare the relative fairness of different scheduling policies [26]. The GPS scheduler has a fairness measure of 0.

Most schedulers that are based on GPS can be implemented using the notion of a virtual time function. The virtual time is relevant only during a busy period which is defined as a maximal interval of time during which the server is not idle. For GPS the virtual time, $V(t)$, at the start of a busy period is defined to be 0. During any busy period $V(t)$ advances at the rate of

$$\frac{\partial V}{\partial t} = \frac{\gamma}{\sum_{i \in B(t)} \phi_i}.$$

Based on this virtual time, one can define for each packet of a flow, say, the kth packet of flow i, its virtual start time, s_i^k, and virtual finish time, f_i^k, that correspond to its start and finish times, respectively, in the GPS reference system. If we further denote its arrival time by a_i^k, and its length by l_i^k, we then have the following relations,

$$\begin{aligned} f_i^0 &= 0 \\ s_i^k &= \max\left\{V(a_i^k), f_i^{k-1}\right\} \quad k = 1, \ldots \\ f_i^k &= s_i^k + \frac{l_i^k}{\phi_i} \quad k = 1, \ldots \end{aligned}$$

The previously mentioned simulation of GPS on which PGPS relies, is based on this notion of virtual time. It is implemented as follows: the scheduler always picks the packet with the smallest virtual finish time for transmission on the link (ties can be arbitrarily broken). This can be efficiently performed using a priority queue based on the finish times of the packets. The complexity of the insertion and deletion operations from this queue is $O(\log N)$. Note however, that there is additional complexity in computing the advancement of virtual time since $B(t)$ is not likely to remain the same during the entire busy period. In the worst case the virtual time computation can be $O(N)$ as shown in [26].

Another advantage of the PGPS policy is that an end-to-end delay bound can be computed based on the weight assigned to the flow. Let γ^h, $h = 1, \ldots, H$ denote the speed of the links traversed by flow i. For simplicity, assume that the traffic envelope for flow i is given by $A_i(t) = b_i + r_i t$, $t \geq 0$, and let R_i be the minimum rate guaranteed to flow i by each of the links that it traverses. Assuming that the stability condition, $R_i \geq r_i$, holds, the end-to-end delay guarantee for flow i is given by:

$$\hat{D}_i = \frac{b_i}{R_i} + \frac{(H-1)M_i}{R_i} + \sum_{h=1}^{H} \frac{L_{\max}^h}{\gamma^h}, \qquad (2)$$

where M_i denotes the maximum packet length for flow i, and $L_{\max}^h$ denotes the maximum packet length at link h. Notice how the $\frac{b_i}{R_i}$ term appears only once regardless of the number of hops that are traversed. The reason for this, is that once a bottleneck link is encountered, the PGPS scheduler effectively smooths out the burst in the flow, so that the burst delay $(\frac{b_i}{R_i})$ is no longer encountered downstream. Also notice how the end-to-end delay bound is independent of the number of other connections that are multiplexed at each of the links traversed by flow i. This property makes the PGPS policy one of the best in terms of providing tight end-to-end delay guarantees.

2.6 Variants of Fair Queueing

One of the main drawbacks of PGPS is the complexity involved in computing the virtual time function. Recently, several scheduling policies with simpler computations of virtual time have been proposed. These policies can guarantee end-to-end delay bounds that have a form similar to that of equation (2), and we briefly highlight some of them next.

Self Clocked Fair Queueing (SCFQ) was proposed in [26] as a simpler alternative to PGPS. Rather than use the GPS reference model for a computation of the virtual time, it based the evolution of $V(t)$ on the virtual start time of the packet currently in service. This greatly reduced the amount of computation needed to keep track of the virtual time. This reduction in complexity results in a larger end-to-end delay bound than PGPS. Roughly, it adds a delay term of the form $\sum_{i=1}^{N} \frac{M_i}{\gamma^h}$ at each of the links that are traversed [27]. The fact that the end-to-end delay now depends on the number of flows that are multiplexed on the link is the main drawback of this policy.

Start-time Fair Queueing (SFQ) was proposed in [28] as another Fair Queueing policy. It is very similar to SCFQ with the main difference being that the SFQ scheduler picks that packet with the smallest virtual start time (as opposed to the smallest virtual *finish* time) for transmission on the link. The end-to-end delay of SFQ is very close to SCFQ, but is slightly smaller.

While PGPS tracks the GPS policy it only uses the finish times of the packets in the reference system to select the next packet for transmission. Thus packets which have not yet started service in the reference system may be selected for service simply because their finish times are earlier than all the other packets waiting to be transmitted. This can result in a significant difference between packet departures in the PGPS and the reference system, with packets departing much earlier in PGPS than in the reference system. This discrepancy can be measured using the *worst-case fairness index* as defined in [4]. The Worst-case Fair Weighted Fair Queueing ($\mathrm{WF}^2\mathrm{Q}$) scheduler proposed in [4] uses both the start and finish times of packets in the reference GPS system to achieve a more accurate emulation of GPS. The $\mathrm{WF}^2\mathrm{Q}$ policy selects the packet with the smallest virtual finish time in the reference system provided that its virtual start time is less than the current time. This prevents it from running too far ahead of the reference system. The end-to-end delay bound for a $\mathrm{WF}^2\mathrm{Q}$ scheduler is identical to that of PGPS.

There are a few more scheduling policies that also seek to emulate Fair Queueing. The interested reader is referred to [56] for a description of Deficit Round Robin, which extends Weighted Round Robin [39] by taking into account variable sized packets. Frame-based Fair Queueing[57] is another scheduling policy that provides the same end-to-end delay guarantees as PGPS but is simpler to implement.

2.7 Earliest Deadline First

The Earliest Deadline First (EDF) scheduler is a form of a dynamic priority scheduler where the priorities for each packet are assigned as it arrives. Specifically, a deadline is assigned to each packet which is given by the sum of its arrival time and the delay guarantee associated with the flow that the packet belongs to. The EDF scheduler selects the packet with the smallest deadline for transmission on the link and hence the name. The dynamic nature of the priority in the EDF scheduler is evident from the fact that the priority of the packet increases with the amount of time it spends in the system. This ensures that packets with loose delay requirements obtain better service than they would in a static priority scheduler, without sacrificing the tight delay guarantees that may be provided to other flows. It is well known that for any packet arrival process where a deadline can be associated with each packet, the EDF policy is optimal in terms of minimizing the maximum lateness of packets [22]. Here, lateness is defined as the difference between the deadline of a packet and the time it is actually transmitted on the link.

As mentioned earlier, the GPS policy guarantees a delay bound on a per-flow basis based on the weight that is assigned to the flow. These weights are closely coupled to a reserved rate, and for a flow to receive a small end-to-end delay guarantee it is necessary that it be allocated a relatively large rate. This can lead to an inefficient use of resources, particularly if a low bandwidth flow requires tight end-to-end delay guarantees. One of the main attractions of the EDF policy is that it allows for the separation of delay and throughput guarantees for a flow.

In terms of implementation the EDF policy is more complex than the FCFS or the static priority scheduler. The complexity arises because the scheduler has to pick the packet with the smallest deadline for transmission on the link. This involves keeping a priority list of deadlines and so the insertion or deletion from this list has a complexity of $O(\log K)$ operations, where K corresponds to the number of packets awaiting transmission. An obvious optimization is to only keep a single packet from each of the flows in the priority list of deadlines, since packets belonging to the same flow must leave in the order in which they arrive. As a result, the complexity is reduced to $O(\log N)$, where N is the number of flows multiplexed onto the link. Assuming that each of these flows are characterized by a traffic envelope denoted by $A_i(t),\ i = 1, \ldots, N$, it is known [22] that the EDF scheduler can provide a delay guarantee of D_i to flow i, $i = 1, \ldots, N$, provided the following feasibility condition is satisfied:

$$\sum_{i=1}^{N} A_i(\tau - D_i) + L_{\max} \leq \gamma\tau, \quad \tau \geq 0, \tag{3}$$

where $A_i(t) := 0$ for $t < 0$, and $L_{\max}$ denotes the maximum transmission unit for the link.

2.7.1 Rate Controlled Service discipline

The EDF policy by itself cannot be used to provide efficient end-to-end delay guarantees [24]. This is because equation (3) only provides a feasibility check for the delay guarantees at a single link. To compute the feasibility check at a link that is downstream we need to know the traffic characterization at that link. The very act of scheduling perturbs the flow and so the traffic characterization of the flow is not likely to remain the same as it traverses the network. It is possible to compute a bound on the traffic characterization of a flow at the link output and use that to compute a feasible delay guarantee for the flow at the downstream link. However this approach does not result in efficient end-to-end delay guarantees since the bound on the traffic characterization at the output link can be rather pessimistic [24]. Alternatively, one could reshape the traffic at each node to a pre-specified envelope before it is made eligible for scheduling. Coupled with traffic shapers the EDF policy can be used to provide efficient end-to-end delay guarantees on a per-flow basis. We refer to this combination as a Rate-Controlled Service (RCS) discipline [24, 62] and briefly discuss some of the implications of the RCS discipline.

The basic idea of the RCS discipline is to reshape traffic from a flow at every link that it traverses. The flow need not be reshaped to its original specification, in fact this turns out to be a particularly bad idea [24]. With the appropriate choice of reshapers it can be shown that the RCS discipline is the most efficient in terms of guaranteeing end-to-end delays of all the scheduling policies that are known today. In [24], it is shown that with an appropriate choice of shaper envelopes the RCS discipline can provide the same delay guarantees as PGPS and in addition accept some more flows. As mentioned at the beginning of this section, the efficiency of scheduling algorithms can be measured in terms of the number of flows they can support with given delay guarantees. This corresponds to the notion of a *schedulable region*, and it can be shown [24] that the RCS discipline has the largest schedulable region among all scheduling policies known today. An example illustrating the schedulable region of different scheduling policies can be found in [48, section 4.3.5].

2.8 Hierarchical Link Sharing

As discussed in Section 4, a network may offer several different services over a single link. The link will, therefore, have to be partitioned to support the different service classes. Alternatively, a single link in the network may be shared by several different organizations or departments within an organization. Each of these may want to receive a guaranteed portion of the link capacity but are willing to allow other departments or organizations to borrow unutilized link resources. The hierarchical structure of organizations suggests a hierarchical partitioning of the link resources. A hierarchical link sharing structure consisting of classes that correspond to some aggregation of traffic is suggested in [19] and is often referred to as Class Based Queueing (CBQ). Each class is associated with a link-sharing bandwidth and one of the goals of CBQ is to roughly guarantee this bandwidth to the traffic belonging to the class. Excess bandwidth should be shared in a fair manner among the other classes. There is no requirement to use the same scheduling policy at all levels of a link sharing hierarchy, and it is conceivable that classes carrying interactive traffic may benefit from a simple priority scheduling policy as opposed to rate-based schedulers (see [19] for details).

This being said, the GPS policy can be used in a hierarchical manner to provide both link sharing and individual QoS guarantees to flows. At the top-level of the hierarchy, the weights reflect the link sharing requirements, while the lower level weights are used to provide individual QoS guarantees. Whenever an interior node receives service it is distributed among its child nodes in direct proportion to their weights. Note that there may be more than two levels in the hierarchy. Except for leaf nodes, other nodes only receive a logical service. The interested reader is referred to [3] for a detailed description of hierarchical packet fair queueing algorithms.

3 Buffer Management

While the scheduling policy does play a big role in the QoS provided by the network, it is only effective if there are sufficient buffers to hold incoming packets. Since link speeds are currently in the order of Gigabits per second, the amount of memory required to buffer traffic during transient periods of congestion can be large and exceed the amount of memory that most routers and switches have. Packets that arrive during these transient periods will have to be dropped. If the end-to-end transport protocol is a reliable one like TCP, then a lost packet causes a retransmission as well as a reduction in the size of the congestion window. The net result will be a lower end-to-end QoS for the flow concerned.

In order to avoid a haphazard behavior when a link experiences congestion, several different buffer management schemes have been proposed. These can be roughly classified along the following two dimensions:

1. When are packet discard decisions made? Typically, packet discard decisions are made either upon the arrival of a new packet, or at the onset of congestion where currently stored packets may then be discarded to accommodate a new, higher priority packet.
2. What information is used to make packet discard decisions? The main aspect is the granularity of the information, i.e., is per flow buffer accounting done and used to discard packets from individual flows, or is only global, per class, information kept and used.

Different designs correspond to different trade-offs between performance and complexity along these two dimensions, where the performance of a buffer management scheme is measured in terms of its ability to control traffic fairly and efficiently during periods of congestion. Efficient treatment means that the buffer management mechanism should avoid violation of a service agreement by losing many "high priority" (conformant) packets during periods of congestion. For example, this can happen if too many low priority packets are allowed in, and as a result some packets from an arriving high priority burst are lost. One solution is to allow the high priority packets to "push-out" [42] the low priority ones, but this capability adds to the implementation complexity of the scheme. In addition, it may not always be sufficient (see [7] for a discussion of this issue) and additional mechanisms may be needed to try to avoid congestion. In this section, we describe a few such buffer management mechanisms like Early Packet Discard (EPD) [51, 59] and Random Early Discard (RED) [18], where, as a preventive measure, packets are discarded before the onset of congestion.

Detecting the onset of congestion and the associated preventive measures vary with the link technology being used. For example, on ATM links an IP packet may be segmented into smaller units, e.g. 53 byte cells. In this case, congestion is measured with respect to the buffer's ability to store cells. A single cell loss will result in an entire packet being rendered useless, so that it is advantageous to discard an entire packet's worth of cells at a time. One obvious approach is to drop all subsequent cells from a packet if there is no room in the buffer for an arriving cell. This is termed Packet Tail Discarding in [59] or Partial Packet Discard (PPD) in [51]. While such a technique helps avoid unnecessary buffer waste, substantial *packet* throughput degradation still occurs during periods of very high congestion, e.g., offered traffic above twice the link capacity. This is because in such cases, almost every packet will lose at least one cell, and flows with large packets will be even further penalized.

A possible approach to remedy this last problem, is to predict the onset of congestion and discard all cells of packets that are expected to experience a cell loss. For example, this can be achieved using a simple threshold on the buffer occupancy to decide whether to accept or drop a packet at the time its first cell is received. This is referred to as Early Packet Discard (EPD) [51] and has the benefit that link capacity is not wasted in the transmission of partial packets. EPD performs better than PPD and results in an effective throughput of close to the link capacity with relatively fewer buffers [51, 59].

However, it turns out that EPD can be unfair to low bandwidth flows. In particular, if the buffer occupancy is hovering around the threshold value, then high rate flows are likely to have more opportunities at getting their packet accepted (assuming identical packet sizes). Once a packet is accepted it pushes the buffer occupancy above the threshold and a subsequent packet from a low bandwidth flow is then likely to be dropped. Some of these issues are discussed in [59], which presents a number of enhancements to the basic EPD scheme aimed at improving not only goodput, i.e., the number of complete packets transmitted, but also fairness in selecting flows that can start buffering a new packet.

In general, it is desirable to devise buffer management schemes, that preserve the goodput benefits of EPD while also ensuring fairness in how this goodput is distributed across flows. The fairness of a buffer management scheme is a function of how it penalizes packets from non-conformant flows, i.e., flows sending at a higher rate than they are entitled to. Ideally, the scheme should ensure that no single flow can grab a disproportionate amount of the buffer space, thereby affecting the performance level of other flows. If per flow buffer accounting is performed, it is relatively easy to identify misbehaving flows and take appropriate actions to ensure a fair allocation of the buffer space. However, there is a cost associated with per flow buffer accounting, and in some environments where scalability is a concern, buffer management may need to be done at a coarser level. The penalty is that it is harder to ensure fairness when many flows share a common buffer. In some instances, i.e., with adaptive applications such as TCP, it is, however, possible to ensure some level of per flow fairness without maintaining per flow state. This was one of the goals of the Random Early Drop (RED) [18] mechanism, which relies on *random* dropping decisions when the buffer content exceeds a given threshold, so that heavy flows experience a larger number of dropped packets in case of congestion. Hence RED aims at penalizing flows in proportion to the amount of traffic they contribute, and therefore preventing any of them from grabbing a disproportionate amount of resources.

Specifically, assuming that flows use TCP as their transport protocol, RED provides a feedback mechanism so that sources can be notified of congestion in the router and accordingly reduce the amount of traffic that they inject into the network. First the average queue size is computed using a low-pass filter with an exponentially weighted moving average, so that the router does not react too quickly to transient phenomena. Next, this queue size is compared to a *maximum* and *minimum* threshold, often denoted by max_{th} and min_{th}. If the average queue size is below min_{th} or above max_{th}, all packets are accepted or dropped respectively[2]. When the average queue size is between max_{th} and min_{th}, an arriving packet is dropped with a probability that depends on the average queue size.

The probability that a packet from a particular flow is dropped is roughly proportional to the flow's share of the link bandwidth. Thus a flow that is utilizing a larger share of the link bandwidth is forced to reduce its rate rather quickly. One of the main advantages of RED is that it does not require per flow state to be maintained in the router. As a result, RED is relatively simple to implement and can be used in conjunction with the simple FCFS scheduler described in Section 2 to reduce congestion in the network. This can be of significance in the Internet backbone, where there may be hundreds of thousands of flows on a given link.

While RED does not discriminate against particular types of flows, it should be noted that it does not always ensure

[2]Packets can be alternatively marked, instead of dropping them, if packet marking is supported by the network and the end-stations

all flows a fair share of bandwidth. In fact, it turns out that RED is unfair to low speed TCP flows. This is because RED randomly drops packets when the maximum threshold is crossed and it is possible that one of these packets belongs to a flow that is currently using less than its fair share of bandwidth. Since TCP reacts rather strongly to packet loss, the lost packet will force further reduction in the congestion window resulting in an even lower rate. The Fair Random Early Drop (FRED) mechanism is presented in [44] as a modification to RED in order to reduce some of its unfairness. In addition to the RED thresholds, FRED maintains thresholds for individual flows as well as the current buffer occupancies for each of the active flows. This is used to decide which flows are using a larger amount of bandwidth than they are entitled to, so that a selective discard mechanism can be effected. This per-flow state also allows FRED to detect misbehaving flows, and it can use this information to prevent them from consistently monopolizing the buffers.

In some environments, notably ATM, it is quite natural to maintain state information for each flow or virtual circuit. Also, the fixed cell size for ATM facilitates hardware implementation of complex scheduling and buffer management algorithms. For ATM, there are many single chip solutions that perform per-flow scheduling and buffer management, e.g., IBM's CHARM chip [13]. Here each virtual circuit can be guaranteed to receive a certain rate regardless of the other flows that are multiplexed on the link. However, as mentioned earlier, this rate guarantee is good only if there is sufficient memory to buffer incoming cells. One might argue for the buffers to be strictly partitioned among each of the virtual circuits but this is not necessarily a good strategy. In [38] it is shown that a complete partitioning of the buffers results in a higher packet loss probability than when the buffers are shared among all the flows. Of course, complete sharing comes with its own problems, particularly when a diverse set of flows are to be multiplexed on the link. A good compromise is to guarantee a few buffers to each individual flow, while retaining a certain fraction of the total buffer space for sharing across the different flows [38]. As a matter of fact, the relationship between buffer allocation and rate guarantee can be made precise. In [33], it is shown that even with an FCFS scheduler, rate guarantees can be provided by appropriately allocating buffers to flows. In particular, [33] establishes the intuitive result that rate guarantees are in proportion to buffer allocation.

Overall, the buffer management schemes currently being deployed should ensure that service guarantees of applications are met, i.e., packets that conform to a service contract will rarely be dropped. However, substantial differences may exist in how excess traffic is being supported. In particular, networks that do not differentiate between adaptive and non-adaptive applications and do not support per flow buffer accounting, are likely to provide only relatively poor (unfair) performance to excess traffic.

4 Service Definitions and Associated Requirements

In this section, we review the two services that have been standardized by the IETF Integrated Service Working Group, the Controlled Load (CL) service [60] and the Guaranteed Service (GS) [54]. We also discuss briefly a recent IETF effort to introduce less rigorous service specifications than those of CL and GS, to facilitate support for service aggregation. We expand further on this last issue in Section 5.

4.1 Controlled Load Service

The definition of the Controlled Load service is qualitative in nature. It aims at approximating the service a user would experience from an *unloaded* network, where unloaded is not meant to indicate the absence of any other traffic, but instead the avoidance of conditions of heavy load or congestion. Of significance is the fact that while the Controlled Load service specification is qualitative in nature, it does require a quantitative user traffic specification in the form of the TSpec of Section 2.1. This is of importance as this represents a key input to the *call admission* process, that is required to limit the number of flows the network is willing to accept and, therefore, ensure the desired "unloaded" network behavior.

The guarantees that the Controlled Load service provides are essentially a guaranteed sustained transmission rate equal to the token bucket rate r, and the possibility to send occasional bursts (of size limited by the token bucket depth b) at its peak rate p. There are no explicit delay or even loss guarantees associated with the service, and this allows for some flexibility in the call admission process and the associated scheduling and buffer management mechanisms.

For example, Controlled Load flows on a given network link can be supported through a basic FCFS scheduler whose load is controlled using an effective bandwidth based call admission control, e.g., [25, 31, 40], together with a simple threshold based buffer management scheme. The benefits of such an approach are its simplicity in terms of both data path and control path. Neither the scheduling nor the buffer management require awareness of individual flows, and the call admission process only involves additions and subtractions as flows come and go. Further efficiency in the call admission control can be achieved, at the cost of some added complexity, by using better models for computing the effective bandwidth of a flow based on its TSpec as suggested in [34, 35], or by using measurement based call admission control techniques as described in [30, 37].

It should be noted that the approach just outlined assumes that excess traffic from individual flows can be readily identified, e.g., using explicit marking as suggested in [9]. If it is not possible to explicitly mark and, therefore, easily identify excess traffic from individual flows, more complex data path mechanisms may be required to properly support the Controlled Load service. This is required to ensure that excess traffic from one flow does not impact the service guarantees of other flows. One possible alternative is to rely on per flow buffer accounting mechanisms as described in Section 3. The tighter control of the amount of buffer space each flow can occupy provides the necessary protection against excess traffic, but comes at the cost of maintaining per flow buffer state.

A further refinement to per flow buffer state is to replace the FCFS scheduler by one of the schedulers of Section 2, that is capable of providing rate guarantees to individual flows. This can be a Weighted Fair Queueing (WFQ) scheduler [14, 45, 46, 47], or even a Self Clocked Fair Queueing scheduler (SCFQ) [26, 27] since strict delay guarantees are not critical for Controlled Load flows. This increases the amount of per flow state by adding transmission state to buffer state, but offers a number of additional benefits.

It improves the accuracy of the buffer management mechanism by ensuring that packets from each flow get regularly transmitted, so that any increase in a flow's buffer occupancy can be readily associated with a traffic increase for the flow. In contrast, a FCFS scheduler provides a much less regular service to individual flows, e.g., a flow whose packets are all at the back of the queue must wait until all packets ahead of them have been sent. As a result, it is more difficult to determine if variations in the buffer occupancy of a flow are caused by an increase in traffic or are due to service fluctuations.

In general, the relatively loose definition of the guarantees provided by the Controlled Load service allows a relatively wide range of implementation trade-offs. In particular, most of the mechanisms described in Sections 2 and 3 can be used to build a system that abides by the Controlled Load specifications [60, Section 10]. As we shall see in the next section, this is less so for the Guaranteed Service because of its stricter service definition.

4.2 Guaranteed Service

The Guaranteed Service shares with the Controlled Load service the use of a TSpec to specify the user traffic to which the service guarantees apply, but this is where any resemblance stops. While the Controlled Load service offers qualitative service guarantees, the Guaranteed Service gives not only quantitative but also hard (deterministic) service guarantees. Those guarantees include, for conformant packets, lossless[3] transmission and an upper bound on their end-to-end delay. The goal of the service is to emulate, over a packet-switched network, the guarantees provided by a dedicated rate circuit. Clearly, such guarantees are not warranted for all applications because of their cost, but they are important for applications with hard real-time requirements, e.g., remote process control, tele-medicine, haptic rendering in distributed CAVEs [20, Part II, Chapter 6], etc.

Requesting the Guaranteed Service is carried out using a one pass with advertisement (OPWA) approach as supported by the RSVP [6] protocol. The process starts with the sender specifying its traffic in the form of a TSpec, which is then sent in an associated signalling message, e.g., using an RSVP PATH message, towards the intended receiver(s). As the message propagates towards the receiver(s), it is updated by each node on the path so that the characteristics of the path are recorded [55] and, therefore, available to the receiver once the message reaches it. Characteristics of particular interest to the Guaranteed Service are captured in an *ADSPEC*, that is used to record the delay "contribution" of the scheduler used at each node. This contribution is in the form of two *"error terms,"* C and D, that account for different aspects of a scheduler's behavior and whose impact is *additive* along the path of the flow. In other words, the cumulative impact of the schedulers of all the nodes on path $\mathcal{P}$ is itself represented by two error terms C_{tot} and D_{tot} of the form:

$$C_{tot} = \sum_{i \in \mathcal{P}} C_i, \qquad D_{tot} = \sum_{i \in \mathcal{P}} D_i$$

The Guaranteed Service is built around the model of *rate-based* schedulers as first investigated in [47] and discussed in Section 2. Rate based schedulers attempt to approximate a perfect fluid server, that guarantees delay by ensuring the user a minimal service rate at **any** time, i.e., the GPS model of Section 2.5. Under such a model, a scheduler can be represented by how it deviates from the behavior of this perfect fluid server. The error terms C and D of the Guaranteed Service are used to capture those deviations (see [47, 54] for details).

The behavior of a Guaranteed Services scheduler can be accurately described using the notion of *service curves* [11, 12]. Analogous to the way a traffic envelope bounds the amount of input traffic, a service curve can be used to provide a lower bound on the amount of service received by a flow at an individual network element (see [12, 43, 53] for definitions of service curves). The Guaranteed Services scheduler can be represented as having a service curve of the rate-latency form [43], with a latency of $\frac{C}{R} + D$, for all values of the rate R. The end-to-end service curve for a given flow is computed using a convolution of the service curves at each of the network elements traversed by the flow. An advantage of the service curve methodology is that it enables a simple graphical representation of the worst-case delay and buffer requirements. For example, as shown in Figure 1, the worst-case delay and buffer requirements for a flow at network element i are computed as the horizontal and vertical distance from the traffic envelope and the service curve, respectively.

Specifically, a user with a TSpec of the form (b, r, p, M), and which has been allocated a "service rate" of R at each of a set of schedulers characterized by cumulative error terms C_{tot} and D_{tot}, is guaranteed an upper bound $\widehat{D}$ on its end-to-end **queueing** delay of the form [23, 54]:

[3] Except for losses due to line errors.

$$\widehat{D} = \begin{cases} \dfrac{(b-M)}{R}\dfrac{(p-R)}{(p-r)} + \dfrac{M}{R} + \left[\dfrac{C_{tot}}{R} + D_{tot}\right] & \text{if } p > R, \\ \dfrac{M}{R} + \left[\dfrac{C_{tot}}{R} + D_{tot}\right] & \text{if } p \leq R. \end{cases} \quad (4)$$

A similar expression is available to determine the necessary buffer space at a given node, so as to avoid any packet loss [23, 54].

In the context of the Guaranteed Service, equation (4) is used by the receiver to determine which service rate R to request in order to obtain the desired end-to-end delay bound, given the TSpec specified by the user and the C_{tot} and D_{tot} values associated with the schedulers on the path. This request for a service rate R is then sent back towards the sender, e.g., using an RSVP RESV message, and processed at each node which determines if it has enough capacity to accept the request, i.e., performs call admission. Call admission varies with the type of scheduler used, and this is one of the areas where different trade-offs between efficiency and complexity are possible. Before proceeding further with this issue, we briefly review a number of interesting properties of the Guaranteed Service.

First, while the use of a rate-based model introduces some limitations in terms of efficiency (see [24] for an example), it also affords significant simplicity. In particular, the use of a single service rate to be guaranteed at each scheduler on the path, avoids altogether the complex problem of trying to determine and assign individual resource requirements at each scheduler. Such a "one size fits all" approach precludes a lightly loaded scheduler from compensating for a more heavily loaded one, e.g., as is feasible with a delay based scheduler such as EDF, where a large deadline at one node can be compensated by a small one at another node. However, exploiting such potential is a challenging problem[4].

Second, the rate-based model does not preclude the use of other types of schedulers. For example, a delay based scheduler such as EDF can be used instead, albeit with some constraints on how the deadlines are to be chosen at each node (see [23] for details).

Third, as mentioned earlier, there is a cost associated with the deterministic guarantees that the Guaranteed Service provide. For example, as is shown in [23, Section 4.3] an average load of only 40% is not uncommon on a link carrying Guaranteed Service flows. The remaining bandwidth will most certainly not be left unused, i.e., it will be assigned to lower priority traffic, but this indicates that the network is likely to charge a premium for Guaranteed Service. The hard guarantees it provides may be important to certain applications, but the cost may not be justified for all.

[4] The optimal assignment of local deadlines for a given end-to-end delay bound is known to be intractable.

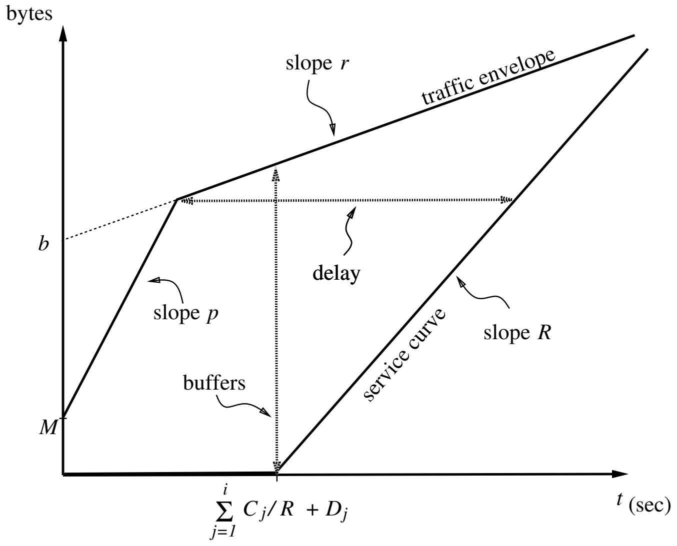

Figure 1: Delay and Buffer calculations for an (b, r, p, M) flow.

4.2.1 Trade-Offs in Implementing Guaranteed Service

We now briefly discuss some of the trade-offs that are available when implementing the Guaranteed Service. As for the Controlled Load service, trade-offs involve both the data path and the control path.

Data path trade-offs are primarily in terms of scheduler efficiency versus implementation complexity, where efficiency is measured through the scheduler's schedulable region [22, 24] (the larger, the better), and complexity is a function of the operations performed when scheduling the transmission of a packet. For example, as was discussed in Section 2, the tighter scheduling (smaller error terms) provided by the WFQ scheduler when compared to, say, the SCFQ scheduler, comes at the cost of more complex computations when scheduling a packet. This affects the implementation cost and speed of the scheduler.

However, it should be noted that the tighter the scheduling, the lower the buffer cost. This is illustrated in Figure 1, which shows the cumulative worst-case delay and buffer requirements at node i. Tightening the scheduling at node i amounts to reducing the error terms C_i and D_i, which in turn shifts to the left the service curve at node i. This shift helps reduce not only the delay, but also the buffering needed at node i. Furthermore, this reduction in buffering also applies to downstream nodes, at least until the next reshaping point (see [23] for a discussion on the benefits of reshaping). As a result, the choice of a particular scheduler needs to take into consideration not only the cost of the scheduler itself, but also the cost of other system wide resources such as link bandwidth and buffer space.

These trade-offs need to be further combined with control path issues, which can also contribute significant differences in terms of complexity and efficiency. In order to better illustrate the issues, we focus the discussion on

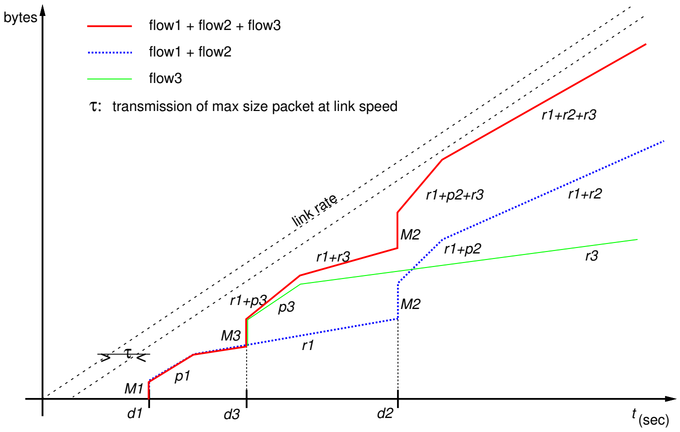

Figure 2: Call Admission Rule for EDF Scheduler.

the differences between rate-based, e.g., WFQ, and delay-based, e.g., EDF, schedulers. Delay-based schedulers have been shown, e.g., [22, 24], to yield larger schedulable regions than rate-based ones and to, therefore, afford more efficient use of link bandwidth when it comes to providing hard delay guarantees. However, this comes at the cost of a more complex call admission process.

In the case of a rate-based scheduler such as WFQ, the call admission decision amounts to determining if the requested service rate R is available. This is easily accomplished by keeping track of the *sum* of the service rate requests that have been granted, and ensuring that this sum remains smaller than the link speed after the new request is added. In other words, call admission amounts to a simple addition and comparison.

The case of a delay-based scheduler such as EDF is more involved. First, in the context of the Guaranteed Service, the requested service rate R needs to be translated into a local deadline which the EDF scheduler can use. As mentioned earlier, this first task is easily performed, e.g., using the approach of [23], so that the bulk of the complexity lies in determining if the new request with its associated deadline and TSpec can be accepted. The exact procedure is as given in equation (3) (see also [23, Section 4]), but is more easily described using an example.

Figure 2 provides a graphical representation of the call admission procedure used with an EDF scheduler. It essentially amounts to ensuring that the "sum" of the shifted traffic envelopes (as specified by equation (1)) of all the flows remains at least one maximum size packet below the line corresponding to the maximum number of bits that can be transmitted at the link rate. The shift applied to each traffic envelope is equal to its local deadline at the EDF scheduler. Figure 2 shows the case where two flows (flow 1 and flow 2) are already in, and a third one (flow 3) is being added. The complexity of the procedure is that it requires both storing the current sum of all shifted traffic envelopes, e.g., by keeping the associated inflexion points, and checking that adding a new shifted envelope does not cause any point of the new curve to cross the link capacity line (this can again be done by checking only at inflexion points). There have been some proposals, e.g., [17], with simplified procedures, but the call admission procedure does remain more complex than the simple addition of rate-based schedulers. As a result, it is again important to consider this cost in the context of an overall system design.

4.3 Differentiated Service

In this section, we briefly discuss some recent efforts started in the IETF to introduce "looser" service specifications than those that have been discussed in the two previous Sections. It is actually interesting to note that this represents somewhat of a backwards evolution. Specifically, some of the earlier works in the area of QoS support had focused on simple rules for call admission, coupled with aggregate priority-based structures for service differentiation (see, for example, [1, 35] for the description of a general framework based on those premises). The main reason was that the data path technology was at the time not considered capable of finer granularity (per flow) control. However, these initial works were quickly followed by many proposals aimed at strengthening service guarantees through the use of more sophisticated scheduling and buffer management algorithms as described in Sections 2 and 3. This evolution was fueled in part by the availability of new components[5] capable of such advanced functions, but also by the desire to better control the service experienced by individual flows (see [8, Section 3.1] for an example of potential discrepancies between global and per flow service guarantees).

Those activities and efforts not withstanding, there has been a renewed interest and focus on "simple" QoS guarantees, and in some instances guarantees that are even more *qualitative* than the Controlled Load service specification. For example, many of the recent IETF discussions on these issues[6] have focused on defining services that map to different levels of *"sensitivities"* to delay and loss, i.e., without being associated to explicit values or guarantees. There are many reasons that have been mentioned to motivate a return to those simple solutions.

One of them is clearly the difficulty of upgrading an infrastructure of the size of today's Internet. The sheer size of the Internet means that it is likely that the path of any flow will, at least for some time, cross portions of the network which are either not QoS capable, or only support crude service differentiation capabilities. This weakens the case for strong per flow end-to-end guarantees. Similarly, support for the signalling capabilities, e.g., RSVP, required to let users and the network negotiate precise service definitions may also not be widely deployed initially.

There are, however, more fundamental arguments that have been put forward. The more prevalent one is that

[5]For example, the CHARM chip [13] can reshape and schedule up to 64,000 individual flows at speeds up to OC-12, and there are today many instances of components with similar capabilities.

[6]Those discussions are currently been carried in the Diff-Serv working group and its mailing list at diff-serv@BayNetworks.com.

there is no real need for strong and explicit QoS guarantees. Instead, proper network engineering and broad traffic classification, e.g., into a small number of "priority" classes, coupled with the adaptive nature of many applications, should be sufficient to offer the necessary functionality. The basic premises and reasoning behind this argument are that only a small fraction of applications have the need for strong guarantees. As a result, adequate provisioning for their peak traffic load, together with protection (through coarse classification) from lower priority traffic, will provide them the desired level of service. Adequate provisioning will also ensure that the rest of the traffic experiences, most of the time, adequate service, possibly with again some coarse differentiation. In case of congestion, flows will adapt their traffic to the available resources, and continue operating albeit at a lower level of service. The benefits are higher *overall* efficiency, i.e., more flows getting through, and greater simplicity, i.e., minimal signalling support and simple data path mechanisms.

The analogy most often used to illustrate this model, is a comparison between IP telephony and the circuit-switched telephone system. In cases of unusually high congestion, a circuit-switched system will block new calls and may even operate at a lower level of throughput because of resources held by calls that are being blocked. In contrast, upon detecting congestion, IP telephones may be able to adjust their coding rate and even be able to still operate with higher packet losses and delays. Assuming that this is achieved without preventing emergency calls from getting through, e.g., by assigning them to a higher priority class, the claim is that this will allow more calls to get through.

There are obviously many assumptions behind the above model, e.g., level of over-provisioning needed, range of adaptation, etc., and it is not the intent of this paper to provide a comprehensive investigation of these issues. This would in itself be the topic of an entire paper. However, there are several other less controversial perspectives for why a coarser QoS may be of value. In particular, the same level of QoS control may not be required in all environments. For example, it may be warranted to provide tight per flow delay control in an access switch or router, but such fine control may be an overkill in a backbone switch running at OC-48 (2.4Gbps) speed or higher. Similarly, maintaining awareness of individual 16kbps flows on an OC-48 link translates into a very substantial and potentially costly amount of state information (there could be up to $150,000$ such flows), as well as a significant signalling load. As a result, it may be worth considering different QoS models in different environments. In particular, the potential for aggregation offered by a "differentiated service" model can be of benefit in the backbone of the Internet, while the finer grain "integrated service" and RSVP model may be more appropriate for access and campus networks. The main challenge is then to ensure inter-operability between these two environments. In the next section, we explore some of these issues and possible approaches to inter-operability.

5 End-to-End QoS: Aggregation Issues and Models

As was mentioned in the previous section, scalability requirements are likely to introduce the need for aggregation. This is especially applicable to the core of the backbone where the potentially large number of flows and the high speeds of the links, can stress cost and complexity. This need for aggregation is by no means a prerogative of IP networks. For example, the hierarchical structure of the VPI and VCI fields in the header of ATM cells [52], was primarily intended to support both data path and control path aggregation. Specifically, forwarding and service differentiation decisions can be made by looking only at the shorter (8 to 12 bits) VPI field instead of the full 28 header bits, and similarly setup and take down of individual VCs within a VP does not require processing of individual signalling messages.

In the case of an Int-Serv/RSVP IP network, the granularity of QoS requests is currently determined through filters that specify destination address and port number, as well as source address and port number in some instances (see [6] for details). This corresponds to a very fine granularity of QoS guarantees, and while this provides end-users with accurate control, it has been identified as a potential problem for scalability. As a result, the goal of aggregation is to allow such individual end-to-end QoS guarantees, but without requiring awareness of individual flows on each and every segment of their path. This goal translates into a number of specific requirements that must be satisfied by any aggregation technique. Those requirements are best understood through an example.

Consider a network topology consisting of three separate Autonomous Systems (AS), with the two edge AS, say, AS1 and AS3, corresponding to local AS and the middle one (AS2) representing a backbone interconnecting the two. For the purpose of our discussion, scalability is of concern only in the backbone AS2, i.e., AS1 and AS3 are capable of maintaining individual flow state information. Aggregation of individual flows in AS2 should then satisfy the following requirements:

- $R1$ AS2 should not have to maintain awareness of individual flows between AS1 and AS3. Instead, AS2 should be able to map individual flows onto few internal service "classes".
- $R2$ Isolation between flows should be maintained in AS2, i.e., even when flows are aggregated into a common service class, the excess traffic of one flow should not affect the performance guarantees of another flow.
- $R3$ AS2 should ensure that it satisfies the QoS requirements of individual flows, e.g., the resources allocated to a service class in AS2 should be consistent with the aggregate resources required by all the individual flows mapped onto it.
- $R4$ Aggregation in AS2 should not prevent support for individual flow reservations in AS1 and AS3.

Requirement $R1$ is the core scalability requirement expressed by AS2. Requirements $R2$ and $R3$ specify properties, that have to be satisfied by the mapping of individual RSVP flows onto the coarser "classes" of AS2. Requirement $R2$ states that the QoS guarantees provided to an individual flow must remain limited to its conformant packets to avoid affecting the service guarantees of other flows. This in turn implies the need for some means to identify non-conformant packet even after flows have been merged, e.g., packet marking as mentioned in Section 4.1. Requirement $R3$ expresses the need for some coupling between the resources (bandwidth and buffer) and level of service (priority) assigned to a class in AS2, and the aggregation of the individual flows mapped onto that class. Requirement $R4$ expresses the important constraint that satisfying scalability in AS2, should not come at the expense of functionality in AS1 and AS3. In other words, the aggregation of control and data path information in AS2 should be reversible, so that reservations in AS1 and AS3 can default back to individual flows after crossing AS2.

The last two requirement, $R3$ and $R4$, are primarily control path issues and represents the main interoperability challenge that QoS aggregation faces. As a result, we focus the discussion in the rest of this section on this specific issue, and assume, as suggested in [9, 10, 15, 16, 36, 41], that data path aggregation is achieved through the use of specific bit patterns in the IP header, e.g., the type-of-service (TOS) octet field [49] is used to specify different service classes and drop precedence[7].

5.1 End-to-end Control Path Interoperability

In the context of aggregation, a key aspect of service guarantees is that the resources allocated to a set of aggregated flows should reflect the "sum" of the reservation requests conveyed in individual reservation messages. There are two generic solutions that can be used to satisfy such a requirement, and they can be broadly categorized as *static* and *dynamic*. Static solutions rely on provisioning and support of different classes of service on backbone links, so that "soft" guarantees can be provided between given ingress and egress points. Dynamic solutions involve the ability to adjust the level of (aggregate) reservation of each service class on individual backbone links as a function of the actual offered traffic. This allows for "harder" service guarantees.

Static solutions are obviously simpler and also easily satisfy requirement $R4$ as dynamic reservation requests, e.g., RSVP PATH and RESV messages, are not processed in the backbone. As a result, such individual setup messages can be directly propagated through the backbone, and require minimal additional processing at the ingress and egress points. For example, update of *ADSPEC* fields in PATH messages can be done using pre-configured information for each ingress. However, static solutions also have limitations in how they handle changes in the backbone. In particular, route changes in the backbone affect the accuracy of provisioning on backbone links, as well as the ability to properly update the *ADSPEC* field to characterize the path actually taken through the backbone.

As a result, even if static solutions are likely to be the first ones deployed, it is desirable to provide solutions that allow a more accurate control of service guarantees while still enjoying the benefits of aggregation. Satisfying this goal has several requirements and implications in the context of Int-Serv/RSVP services:

1. Disable processing of individual RSVP messages in the backbone, while still allowing their identification when they arrive at egress or ingress routers.
2. Identify transparently, i.e., without relying on continuous interactions with routing, the egress routers corresponding to individual flows entering the backbone at an ingress router.
3. Properly update the *ADSPEC* of RSVP PATH messages of individual flows at egress routers.
4. Reserve the appropriate amount of resources on backbone links to satisfy the QoS requirements of individual flows routed over them.

The first two items are protocol specific, and can be handled either by using a separate signalling protocol for setting up aggregate reservation, e.g., see [2, 5], or by reusing the RSVP protocol itself, e.g., [32]. In either cases, individual RSVP reservations are not processed (hidden) inside the backbone, where separate (aggregate) messages are used to reserve resources based on the requirements of the individual flows being aggregated on each link. As far as QoS guarantees are concerned, the main challenges lie in properly specifying aggregate reservations and in ensuring that end-to-end service guarantees of individual flows are preserved. The complexity of those tasks varies with the type of the service, e.g., Controlled Load and Guaranteed Service, and is discussed further in the rest of this section.

In the case of an aggregated Controlled Load reservation, it is only necessary that the queues assigned to Controlled Load traffic on backbone links be allocated the proper service rate. Since rate is an additive quantity, aggregate reservations can be based on the sum of the TSpec's of individual flows (see [61] for a discussion on how TSpec's are summed), although the potential benefits of statistical multiplexing gain when aggregating many flows may allow a lower reservation level. Overall, the situation is similar to what is described in Section 4.1 when using a FCFS scheduler to support Controlled Load flows, i.e., the main requirements are to allocate a sufficient aggregate service rate and to control the buffer space that excess traffic can occupy.

The situation is more complex for Guaranteed Service flows. The main issue is that the provision of individual delay guarantees is tightly coupled with the ability to precisely guarantee a specific clearing (service) rate to a flow. Aggregation affects this ability as characterizing how individual flows share the total rate allocated to them is a

[7] Note that another possible approach is to rely on tunnels (layer 2 or layer 3), but we do not discuss this alternative here, and refer the reader to [32] for more details.

difficult, if not impossible, task. Such a characterization is feasible when aggregation is limited to flows with a common ingress and ingress points (see [32, 50] for details), but not when aggregation is done locally at each backbone link, e.g., all Guaranteed Service flows are assigned to the same queue. In the latter case, the aggregate service rate on a given backbone link is not even known to either the ingress or egress routers associated with the individual flows whose route through the backbone includes that link.

In order to support aggregated Guaranteed Service flows, it is then necessary to change the "node model" used to represent the backbone. A similar problem exist when crossing ATM networks, and this has been addressed by the ISSLL working group [21]. The approach used by ISSLL is to characterize an *entire* ATM network as a single Int-Serv node, that contributes only to the D error term, i.e., represent an ATM network as a delay-only node. The value selected for the D term of the ATM network, is an estimate of the delay bound available from the ATM network between the ingress and egress points. A similar approach can be used here by representing each IP backbone router as a delay only node, which removes the previously mentioned dependency on the unknown aggregate service rates. The main difference with the case considered by ISSLL, is that backbone IP routers can process and update the *ADSPEC* field[8]. The characteristics of the path through the backbone are, therefore, based on the nodes actually traversed, instead of relying on an estimate of the network characteristics between the ingress and egress points.

6 Summary

As described in Sections 2 and 3, a wide range of data path mechanisms exist that enable support for QoS guarantees in packet networks. They offer a broad choice of trade-offs between complexity, performance, and strength of the guarantees being provided. New services have been and are being defined based on the availability of these mechanisms. However, deployment of these services requires more than data path innovation. Enhanced control path mechanisms are needed to enable the specification of service guarantees and the identification of who they apply to. As described in Section 4, progress has also been made in developing such control path mechanisms, but their deployment is facing a number of challenges.

In particular, the control path and data path cost of QoS can vary greatly as a function of the environment where QoS is to be provided. As a result, it is desirable to allow the use of different control path and data path mechanisms in different environments. Specifically, it should be possible to support QoS guarantees using either per flow or aggregated mechanisms based on the characteristics and capabilities of the environment where those guarantees are to be provided. The challenge is then to allow interoperability of those different mechanisms, so that end-to-end guarantees can be provided. Issues and possible approaches to satisfy this goal were discussed in Section 5.

There are many unanswered questions when it comes to determining the appropriate QoS model for each environment, and this paper certainly did not attempt to answer them all. Instead, it tried to clarify trade-offs and requirements in the hope that this would help focus future investigations.

[8] This assumes that RSVP is the signalling protocol used to setup aggregate reservations.

Acknowledgment: The authors would like to acknowledge their many colleagues at the IBM T.J. Watson Research Center for many spirited discussions on the topic of QoS. Their inputs have helped shape much of the material in this paper.

References

[1] A. Ahmadi, P. Chimento, R. Guérin, L. Gün, B. Lin, R. Onvural, and T. Tedijanto. NBBS traffic management overview. *IBM Systems Journal*, 34(4):604–628, December 1995.

[2] W. Almesberger, J.-Y. Le Boudec, and T. Ferrari. Scalable resource reservation for the internet. In *Proceedings of PROMS-MmNet'97*, Santiago, Chili, November 1997.

[3] J. C. R. Bennett and H. Zhang. Hierarchical packet fair queueing algorithms. In *Proceedings of SIGCOMM*, pages 143–156, Stanford University, CA, August 1996.

[4] J. C. R. Bennett and H. Zhang. WF^2Q: Worst-case fair weighted fair queueing. In *Proceedings of INFOCOM*, pages 120–128, San Francisco, CA, March 1996.

[5] S. Berson and S. Vincent. Aggregation of internet integrated services state. Internet Draft, `draft-berson-classy-approach-01.txt`, November 1997. (Work in progress).

[6] R. Braden (Ed.), L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource reSerVation Protocol (RSVP) version 1, functional specification. Request For Comments (Proposed Standard) RFC 2205, Internet Engineering Task Force, September 1997.

[7] I. Cidon, R. Guérin, and A. Khamisy. Protective buffer management policies. *IEEE/ACM Trans. Networking*, 2(3):240–246, June 1994.

[8] I. Cidon, R. Guérin, A. Khamisy, and K. Sivarajan. Cell versus message level performances in ATM networks. *Telecommun. Sys.*, 5:223–239, 1996.

[9] D. Clark. Adding service discrimination to the Internet. *Telecommunications Policy*, 20(3):169–181, April 1996.

[10] D. Clark and J. Wroclawski. An approach to service allocation in the Internet. Internet Draft, `draft-clark-diff-svc-alloc-00.txt`, July 1997. (Work in progress).

[11] R. L. Cruz. Service burstiness and dynamic burstiness measures: A framework. *J. High Speed Networks*, 1(2):105–127, 1992.

[12] R. L. Cruz. Quality of service guarantees in virtual circuit switched networks. *IEEE J. Sel. Areas Commun.*, 13(6):1048–1056, August 1995.

[13] G. Delp, J. Byrn, M. Branstad, K. Plotz, P. Leichty, A. Slane, G. McClannahan, and M. Carnevale. ATM function specification: CHARM introduction - version 3.0. Technical Report ROCH431-CHARM Intro, IBM AS/400 Division, LAN Development, February 1996.

[14] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queueing algorithm. *Journal of Internetworking: Research and Experience*, 1:3–26, January 1990.

[15] E. Ellesson and S. Blake. A proposal for the format and semantics of the TOS byte and traffic class byte in IPv4 and IPv6 headers. Internet Draft, `draft-ellesson-tos-00.txt`, November 1997. (Work in progress).

[16] P. Ferguson. Simple differential services: IP TOS and precedence, delay indication, and drop preference. Internet Draft, `draft-ferguson-delay-drop-00.txt`, November 1997. (Work in progress).

[17] V. Firoiu, J. Kurose, and D. Towsley. Efficient admission control for EDF schedulers. In *Proceedings of INFOCOM*, Kobe, Japan, March 1997.

[18] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Trans. Networking*, 1(4):397–413, August 1993.

[19] S. Floyd and V. Jacobson. Link-sharing and resource management models for packet networks. *IEEE/ACM Trans. Networking*, 3(4):365–386, August 1995.

[20] I. Foster and C. Kesselman, editors. *The Grid: A Blueprint for the New Computing Infrastructure.* Morgan-Kaufman, San Francisco, CA, 1998. (To appear).

[21] M. W. Garrett and M. Borden. Interoperation of controlled-load service and guaranteed service with ATM. Internet Draft, `draft-ietf-issll-atm-mapping-04.txt`, November 1997. (Work in progress).

[22] L. Georgiadis, R. Guérin, and A. Parekh. Optimal multiplexing on a single link: Delay and buffer requirements. *IEEE Trans. Infor. Theory*, 43(5):1518–1535, September 1997.

[23] L. Georgiadis, R. Guérin, V. Peris, and R. Rajan. Efficient support or delay and rate guarantees in an Internet. In *Proceedings of SIGCOMM*, pages 106–116, San Francisco, CA, August 1996.

[24] L. Georgiadis, R. Guérin, V. Peris, and K. N. Sivarajan. Efficient network QoS provisioning based on per node traffic shaping. *IEEE/ACM Trans. Networking*, 4(4):482–501, August 1996.

[25] R. J. Gibbens and P. J. Hunt. Effective bandwidths for the multi-type UAS channel. *Queueing Systems*, 9:17–28, September 1991.

[26] S. J. Golestani. A self-clocked fair queueing scheme for broadband applications. In *Proceedings of INFOCOM*, pages 636–646, Toronto, Canada, June 1994.

[27] S. J. Golestani. Network delay analysis of a class of fair queueing algorithms. *IEEE J. Select. Areas in Commun.*, 13(6):1057–1070, August 1995.

[28] P. Goyal, H. Chen, and H. Vin. Start-time fair queueing: A scheduling algorithm for integrated services packet switching networks. In *Proceedings of SIGCOMM*, pages 157–168, Stanford University, CA, August 1996.

[29] P. Goyal and H. Vin. Generalized guaranteed rate scheduling algorithms: A framework. Technical Report TR-95-30, Department of Computer Sciences, University of Texas at Austin, 1995.

[30] M. Grossglauser and D. Tse. Framework for robust measurement-based admission control. In *Proceedings of SIGCOMM*, pages 237–248, Sophia Antipolis, France, September 1997.

[31] R. Guérin, H. Ahmadi, and M. Naghshineh. Equivalent capacity and its application to bandwidth allocation in high-speed networks. *IEEE J. Select. Areas Commun.*, SAC-9(7):968–981, September 1991.

[32] R. Guérin, S. Blake, and S. Herzog. Aggregating RSVP-based QoS requests. Internet Draft, `draft-guerin-aggreg-rsvp-00.txt`, November 1997. (Work in progress).

[33] R. Guérin, S. Kamat, V. Peris, and R. Rajan. Scalable QoS provision through buffer management. In *Proceedings of SIGCOMM*, Vancouver, British Columbia, Canada, September 1998. (To appear).

[34] L. Gün. An approximation method for capturing complex traffic behavior in high speed networks. *Performance Evaluation*, 19(1):5–23, January 1994.

[35] L. Gün and R. Guérin. Bandwidth management and congestion control framework of the broadband network architecture. *Computer Networks and ISDN Systems*, 26(1):61–78, September 1993.

[36] J. Heinanen. Use of the IPv4 TOS octet to support differential services. Internet Draft, `draft-heinanen-diff-tos-octet-00.txt`, October 1997. (Work in progress).

[37] S. Jamin, P. B. Danzig, S. J. Shenker, and L. Zhang. Measurement-based admission control algorithm for integrated service packet networks. *IEEE/ACM Trans. Networking*, 5(1):56–70, February 1997.

[38] F. Kamoun and L. Kleinrock. Analysis of shared finite storage in a computer network node environment under general traffic conditions. *IEEE Trans. Commun.*, COM-28:992–1003, 1980.

[39] M. Katevenis, S. Sidiropoulos, and C. Courcoubetis. Weighted round-robin cell multiplexing in a general-purpose ATM switch chip. *IEEE J. Select. Areas in Commun.*, SAC-9(8):1265–1279, Oct. 1991.

[40] F. P. Kelly. Effective bandwidth at multi-class queues. *Queueing Systems*, 9:5–16, September 1991.

[41] K. Kilkki. Simple integrated media access (SIMA). Internet Draft, `draft-kalevi-simple-media-access-01.txt`, June 1997. (Work in progress).

[42] H. Kröner, G. Hébuterne, P. Boyer, and A. Gravey. Priority management in ATM switching nodes. *IEEE Trans. Commun.*, COM-9(3):418–427, April 1991.

[43] J.-Y. Le Boudec. Application of network calculus to

guaranteed service networks. *IEEE Trans. Infor. Theory*, 44(3), May 1998.

[44] D. Lin and R. Morris. Dynamics of random early detection. In *Proceedings of SIGCOMM*, pages 127–137, September 1997.

[45] A. K. Parekh and R. G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: The single-node case. *IEEE/ACM Trans. Networking*, 1(3):344–357, June 1993.

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

[47] A. K. J. Parekh. *A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks.* PhD thesis, Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA 02139, February 1992. No. LIDS-TH-2089.

[48] V. Peris. *Architecture for Guaranteed Delay Service in High Speed Networks.* PhD thesis, University of Maryland, College Park, 1997.

[49] J. Postel. Internet protocol. Request For Comments, RFC 791, (standard), Internet Engineering Task Force, September 1981.

[50] S. Rampal and R. Guérin. Flow grouping for reducing reservation requirements for Guaranteed Delay service. Internet Draft, `draft-rampal-flow-delay-service-01.txt`, July 1997. (Work in progress).

[51] A. Romanow and S. Floyd. Dynamics of TCP traffic over ATM networks. *IEEE J. Sel. Areas Commun.*, 13(4):633–641, May 1995.

[52] P. Samudra (Ed.). ATM User-Network Interface (UNI) signalling specification, version 4.0. ATM Forum Signalling Working Group, July 1996.

[53] H. Sariowan. *A Service Curve Approach to Performance Guarantees in Integrated Service Networks.* PhD thesis, University of California, San Diego, 1996.

[54] S. Shenker, C. Partridge, and R. Guérin. Specification of guaranteed quality of service. Request For Comments (Proposed Standard) RFC 2212, Internet Engineering Task Force, September 1997.

[55] S. Shenker and J. Wroclawski. General characterization parameters for integrated service network elements. Request For Comments (Proposed Standard) RFC 2215, Internet Engineering Task Force, September 1997.

[56] M. Shreedhar and G. Varghese. Efficient fair queuing using deficit round-robin. *IEEE/ACM Trans. Netw.*, 4,3:375–385, 1996.

[57] D. Stiliadis and A. Varma. Frame-based Fair Queueing: A new traffic scheduling algorithm for packet-switched networks. In *Proceedings of SIGMETRICS'96*, 1996.

[58] D. Stiliadis and A. Varma. Latency-rate servers: A general model for analysis of traffic scheduling algorithms. In *Proceedings of INFOCOM*, pages 111–119, San Francisco, CA, April 1996.

[59] J. Turner. Maintaining high throughput during overload in ATM switches. In *Proceedings of INFOCOM*, pages 287–295, San Francisco, CA, April 1996.

[60] J. Wroclawski. Specification of the controlled-load network element service. Request For Comments (Proposed Standard) RFC 2211, Internet Engineering Task Force, September 1997.

[61] J. Wroclawski. The use of RSVP with IETF integrated services. Request For Comments (Proposed Standard) RFC 2210, Internet Engineering Task Force, September 1997.

[62] H. Zhang and D. Ferrari. Rate-controlled service disciplines. *J. High Speed Networks*, 3(4):389–412, 1995.