# Chapter 1

# Network Quality of Service

## 1.1 Introduction

This chapter is devoted to the issue of Quality-of-Service (QoS) support at the network level as well as the use of such capabilities by distributed applications. In particular, this chapter reviews existing network mechanisms available to support QoS, and attempts to identify what they imply in terms of application performance and behavior. It also points to differences in *cost* between network services, as applications should also consider this aspect when selecting a service.

Before proceeding further and without attempting a rigorous or comprehensive definition, we first try to clarify what is meant by network QoS. The network is responsible for the delivery of data between entities involved in a distributed application, and this delivery has several dimensions that reflect the operational requirements of the application. Examples of those dimensions include the amount of data that needs to be delivered (rate guarantees), the timeliness of their delivery (delay and jitter guarantees), the quality of their delivery (loss guarantees). Network QoS implies some form of commitments along one or more of these dimensions. This differs from the current "best-effort" Internet model, which does not differentiate along such dimensions and where the network makes no commitment regarding the delivery of data.

In this chapter we focus on network services supported in IP networks. While there are a number of networking technologies that offer QoS services, our choice of IP is primarily motivated by the fact that it is likely to be the network technology that is visible to applications. In other words, we believe that, in most instances, the API that applications will use to request QoS services from the network will be IP based. In order for applications to better understand the capabilities and limitations of various network QoS services, it is helpful to first identify the basic building blocks that networks use in offering and supporting those services. Next, we briefly introduce the two major ones.

**Control path:** The control path consists of the mechanisms that let an application describe the kind of service it wants to request, and allow it to propagate that information through the network. The control path is of particular importance to applications as it determines the semantics of the *interface* that the network makes available to applications, in order for them to request services. One example of such an interface to be used with the RSVP[1] protocol can be found in [6].

**Data path:** The data path encompasses those mechanisms that actually ensure that the delivery of application packets meets the agreed-upon guarantees. These mechanisms are not specific to a given network technology and typically need not be made visible to applications requesting service from the network.

Another very important aspect that applications need to understand is **cost**. There is a cost associated with the provision of QoS guarantees in the network because the network has to allocate (dedicate) some amount of resources, in order to ensure it can satisfy the applications requirements. The amount of resources needed will vary as a function of how stringent the requested QoS guarantees are and how efficient the network is at utilizing its resources. However, applications need to be aware of the fact that certain guarantees may be expensive to provide. For example, as discussed in Section 1.4.2, services that allow applications to request hard delay guarantees, while feasible even over IP networks, are likely to be among the more expensive ones. As a result, an application may want to evaluate its requirements for hard delay bounds against this cost. As discussed in Section 1.6, this is especially true for applications that can adapt to a certain range of fluctuations in network performance.

### Chapter Outline

In the rest of this chapter, we expand on the various QoS services that networks offer. Section 1.2 provides a general perspective and classification of the criteria that an application may want to use in order to assess the suitability of different network services. The next two sections, Section 1.3 and Section 1.4, focus on the two main components that influence the kind of network services available from the network, i.e., the signaling and service definition models. In Section 1.3, we describe the characteristic and behavior of the RSVP protocol

[1] The acronym RSVP stands for Resource reSerVation Protocol.

that lets applications request QoS guarantees from IP networks. In Section 1.4, we review the current service models that have been defined for IP networks, namely Guaranteed Service and Controlled Load. Section 1.5 highlights criteria of significance to applications when selecting a specific service, while Section 1.6 describes alternative techniques that applications can use to adapt their requirements to the available network resources. Finally, Section 1.7 identifies a number of extensions and ongoing activities.

## 1.2 Finding Out Which Service Is Right for You

In this section, we outline a possible road map for how an application may choose between different network services. An application contemplating the selection of a particular network service should evaluate it along four main dimensions:

1. What kind of performance guarantees is it able to provide, e.g., throughput, delay, loss probability, delay variations, etc., and how do those guarantees rank against the application's requirements?
2. What does it require in order to provide such guarantees, i.e., what constraints does it impose on user behavior (traffic descriptors, conformance rules, etc.)?
3. What kind of flexibility does it offer to deal with variations in user requirements, i.e., how does it handle excess traffic?
4. What does it cost?

1. *What kind of service guarantees?*

Typically, network service guarantees specify the reliability of data transfer across the network, the kind of transit delay that the data will experience as well as possibly the variations in this transit delay. Loss guarantees can range from the promise not to lose any data except due to transmission errors, to specific loss probabilities, to only the avoidance of excessive losses through "adequate" provisioning. Similarly, delay guarantees can cover a wide range, from hard (deterministic) bounds on the maximum delay, to loose guarantees that "typical" delays won't exceed some generic value. Guarantees on delay variations follow a similar pattern. For each of the above guarantees, the application needs to assess their significance on the quality of its operation: The more stringent its requirements, the higher the associated network cost will be.

Another important aspect that applications need to consider is the relation between the data units of relevance to the application, e.g., a complete video frame, and the network data units to which the service guarantee will apply. The two are often quite different. For example, the loss of a single network data unit may render unusable a whole application data unit consisting of multiple network data units. It is, therefore important for applications to factor in those potential differences when requesting a certain level of service from the network (see [10] for discussions on these issues).

2. *What does the network require in order to provide a given service guarantee?*

Typically, networks only provide service guarantees for a specific amount of traffic that the application must specify when contracting the service. In other words, a *service* contract is typically associated with a *traffic* contract. Networks support a range of traffic contracts, but the challenge is for an application to select a contract that accurately captures its traffic characteristics. Alternatively, the application may choose to renegotiate its service and traffic contracts as its requirements change. For example, an application could ask for a higher level of service, e.g., lower delay, or request the permission to inject more traffic in the network to accommodate an increase in activity.

3. *How does the network handle excess traffic?*

While a flexible traffic contract or support for service renegotiations can help provide a service that matches an application requirements, there are still instances where an application will exceed its contract. For example, some applications often specify minimal traffic and service contracts on the basis of cost. The contracts are only meant to ensure the application's ability to operate even in the presence of heavy network congestion, and are not really representative of their typical requirements. As a result, such applications will usually generate a volume of traffic that is much higher than the value specified in their traffic contract. Network services that allow transmission of this excess traffic, albeit at a lower priority and to the extent that network resources are available, are clearly preferred for such applications.

Many of the above aspects are tightly coupled and in general it is possible for an application to come up with multiple answers, each representing a different trade-off. In order to better understand the process that an application may want to follow, it is useful to use an example as a guiding thread when reviewing the many different aspects that an application must evaluate when selecting a network service.

### 1.2.1 A Road map for Service Selection

In the rest of this section, we use the Tele-Immersive collaborative design system application[2] of Chapter **??** to illustrate some of the possible choices that an application may face. The TI application actually consists of multiple types of flows, each with different requirements and as a result possible trade-off in terms of network QoS. Its flows and the requirements that are of greatest significance to network QoS are summarized in Table 1.1.

The parameters shown in Table 1.1 are as follows (see Chapter **??** for details). The *latency* requirement specifies the end-to-end delay, including propagation and network queueing delays, that each component flow can tolerate, while the *stream* characteristic identifies flows with explicit synchronization constraints and which are, therefore, also sensitive to delay variations, i.e., *jitter*. The *bandwidth* requirement expresses the peak amount of data each flow is expected to generate. Flows that are identified as *multicast* expect the network to replicate and deliver their data to multiple ($N$) entities. Finally, the *Dynamic QoS* column specifies the expected range of fluctuations of each flow's traffic characteristics and service requirements.

| Flow Type | Latency | Bandwidth | Multicast | Stream | Dynamic QoS |
|---|---|---|---|---|---|
| Control | medium | 64 kb/s | No | No | Low |
| Text | high | 64 kb/s | No | No | Low |
| Audio | medium | $2 * 128$ kb/s | Yes | Yes | Medium |
| Video | medium | $N * 5$ Mb/s | Yes | Yes | Medium |
| Tracking | low | $N * 128$ kb/s | Yes | Yes | Medium |
| Database | high | $> 1$ GB/s | Maybe | Maybe | High |
| Simulation | medium | $> 1$ GB/s | Maybe | Maybe | High |
| Haptic | low | $> 1$ Mb/s | Maybe | Maybe | High |
| Rendering | medium | $> 1$ GB/s | Maybe | Maybe | Medium |

Table 1.1: Requirements of Tele-Immersive Application.

In the rest of this section, we review possible network service choices for the different flows of the TI application.

**A Conservative Service Selection**

At one extreme, the application may insist on having the network deliver all of its data as it is being generated, with minimal disruption, and with the smallest possible latency. This is essentially the service that would be provided

[2] We'll refer to it as the TI application in the rest of this chapter.

by a fixed rate *circuit*. Such a service is also available from packet (IP) networks in the form of the *Guaranteed Service* (see Section 1.4.2), and may be the right choice for flows such as *audio, video, tracking,* and *haptic* which have relatively stringent delay and/or synchronization requirements.

Requesting this service will result in the allocation of a guaranteed bandwidth pipe between the sender and all receivers. The amount of bandwidth of the pipe is set to guarantee that the flow's delay requirements are met. The above service selection could also be suitable for the *simulation* and *rendering* flows, which have similar stringent delay requirements. However, their much higher peak data rates (1 GB/s) makes such a choice all but impractical as it would be prohibitively expensive.

The guaranteed bandwidth service represents one extreme in the spectrum of solutions, where the worst case resources are allocated. There are, however, other alternatives whose desirability depends on both the efficiency of the network resource allocation mechanisms, and the traffic characteristics of the application.

**Relaxing Service Guarantees for a Lower Cost**

There are many approaches an application can follow to lower the cost of its network service. One is to basically tell the network that it won't need all the bandwidth all the time. The issue there is how to express this in a quantitative manner. One approach is to "predict" one's expected range of traffic fluctuations. Another approach is to only ask for bandwidth when it is needed. This requires continuous "renegotiations" between the network and the application as the rate of a flow changes. This avoids the difficulty of predicting traffic fluctuations, but creates the risk of the bandwidth not being available when requested. Yet another approach is to ask only for a minimal service guarantee (none?) and adapt to the availability of network resources. There are pros and cons with each approach and we briefly review them.

The specification of an *"envelope"* will secure the availability of the necessary network resources for the traffic contained within that envelope. However, identifying the "right" envelope can be a difficult task for an application, e.g., it may be feasible for the *audio* flow but much harder for the *simulation* flow whose behavior can widely vary. The impact of this uncertainty can be mitigated by the fact that there is not a unique correct answer, and a range of traffic envelopes is likely to be adequate. For example, the flow's traffic can be *reshaped*, i.e., by buffering it and delaying its transmission, to make it fit different traffic profiles. However, this assumes that the flow can tolerate the additional (reshaping) delay.

Renegotiating with the network entails overhead and latency in receiving the desired service, and even possibly failure to do so, if network resources are unavailable. The impact of such failures can be mitigated by requesting a "floor" service guarantee. In general, the use of renegotiations, with or without floor guarantees, is mainly suitable to *adaptive* applications, i.e., applications that support different levels of operation and can adjust to accommodate changes in the availability of network resources (see Section 1.6 for additional discussions on the issue of adaptation). For example, the *database* flow could request a floor bandwidth of, say, 10 Mb/s (substantially lower than its peak rate of 1 GB/s), but attempt higher speed transmissions whenever possible. Similarly, if the *video* flow supports multi-level encoding, it could select a floor bandwidth matching the transmission requirements of the coarsest coding level, and send additional coding levels on the basis of available network bandwidth.

In general, and even often independent of which level of service an application requests from the network, it is often the case that it will at some point be inadequate. In other words, the application will be generating traffic in *excess* of its contract with the network. How the network reacts to such situations is of importance to applications.

**What happens when the selected service is insufficient?**

In most cases, networks allow applications to transmit traffic in excess of what they have requested. However, the network will usually carry this excess traffic at a lower *loss* priority, i.e., excess traffic will be dropped first in case of congestion. The main difference is in how the network identifies excess traffic. This is done either implicitly or explicitly.

Implicit identification of excess traffic is typically used in networks that are able to guarantee resources to *individual* applications. This means that network elements monitor the resource usage of each individual application. If a network element is uncongested and has idle resources, the application will be able to access them and its excess traffic will get through. However, when congestion is present, the network element will limit the amount of resources that the application can only use what it is entitled to from its service contract, and its excess traffic will be dropped.

Controlling the resource usage of individual flows can be costly. As a result, networks often rely on another mechanism[3] based on explicit marking (tagging) of excess traffic. Packets that exceed the service contract requested by the application are identified and "tagged" as they enter the network, and dropped first in case of congestion. Because tagging is a global indicator, it eliminates the need to control individual flows. The penalty is a coarser control, so that applications can experience substantial differences in the amount of excess traffic they can successfully transmit. However, note that application can decide to

[3]This mechanism is often also used in conjunction with explicit control mechanisms when they are present.

"pre-mark" their packets, and exercise some control over which packets should be preferentially dropped.

Finally, applications need to also be aware that in some instances the network will send excess traffic on a lower priority path, that is *distinct* from the one used for regular traffic. This means that packets can be delivered *out-of-order.* For streaming flows such as the audio, video, and tracking flows of the TI application, the resulting packet misordering can have a substantial impact on the delay and delay variations they experience.

In the next two sections, we review existing network services, and identify the guarantees and options they offer.

## 1.3 RSVP: A Protocol for QoS Signaling in IP Networks

The RSVP protocol is a resource reservation setup protocol designed for an integrated services Internet. The protocol is documented in several IETF documents, primarily [5, 7, 32], that describe various aspects related to its operation and use. The protocol is intended to be used by both hosts (end-systems) and routers (network elements), i.e., the same RSVP messages are used by hosts to communicate with routers and by routers to communicate with each others. From the point of view of an application, the RSVP protocol has a number of characteristics that are of significance.

- Requests for resources are initiated by the receiver based on information it receives from the sender. The sender communicates with the receiver(s) through *PATH* messages. The information carried in *PATH* messages includes characteristics of the sender's traffic, **and** of the path that the flow will take through the network. The main *potential* advantage of knowing the characteristics of the path is that this can assist the application in reserving the "right" amount of resources. For example, knowing the maximum amount of bandwidth available on the path can avoid requesting more bandwidth than is feasible. Additionally, knowledge of the path characteristics is key to supporting hard end-to-end delay guarantees (see Section 1.4.2). Reservations are communicated from the receiver(s) using *RESV* messages, that propagate hop-by-hop back towards the sender.

- RSVP requests resources for simplex flows, i.e., it requests resources in only one direction, from sender to receiver. This means that in the case of an application requiring duplex communications, separate requests need to be sent by the application at each end. For example, in the case of the TI application of Chapter **??**, each of the sites involved in a TI session needs to individually reserve resources to receive data from all other sites.

- The protocol allows renegotiations of QoS parameters at any time. This is simply achieved by sending a new reservation message with updated QoS parameters. The network will attempt to satisfy the new request, but if it fails will leave the old one in place. This support for dynamic QoS negotiations can be a powerful tool for applications, especially those that may not be able to accurately predict their traffic patterns or exhibit a wide range of variations.

- The RSVP protocol allows *sharing* of reservations across flows. This means that a receiver can specify that a given reservation request can be shared across multiple senders. In addition, RSVP allows two styles of reservations, *implicit* and *explicit* reservations (see [5] for details).

  The availability of shared reservations can be of benefit to applications involving multiple parties, but some care must be exercised because of the lack of specification on how sharing of the reservation is to be carried out. There are currently no mechanism available that will let an application specify to the network how it wants sharing of resources to be done. As a result, resource sharing can only be controlled at the application level. For example, in the context of an audio conference with ten participants, bandwidth could be reserved assuming that at most two speakers will be simultaneously active. The implied coordination mechanism is that speakers will be backing off as soon as they realize that several of them are talking simultaneously. For other applications, special coordination mechanisms may be needed to take advantage of shared reservations.

- The RSVP protocol supports heterogeneous reservations in the case of multicast connections. This means that receivers listening to the same sender can request different levels of service. As reservation requests from the receivers travel back to the sender, they are *merged* so that only the "larger[4]" one is propagated upstream. Heterogeneous reservations can be useful to support different end-to-end delay requirements of geographically distributed receivers.

The RSVP protocol is of importance to applications as it is the mechanism through which they will talk to the network to request specific service guarantees. This determines not only the semantics of the network services that will

[4]The RSVP protocol defines the operation of *Least Upper Bound* (LUB) to specify how merging of reservations is to be done.

be supported, but also the API that applications will need to talk to in order to request those services (see [6] for an example). In particular, application developers need to be aware that applications will have to be modified to interact and exchange parameters with the RSVP signaling daemon. In addition, in the case of an OS with support for real-time scheduling and/or prioritization as discussed in Chapter **??**, an interface is also needed between the RSVP daemon and the OS QoS manager to ensure the allocation of appropriate OS resources. In that context, RSVP is then also the entity that applications will use to request QoS guarantees from the OS as well as the network (see [1] for an example of such a system).

In the next section, we describe the two services currently available to obtain QoS guarantees in IP networks.

## 1.4 Service Specifications for QoS Guarantees in IP Networks

There has been a number of proposals defining different types of service guarantees to be provided in IP networks, but currently only two are being standardized. These two services, Controlled Load service [31] and Guaranteed Service [29], can be considered to be at opposite ends of the spectrum of QoS guarantees. The Controlled Load service only provides loose delay and throughput guarantees, while the Guaranteed Service ensures lossless operation with hard delay bounds. Despite those differences, both services share a number of common elements, such as the formats and parameter sets used to *characterize* flows and service capabilities in the networks.

A first important set of such parameters is the specification used to characterize the traffic from a sender on behalf of which a reservation is being made (by a receiver). For applications, this is a key set of parameters as it determines which of the application packets are eligible to receive the requested QoS guarantees. This traffic specification[5], or TSpec, consists of a token bucket, 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 used by the network to identify the packet to which the reservation applies. This conformance test states that the reservation applies to all packets of the flow as long as the amount of traffic $A(t)$ it generates in any time interval of duration $t$ verifies:

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

$$A(t) \leq \min(M + pt, b + rt) \tag{1.1}$$

Equation (1.1) limits the amount and speed at which the application can inject data in the network. In addition, the minimum policed unit $m$ is used to impose that any packet of size less than $m$ be counted as being of size $m$. This is to account for possible per packet processing or transmission overhead, e.g., the number of cycles required to schedule a packet transmission.

As was mentioned in Section 1.2.1, the challenge for an application is to determine which values to pick to characterize its traffic to the network. The peak rate $p$ is often set to the raw speed of the network interface of the end-system where the application resides. However, this can be quite expensive, e.g., in the case of SONET OC-12 or Gigabit Ethernet interfaces, as the network usually charges a premium for high peak transmission rates. This is because the network needs to provide buffering to absorb very high speed bursts. Hence, an application may want to specify a lower peak rate and control the transmission of its packets to ensure that it complies with this lower rate. However, note that this requires support for such pacing in the network interface (see Chapter **??** for additional discussions on network interface characteristics).

The selection of the token rate $r$ and token bucket depth $b$ is a more complex task. A large token bucket depth gives the application the ability to burst (at its peak rate) data into the network without any delay. This is key to minimizing latency. However, as was mentioned earlier, large bursts are difficult for the network to handle and require additional resources (buffers), which therefore increase the cost of the service. A possible alternative is to trade a large token bucket depth for a smaller one, but increase the value of the token bucket rate $r$. Transmissions can then proceed at a reasonably high rate even when the token bucket is empty. The higher the token rate $r$, the lower the additional transmission latency when the token bucket is empty, but on the other hand, the higher the token rate $r$, the higher the service cost. This is because the value of $r$ directly corresponds to the minimum amount of bandwidth the network needs to allocate to the flow.

The right choice for $b$ and $r$ depends on both the service pricing model used by the network and the application characteristics. For instance, the *haptic* flow of the TI application has stringent delay requirements, so that it may want to minimize the likelihood of running out of tokens. This would lead to the choice of a token bucket depth $b$ sufficient to accommodate the maximum transmission burst the application can generate. Similarly, its token rate $r$ would be chosen high enough to ensure that the token bucket is always replenished between consecutive transmission, e.g. $r \geq 1$ Mb/s. On the other hand, less dynamic flows such as the the *control* and *text* flows should be able to select a small value for $b$, e.g., one or two packets, and a token bucket rate $r$ approximately equal to their long term bandwidth requirements of 64 kb/s.

Independent of how the TSpec was selected, packets whose transmission violates the condition of equation (1.1) are deemed *non-conformant* with the TSpec, and are not eligible for the service guarantee implied by the reservation. Those packets will then be treated as *best-effort* by the network.

There are a number of other characterization parameters shared by the Controlled Load service and the Guaranteed Service. Some of the more interesting for applications are the *minimum path latency*, the *available path bandwidth*, and the *path MTU*, and we refer the reader to [30] for their definitions and discussions on their uses.

### 1.4.1 Controlled Load Service

The service definition of the Controlled Load service is qualitative in nature. Its stated aim is to approximate the kind of service that the application would experience from an *unloaded* network. The main aspect of the Controlled Service is that it assumes the use of *call admission* to ensure that flows with reservations see this level of performance, irrespective of the actual volume of traffic in the network.

When using the RSVP protocol[6], this request for reservation is communicated back from the sender using a *FLOWSPEC*, that essentially specifies a TSpec of the same form as the one used by the sender. It should be noted that the TSpec specified by the receiver need not be the same as the one used by the sender. This ability can be used by applications in a number of ways. For example, a receiver with limited ability to buffer bursts, may specify a smaller value of the token bucket depth $b$ than the one used by the sender. This will signal to the sender that it should "reshape" its traffic accordingly. For example, the recipient of a database synchronization in the TI application of chapter **??** may not want to be dumped with Gigabytes of data, and may specify a TSpec that will prevent this from happening.

As mentioned earlier, the service guarantees only apply to conformant packets, i.e., packets that pass the conformance test of equation (1.1). The treatment of packets in excess of the TSpec is left unspecified in the Controlled Load specifications. Quoting from [31], the specifications state:

> "The controlled-load service does not define the QoS behavior delivered to flows with non-conformant arriving traffic. Specifically, it is

[6]The Int-Serv specifications have been defined to be independent of the signaling protocol used, even if the RSVP protocol is likely to be the prevalent one.

> permissible either to degrade the service delivered to all of the flow's packets equally, or to sort the flow's packets into a conformant set and a non-conformant set and deliver different levels of service to the two sets."

For an application, the two behaviors have a very different impact.

The first approach corresponds to an implementation where the network guarantees a transmission rate to each Controlled Load flow that is no less than the flow's token rate $r$, e.g., using mechanisms such as Weighted Fair Queueing (WFQ) [12, 22]. In this case, if the flow sends at a rate greater than $r$ for an extended period of time during which the network is congested, packets from the flow will start accumulating in the network buffers as they are arriving faster than they can be transmitted. As a result of these larger queues in the network, the end-to-end delay seen by all packets will increase. This may be an adequate behavior in the case of an adaptive application that will detect the increase in delay and use it to lower its rate, e.g., all the way down its original TSpec. However, this may not be a desired behavior for an application, such as Internet telephony, that is sensitive to increases in delay and can tolerate some losses. Such applications would prefer to see non-conformant packets dropped during congestion rather having all packets experience additional delay.

The second network behavior where conformant and non-conformant packets are treated differently may then be more appropriate for delay sensitive applications. However, it should be noted that the identification of non-conformant Controlled Load packets *inside* the network is a difficult, if not impossible, task[7]. The end result is that even when the network implementation of Controlled Load handles non-conformant packets by downgrading them to a lower priority service, this will only happen after the application experiences some initial increase in end-to-end delay.

There is currently no mechanism available to let an application signal to the network which of the two above behaviors it would like to see to handle its non-conformant traffic. Dealing with this uncertainty may require additional functionality in some applications, e.g., a delay sensitive application may want to preemptively drop packets to limit increases in end-to-end delay when it detects congestion. Alternatively, an application can also take advantage of the renegotiation ability of the RSVP protocol, and attempt to increase its reservation in case its traffic warrants it and the network is becoming congested.

Flows from the TI application that could use the Controlled Load Service are essentially those with relatively loose delay and synchronization requirements.

[7]Note that some recent proposals [11] that would allow "marking" of non-conformant packets could help alleviate this problem

The Controlled Load service should be suitable for the *text* and *database* flows, and even possibly the *audio* and *video* flows. The ability of the latter two to use Controlled Load will depend on how tolerant they are to delay variations.

### 1.4.2 Guaranteed Service

The Guaranteed Service aims at providing hard (deterministic) service guarantees to applications. Those hard guarantees apply again only to conformant packets. For conformant packets, the network commits to an upper bound on the end-to-end delay which they will experience and ensures that no packet will be lost except for transmission errors. The goal of the service is to emulate, over a packet-switched network, the behavior of a dedicated rate circuit.

An application requesting the Guaranteed Service needs to specify the characteristics (TSpec) of the traffic to which it wants the guarantees to apply, as well as the value of the maximum end-to-end delay it wants. Rather than getting into details on how such guarantees are to be supported inside the network, we refer the reader to [29], and instead we concentrate on the features of the Guaranteed Service of relevance to applications.

First, applications should be aware that the Guaranteed Service is likely to be an "expensive" service. This is mainly because of the deterministic nature of the guarantees it provides. We illustrate this next for three applications with different traffic characteristics and end-to-end delay requirements. The three applications are 64 kb/s packetized voice, packet video conference, and playback of stored video. The traffic parameters of those three applications are given in Table 1.2, together with the associated end-to-end delays and corresponding rate. The service rate was computed assuming a five hops path, where the end-to-end propagation delay was taken to be 20 ms. As can be seen from the table, the service rate $R$ can be substantially higher than the token rate $r$. This is particularly significant for the voice application, where the token rate $r$ equals the peak rate $p$, but the required service rate $R$ is about three times as much.

| Traffic Type | $M$ kB | $b$ kB | $r$ Mb/s | $p$ Mb/s | e2e delay | $R$ Mb/s |
|---|---|---|---|---|---|---|
| 64 kb/s voice | 0.1 | 0.1 | 0.064 | 0.064 | 50 ms | 0.162 |
| Video conf. | 1.5 | 10 | 0.5 | 10 | 75 ms | 2.32 |
| Stored video | 1.5 | 100 | 3 | 10 | 100 ms | 6.23 |

Table 1.2: Sample applications and service requirements.

Another sample point illustrating the high cost of the Guaranteed Service is found in [15], which gives some sample achievable link loads. For example, on an OC-3 link ($\approx$ 150 Mb/s), a typical mix of flow that saturates[8] the link achieves a typical utilization of about 40%. 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 worth it for certain applications, but the cost may not be justified for all.

In the context of the TI application, the two flows for which the Guaranteed Service may be the right service are the *tracking* and *haptic* flows because of their stringent delay requirements. The *control* flow may be another candidate because of its combination of reliability and relatively low delay requirements. The delay and losslessness guarantees of the Guaranteed Service may then be well matched to those requirements.

Another aspect that applications need to be aware of as it can significantly impact the cost of Guaranteed Service, is packet size. In general, the larger the (maximum) packet size used by the application, the higher the service rate that will have to be reserved in order to guarantee a given end-to-end delay bound. This impact is primarily due to the store-and-forward nature of packet networks, that results in packet transmission times being paid at each hop. As a result, it is strongly advisable for applications wishing to use the Guaranteed Service, to specify the smallest possible packet size that is compatible with other system requirements. The impact of packet sizes is shown in Table 1.3 for the three applications described earlier.

| Max. Pkt | Reservation Rate $R$ (Mb/s) | | |
|---|---|---|---|
| Size $L$ (kB) | 64 kb/s Voice | Video Conf. | Stored Video |
| 0.1 | 0.16 | 1.40 | 5.91 |
| 0.5 | 0.81 | 1.66 | 6.00 |
| 1.0 | 1.62 | 1.99 | 6.11 |

Table 1.3: Impact of packet size on reservation rate.

A last aspect of Guaranteed Service that applications need to be aware of, is that its deterministic guarantees do not lend themselves well to dealing with shared reservations and handling of non-conformant traffic.

[8]No more flows can be added without resulting in the violation of the end-to-end delay guarantees of at least one flow.

### 1.4.3 Summary of IP QoS Service Offerings

In this section, we have described the two services that have been defined to offer QoS guarantees over IP networks. When used with the RSVP protocol, those services will let applications interact with the network to request and negotiate a variety of service guarantees. Here, we summarize the main features and constraints associated with those services, and also highlight a number of service guarantees that the network is today not capable of providing.

The network can provide rate, delay, and loss guarantees to both unicast and multicast flows. In particular, the Guaranteed Service can provide hard delay bounds and lossless transmission which should be capable of satisfying the needs of the most stringent real-time applications, albeit at a potentially high cost for the service. The Controlled Load service is a suitable alternative for applications which do not have stringent delay constraints and only require that the network guarantee them a certain transfer rate.

For both services, the guarantees apply only to packets that fall within a specific traffic contract. The specification of this traffic contract is probably the biggest challenge that applications face in requesting QoS guarantees. This is because it imposes that they characterize their expected traffic patterns, which may be a difficult task for many applications. The main impact of the traffic contract is in terms of what happens to data that fall outside. Not only is this data not covered by the service guarantees, but no mechanism is available to the application to specify to the network how it wants it handled, e.g., which packets to drop or delay in case of congestion, whether excess packets can be sent on a separate path, etc. Applications need to be aware of this possible range of behaviors, and they may want to adjust their selection of a traffic contract accordingly.

The network will support dynamic renegotiation of service guarantees, so that applications can adjust both their traffic contract and QoS requirements. This can help offset some of the complexity in selecting an appropriate traffic contract. The network also supports sharing of reservations across traffic from multiple senders. However, this sharing is "blind" in that the network will not distinguish between the individual flows sharing the reservation. Hence, the burden of controlling this sharing lies with the application.

In general, it is important to note that all the service guarantees that the network provides do not extend *across* flows. In other words, the network does not allow the specification of service guarantees that would allow an application to request, for example, a bound on the maximum delay difference between packets from two different flows. This could be useful to, say, synchronize and audio and a video stream, but can only be supported by specifying separate delay bounds for each flow and having the application perform the necessary synchronization in the end-system.

Another important limitation of the current network service models is that advanced reservations are not supported. The network simply grants or denies a service request based on the availability of resources *at the time* the request is made. This can be of significance to applications such as the TI application which require scheduling the availability of many different resources, e.g., supercomputers, workstations, CAVEs, etc. There are currently no mechanisms available to ensure that network resources will also be available at the same time. A solution to that problem should become available using the policy control mechanisms mentioned in Section **??**, but their ubiquitous deployment is still some time in the future.

Finally, it should be noted that network QoS guarantees are only one component in providing applications with the service guarantees they require. There are many other factors that will affect end-to-end performance, but the higher layer, e.g., transport, protocol used by the application and the resource management capabilities of the operating system are two that can have a major impact.

For example, the behavior of a transport protocol such as TCP in the presence of losses can substantially affect the actual useful throughput that an application can achieve. Conversely, the use of a real-time transport protocol such as RTP [26], can help an application recover from delay variations experienced when crossing the network. In general, higher layer protocols play an important in providing applications with the desired service guarantees, and the reader is referred to Chapter **??** for additional discussions on this issue.

Similarly, ensuring that adequate resources are allocated to the application in the operating system is key to delivering end-to-end service guarantees. For example, it may be of little use for an application to select a service such as Guaranteed Service, if similar guarantees cannot be provided in the operating system (see Chapter **??** for additional information on performance issue in the operating system).

## 1.5 Examples of Service Selection Criteria

While typically bulk data transfers such as TI database updates are treated as having no specific QOS requirements, certain applications such as remote back-up or downloading media content for off-line playback may well require a minimum guaranteed throughput.

Transaction-oriented applications such as RPC or remote login require a response time commensurate with human patience, usually on the order of a second or less. Unless RPC can be pipelined and parallelized, propagation delay makes it difficult to use in a wide-area network, so that resource control as described in this chapter tends to be of little help.

Multimedia applications have widely varying delay and throughput requirements, even if the content is similar. We can distinguish four types of continuous media applications, namely stored, non-interactive, stored, with trick modes such as fast forward, interactive (conferencing), without echo; interactive, with echo. The TI audio, haptic and video streams fall into the interactive, no echo, category.

Stored, non-interactive multimedia services are limited in delay only by the ability of the receiver to buffer content. If the viewer is to have the ability to fast-forward or skip through the presentation, the round-trip delay can be no more than about half a second to ensure that control action and visible result can be correlated by the viewer. Surprisingly, the acceptable one-way delay for interactive multimedia such as video conferencing is of about the same magnitude as that for stored video, 200 to 300 ms [8, 17]. The one-way delay tolerance decreases to 45 ms if there is an acoustical or electrical echo for audio. For haptic feedback, delay constraints may be much tighter, but it may be possible to limit the need for feedback to the local environment rather than propagating it across the network.

It should be noted that the variable network delay that can be addressed by the resource control mechanisms described in this chapter may only be a small fraction of the total delay budget: Transcontinental propagation delays add about 25 ms (5 $\mu$s/km). Audio codecs have to wait for a whole block of audio, typically about 20 to 40 ms, and may have an algorithmic look-ahead of around 5 to 10 ms. Video codecs often have coding delays of several frames at 30 ms each. The operating system, unless specifically designed for low latency, may add substantial buffering and DMA delays, with a certain popular operating system adding up to one second of delay. Given the delays inherent in a packet network, packet audio is only feasible in the wide area if echos are suppressed.

Since the queueing delay for weighted fair queueing is proportional to the allocated rate, a flow can decrease its delay by merging into a larger flow, at the cost of loosing protection against other members of the aggregate flow.

While attracting a fair amount of attention, delay jitter is only a secondary quality-of-service parameter. A receiver can always convert delay jitter into additional fixed delay by using a playout delay buffer, possibly adaptive, as discussed in Section 1.6.2 below.

Some applications require that several streams are synchronized in time, e.g., for maintaining lip-sync for video conferences and video delivery. Since individual streams may have very different QOS requirements, it may be more efficient to create separate flows for, say, an audio and video stream rather than multiplexing them into a single packet stream. However, this forces the application to compensate for delay jitter not just within a flow but also between flows. A simple mechanism [18] adjusts the playout delay plus any decoding delay to the maximum of all flows to be synchronized.

The tolerance for packet loss varies widely. Regardless of the level, it is always necessary to make codecs aware of the packetization, so that each packet can be decoded independently [16, 2]. For many continuous-media applications, not only the loss fraction, but also the burstiness of losses matters. For example, if losses are bursty, video frames consisting of several packets may suffer less degradation in quality compared to random losses, as fewer frames are affected. For most audio codecs, on the other hand, bursty losses are more noticeable than random losses, since they disrupt the prediction built into codecs and thus lead to artifacts that are noticeable long after the packet loss burst has subsided.

For applications that are loss-sensitive, such as remote-procedure calls or delivery of stored video, packet loss can also be translated into additional delay, simply by using an ARQ mechanism.

## 1.6 Application Control and Adaptation

### 1.6.1 Bandwidth Adaptation

So far, we have assumed that the network will assure the desired quality of service by reserving appropriate resources to each flow. However, this approach imposes significant costs: First, the network needs to maintain state for each flow; an OC-48 link can easily carry 150,000 audio-rate flows, requiring approximately 75 MB of high-speed memory for storing flowspecs. (Even high-end routers generally can currently only manage about 3,000 flows.) Also, as pointed out earlier, link utilization for the guaranteed service class may be low. In addition, some networks, notably shared-media networks such as non-switched Ethernets and wireless LANs, may not support resource reservation. Finally, because a resource reservation can block a large fraction of the network resources, possibly beyond the capacity of the user's access link, policy control, security and charging mechanisms need to be deployed. Thus, given the practical difficulties, it appears unlikely that resource reservation will be widely deployed for commodity applications in the next few years.

Rather than having applications explicitly reserve resources (and possibly be denied network access), another approach is to have them adapt their requirements to the resources available, possibly combined with a floor guarantee. This is guided by the notion that the utility function for most applications is convex, with a large initial jump at the minimum usable rate. Thus, it is better to have $N+1$ users get $1/(N+1)$ of the bandwidth rather than $N$ users $1/N$, as long $1/(N+1)$ is larger than the minimum acceptable bandwidth.

Routers could use RSVP to signal a lower available bandwidth for existing reservations. However, the necessary flow merging mechanisms are currently not supported by RSVP. Suggested approaches to adaptation thus use packet loss as a congestion indicator. Adapting to the current network conditions requires that the sender has an accurate picture of the quality-of-service conditions among the receiver population that it serves. Receivers also can benefit from obtaining QOS measurements for their fellow receivers, as it allows the application to determine if a QOS problem is likely a local one or widespread. The Real-Time transport Protocol (RTP) [26] is commonly used to deliver continuous media in the Internet. It encompasses a control protocol (RTCP), in which each receiver periodically reports on the packet loss, delay and delay jitter for each sender. Senders report their nominal rate. The reporting interval scales with the number of receivers, so that the traffic due to QOS reports remains below a settable fraction of (typically) 5%. The sender can then use any adaptation algorithm to adjust its sending rate to the current available bandwidth [3, 9], typically with a variation on an additive-increase, multiplicative-decrease algorithm triggered by moving outside given loss thresholds. If the continuous-media application shares bandwidth with an elastic application like TCP, issues of fairness need to addressed. Since the TCP and continuous-media adaptation use very different algorithm, there is the likelihood that either TCP will yield all bandwidth to the continuous media streams or vice versa. For unicast streams, it has also been suggested to use TCP-like congestion control, i.e., emulate slow start and congestion avoidance, but without actually retransmitting packets.

The range and adjustment speed of adaptivity varies widely; many data applications can tolerate zero throughput for a few seconds, with throughput varying at round-trip delay intervals, while changing the audio encodings, quantization factors or frame rate across the whole spectrum at that rate is likely to be rather annoying to the recipient. Thus, the adaptation mechanisms found, for example, in TCP are not applicable to continuous-media applications.

Also, adaptation is difficult in a heterogeneous multicast environment, as this tends to lead to a "race to the bottom", with the poorest-connected received determining the quality of service for all. Instead, it may be necessary to send base and enhancement layers of the content to different multicast groups [19].

If a soft-state reservation mechanism such as RSVP or YESSIR [21] is used, the RSVP PATH or YESSIR forward reservation messages themselves could be used to indicate the "fair share" of available bandwidth on a particular branch of a multicast tree. However, this indication of the minimum available resources still has to be conveyed back to the sender to compute the overall fair share.

### 1.6.2 Delay Adaptation

Continuous media applications are defined by their need to reconstruct the timing of the source at the receiver. With the help of a playout buffer, that is a queue for arriving packets emptied at the playback rate, they convert a variable-delay packet network into a fixed delay. To prevent starvation noticeable as gaps in playback, the playout buffer has to be large enough to compensate for the delay jitter [20, 23]. For best effort and controlled-load service, the delay variation is not bounded, so that the application has to estimate the necessary depth of the buffer, trading off loss due to late arrival against a lower playback delay. With guaranteed service, the delay jitter is bounded, so that a fixed-length playout buffer is sufficient. However, since few packets experience the worst-case delay, the application is better off adapting to the network delays, as long as an occasional packet loss is acceptable. The influence of different scheduling disciplines on playout delay adaptation is not well understood.

### 1.6.3 Loss Adaptation and Concealment

Packets sent at-risk may be lost. For continuous media, the following techniques have been explored to reduce and conceal packet losses: retransmission, redundant transmission, interleaving, and forward error correction. Retransmission is limited by the delay tolerance of the application, but may be useful in regional networks or for delivery of stored media. With redundant transmission, a lower-bitrate version of the same stream is sent at a time-offset. If a packet from the primary encoding is missing, a packet from the lower-rate encoding may be used to fill in the gap [4]. Interleaving distributes a single media block across several packets, turning packet loss into sample loss, presumably easier to conceal by the codec itself. Finally, forward error correction [24, 25] adds redundant parity or Reed-Solomon packets to a group of audio or video packets. It has the advantage of offering perfect reconstruction as long packet losses are isolated and scaling nicely to multicast, but may add substantial overhead. With forward error correction or redundant transmission, decoding delays can be made adaptive, as the receiver only has to wait for the end of the block if a packet was lost. Retransmission and forward error correction may be combined, for example, by retransmitting a parity packet if any packet within a block was lost.

### 1.6.4 Other Service Models

A number of other service models are possible, beyond those offering guaranteed delay bounds or service equivalent to an unloaded network. Recently, there has been interest in services that do not reserve resources, but provide services appropriate, for example, real-time applications. As discussed in the context of delay adaptation, late packets are lost to the application. Thus, it may be desirable to accept excess traffic, but drop packets when they have exceeded their delay bound [27]. As an additional mechanism, one could give scheduling priority to packets that had the bad fortune of experiencing long delays in earlier nodes or still have many nodes to travel [28]. However, these scheduling mechanisms require additional delay information in each packet.

## 1.7 Extensions, Open Issues, and Further Reading

In this last section, we briefly review a number of additional topics of relevance to network service guarantees and provide pointers to additional resources.

The IP networking environment assumed in this chapter will continue to be heterogeneous. As a result, the available end-to-end service guarantees will typically consist of only the intersection of the guarantees offered by the network technologies on the path between end systems. The IETF Integrated Services over Specific Link Layers (ISSLL) working group [9] is addressing the mapping the integrated services onto link layers such as ATM, LANs, and low-speed links.

The IP integrated services model presented here assumes that support for service guarantees was flow or application specific. However, it has been suggested [11, 14] to guarantee performance at the coarser level of a *service class* encompassing many flows. This improves scalability, since the network does not have to maintain state for individual flows. As discussed in Section 1.6, such a model of looser service guarantees may be adequate for adaptive applications (see [13] for its use with TCP). However, it is unlikely to be sufficient for applications that require strong service guarantees, e.g., the control, tracking, haptic, and rendering flows of the TI application of Chapter **??**.

An important component in delivering quality-of-service to users is to control who is entitled to a better service. Addition of such control to the RSVP protocol is being addressed in the IETF Internet Policy working group[10]. This "policy control" mechanism can also be used for other purposes such as allowing

[9] Mailing list address is issll@mercury.lcs.mit.edu.
[10] Mailing list address is ipc@iphighway.com.

a high-priority application to cancel or reduce existing low-priority reservations. This may be useful in the case of emergency conditions, or to support advanced reservations.

Finally, because QoS guarantees do come at a price, networks will have to price their services, at least in part, based on the amount of resources being allocated. However, there are a number of other factors being explored to use pricing as a mechanism to better control networks. Relevant information can be found at the home pages of the *CA$hMAN* project[11] (http://www.uk.infowin.org/ACTS/RUS/PROJECTS/ac039.html), and of the Journal of Electronic Publishing (`http://www.press.umich.edu:80/jep/`) which focuses on Internet-specific activities.

## 1.8 Summary

We have described issues faced by applications when requesting service guarantees from IP networks, and have reviewed the basic components that will be involved in providing those services.

It should be pointed out that the current set of QoS capabilities being deployed represents the "first generation" of such offerings. As such they may not be able to satisfy the requirements of all applications. Furthermore, their introduction has been primarily driven by a number of recent technology and algorithmic developments, rather than to satisfy explicit requirements put forward by applications.

For example, as mentioned in Section 1.4.3, current services do not support QoS specifications across multiple streams, except through the notion of shared reservations. One possible areas to extend this capability, which might be useful to large scale simulations as in the TI application, would be to introduce intelligent discard capabilities for shared reservations.

[11] See also the Home Page of Frank Kelly http://www.statslab.cam.ac.uk/~frank/cashman.html for additional references

# Bibliography


[1] T. Barzilai, D. Kandlur, A. Mehra, D. Saha, and S. Wise. Design and implementation of an RSVP based quality of service architecture for integrated services Internet. In *ICDCS'97*, Baltimore, MD, April 1997. (Extended version to appear in IEEE J. Select. Areas Commun., April 1998).

[2] A. Basso, G. L. Cash, and M. Reha Civanlar. Transmission of MPEG-2 streams over non-guaranteed quality of service networks. In *Proc. of Picture Coding Symposium*, Berlin, Germany, September 1997.

[3] J.-C. Bolot, T. Turletti, and I. Wakeman. Scalable feedback control for multicast video distribution in the internet. In *SIGCOMM*, pages 58–67, London, England, August 1994. ACM.

[4] J.-C. Bolot and A. Vega-Garcia. The case for FEC-based error control for packet audio in the internet. *ACM Multimedia Systems*, 1997.

[5] R. Braden, Ed., L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource ReSerVation protocol (RSVP) – version 1 functional specification. Technical Report RFC 2205, Internet Engineering Task Force, September 1997.

[6] R. Braden and D. Hoffman. RAPI – an RSVP application programming interface, version 5. Internet Draft, Internet Engineering Task Force, June 1997. Work in progress.

[7] R. Braden and L. Zhang. Resource ReSerVation protocol (RSVP) – version 1 message processing rules. Technical Report RFC 2209, Internet Engineering Task Force, September 1997.

[8] P. T. Brady. Effects of transmission delay on conversational behavior on echo-free telephone circuits. *Bell System Tech. J.*, 50(1):115–134, January 1971.

[9] I. Busse, B. Deffner, and H. Schulzrinne. Dynamic QoS control of multimedia applications based on RTP. *Computer Communications*, 19(1):49–58, January 1996.

[10] I. Cidon, R. Gurin, A. Khamisy, and K. N. Sivarajan. Cell versus message level performancees in ATM networks. *Telecommunication Systems*, 5:223–239, 1996.

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

[12] Alan Demers, Srinivasan Keshav, and Scott Shenker. Analysis and simulation of a fair queueing algorithm. *Internetworking: Research and Experience*, 1(1):3–26, January 1990.

[13] W.-C. Feng, Dilip Kandlur, D. Saha, and K. G. Shin. Adaptive packet marking for providing differentiated services in the Internet. Research report, IBM, July 1997.

[14] W.-C. Feng, Dilip Kandlur, D. Saha, and K. G. Shin. Understanding TCP dynamics in an integrated services Internet. In *NOSSDAV'97*, Saint Louis, MO, May 1997.

[15] L. Georgiadis, Roch Guérin, V. Peris, and R. Rajan. Efficient support for delay and rate guarantees in an Internet. In *SIGCOMM*, pages 106–116, Stanford, CA, August 1996.

[16] D. Hoffman, G. Fernando, and V. Goyal. RTP payload format for MPEG1/MPEG2 video. Request for Comments (Proposed Standard) RFC 2038, Internet Engineering Task Force, October 1996.

[17] International Telecommunication Union (ITU). Transmission systems and media, general recommendation on the transmission quality for an entire international telephone connection; one-way transmission time. Recommendation G.114, Telecommunication Standardization Sector of ITU, Geneva, Switzerland, March 1993.

[18] I. Kouvelas, V. Hardman, and A. Watson. Lip synchronisation for use over the internet: Analysis and implementation. In *GLOBECOM*, London, England, November 1996.

[19] S. McCanne, V. Jacobson, and M. Vetterli. Receiver-driven layered multicast. In *SIGCOMM*, Palo Alto, California, August 1996.

[20] S. B. Moon, J. Kurose, and D. Towsley. Packet audio playout delay adjustment algorithms: performance bounds and algorithms. Research report, Department of Computer Science, University of Massachusetts at Amherst, Amherst, Massachusetts, August 1995.

[21] P. P. Pan and H. Schulzrinne. YESSIR: A simple reservation mechanism for the Internet. In *submitted for publication*, 1998.

[22] Abhay 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, Mass., February 1992.

[23] R. Ramjee, J. Kurose, D. Towsley, and H. Schulzrinne. Adaptive playout mechanisms for packetized audio applications in wide-area networks. In *IEEE Infocom*, pages 680–688, Toronto, Canada, June 1994.

[24] L. Rizzo. Effective erasure codes for reliable computer communication protocols. *ACM Comp. Comm. Rev.*, 27(2):24–36, April 1997.

[25] J. Rosenberg and H. Schulzrinne. An A/V profile extension for generic forward error correction in RTP. Internet Draft, Internet Engineering Task Force, August 1997. Work in progress.

[26] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: a transport protocol for real-time applications. Request for Comments (Proposed Standard) RFC 1889, Internet Engineering Task Force, January 1996.

[27] H. Schulzrinne, J. Kurose, and D. Towsley. Congestion control for real-time traffic in high-speed networks. In *IEEE Infocom*, volume 2, pages 543–550, San Francisco, California, June 1990.

[28] H. Schulzrinne, J. Kurose, and D. Towsley. An evaluation of scheduling mechanisms for providing best-effort, real-time communication in wide-area networks. In *IEEE Infocom*, Toronto, Canada, June 1994.

[29] S. Shenker, C. Partridge, and R. Guerin. Specification of guaranteed quality of service. Technical Report RFC 2212, Internet Engineering Task Force, September 1997.

[30] S. Shenker and J. Wroclawski. General characterization parameters for integrated service network elements. Technical Report RFC 2215, Internet Engineering Task Force, September 1997.

[31] J. Wroclawski. Specification of the controlled-load network element service. Technical Report RFC 2211, Internet Engineering Task Force, September 1997.

[32] J. Wroclawski. The use of RSVP with IETF integrated services. Technical Report RFC 2210, Internet Engineering Task Force, September 1997.