Internet Engineering Task Force
INTERNET-DRAFT
draft-ietf-intserv-guaranteed-svc-03.txt

Integrated Services WG
S. Shenker/C. Partridge
Xerox/BBN

15 December 1995
Expires: 5/15/96

Specification of Guaranteed Quality of Service

Status of this Memo

This document is a product of the Integrated Services working group of the Internet Engineering Task Force. Comments are solicited and should be addressed to the working group's mailing list at int-serv@isi.edu and/or the author(s).

This draft reflects changes from the IETF meeting in Dallas.

Abstract

This memo describes the network element behavior required to deliver guaranteed service in the Internet. Guaranteed service provides firm (mathematically provable) bounds on end-to-end packet delays. This specification follows the service specification template described in [1].

Introduction

This document defines the requirements for network elements that support guaranteed service. This memo is one of a series of documents that specify the network element behavior required to support various qualities of service in IP internetworks. Services described in these documents are useful both in the global Internet and private IP networks.

This document is based on the service specification template given in [1]. Please refer to that document for definitions and additional information about the specification of qualities of service within the IP protocol family.

End-to-End Behavior

The end-to-end behavior provided by a series of service elements that conform to this document is an assured level of bandwidth that, when used by a policed flow, produces a delay bounded service with no queueing loss for all conforming datagrams (assuming no failure of network components or changes in routing during the life of the flow).

The end-to-end behavior conforms to the fluid model (described below) in that the delivered delays do not exceed the fluid delays by more than the specified error bounds. More precisely, the end-to-end delay bound is [(b-M)/R*(p-R)/(p-r)]+(M+Ctot)/R+Dtot for p>R, and (M+Ctot)/R+Dtot for p<=R, (where b, r, p, M, R, Ctot, and Dtot are defined later in this document). Guaranteed service does not control the minimal delay of packets, merely the maximal delays.

> NOTE: While the per-hop error terms needed to compute the end-to-end delays are exported by the service module (see Exported Information below), the mechanisms needed to collect per-hop bounds and make the end-to-end quantities Ctot and Dtot known to the applications are not described in this specification. These functions, which can be provided by reservation setup protocols, routing protocols or by other network management functions, are outside the scope of this document.

The end-to-end behavior (as characterized by Ctot and Dtot) provided along a path should be stable. That is, it should not change as long

as the end-to-end path does not change.

This service is subject to admission control.

Motivation

Guaranteed service guarantees that packets will arrive within the guaranteed delivery time and will not be discarded due to queue overflows, provided the flow traffic stays within its specified traffic parameters. This service is intended for applications which need a firm guarantee that a packet will arrive no later than a certain time after it was transmitted by its source. For example, some audio and video "play-back" applications are intolerant of any packet arriving after their play-back time. Applications that have hard real-time requirements will also require guaranteed service.

This service does not attempt to minimize the jitter (the difference between the minimal and maximal packet delays); it merely controls the maximal delay. Because the guaranteed bound is a firm one, it must be large enough to cover extremely rare cases of long queueing delays. Several studies have shown that the actual delay for the vast majority of packets can be far lower than the guaranteed delay. Therefore, authors of playback applications should note that packets will often arrive far earlier than the delivery deadline and will have to be buffered at the receiving system until it is time for the application to process them.

This service represents one extreme end of delay control for networks. Most other services providing delay control provide much weaker assurances about the resulting delays. In order to provide this high level of assurance, guaranteed service is typically only useful if provided by every network element along the path. Moreover, as described in the Exported Information section, effective provision and use of the service requires that the set-up protocol used to request service provides service characterizations to intermediate routers and to the endpoints.

Network Element Data Handling Requirements

The network element must ensure that the service approximates the "fluid model" of service. The fluid model at service rate R is essentially the service that would be provided by a dedicated wire of bandwidth R between the source and receiver. Thus, in the fluid model of service at a fixed rate R, the flow's service is completely independent of that of any other flow.

The flow's level of service is characterized at each network element by a bandwidth (or service rate) R and a buffer size B. R represents the share of the link's bandwidth the flow is entitled to and B represents the buffer space in the router that the flow may consume. The network element must ensure that its service matches the fluid

model at that same rate to within a sharp error bound.

The definition of guaranteed service relies on the result that the fluid delay of a flow obeying a token bucket (r,b) and being served by a line with bandwidth R is bounded by b/R as long as R is no less than r. Guaranteed service with a service rate R, where now R is a share of bandwidth rather than the bandwidth of a dedicated line, approximates this behavior.

More specifically, the network element must ensure that the delay of any packet be less than b/R+C/R+D, where C and D describe the maximal deviation away from the fluid model. It is important to emphasize that C and D are maximums. So, for instance, if an implementation has occasional gaps in service (perhaps due to processing routing updates), D needs to be large enough to account for the time a packet may lose during the gap in service.

Links are not permitted to fragment packets as part of guaranteed service. Packets larger than the MTU of the link must be policed as nonconformant which means that they will be policed according to the rules described in the Policing section below.

Invocation Information

Guaranteed service is invoked by specifying the traffic (TSpec) and the desired service (RSpec) to the network element. A service request for an existing flow that has a new TSpec and/or RSpec should be treated as a new invocation, in the sense that admission control must be reapplied to the flow. Flows that reduce their TSpec and/or

their RSpec (i.e., their new TSpec/RSpec is strictly smaller than the old TSpec/RSpec according to the ordering rules described in the section on Ordering below) should never be denied service.

The TSpec takes the form of a token bucket plus a peak rate (p), a minimum policed unit (m), and a maximum packet size (M).

The token bucket has a bucket depth, b, and a bucket rate, r. Both b and r must be positive. The rate, r, is measured in bytes of IP datagrams per second, and can range from 1 byte per second to as large as 40 terabytes per second (or about what is believed to be the maximum theoretical bandwidth of a single strand of fiber). Clearly, particularly for large bandwidths, only the first few digits are significant and so the use of floating point representations, accurate to at least 0.1% is encouraged.

The bucket depth, b, is also measured in bytes and can range from 1 byte to 250 gigabytes. Again, floating point representations accurate to at least 0.1% are encouraged.

The range of values is intentionally large to allow for the future bandwidths. The range is not intended to imply that a network element must support the entire range.

The peak rate, p, is measured in bytes of IP datagrams per second and has the same range and suggested representation as the bucket rate. The peak rate is the maximum rate at which the source and any reshaping points (reshaping points are defined below) may inject bursts of traffic into the network. More precisely, it is the requirement that for all time periods the amount of data sent cannot exceed M+pT where M is the maximum packet size and T is the length of the time period. Furthermore, p must be greater than or equal to the token bucket rate, r. If the peak rate is unknown or unspecified, then p is set to infinity, which in the IEEE floating point format corresponds to an exponent of all ones (255) and a sign bit and mantissa of all zeroes.

The minimum policed unit, m, is an integer measured in bytes. All IP datagrams less than size m will be counted, when policed and tested for conformance to the TSpec, as being of size m. The maximum packet size, M, is the biggest packet that will conform to the traffic specification; it is also measured in bytes. The flow must be

rejected if the requested maximum packet size is larger than the MTU of the link. Both m and M must be positive, and m must be less than or equal to M.

The RSpec is a rate R and a slack term S, where R must be greater than or equal to r and S must be nonnegative. The RSpec rate can be bigger than the TSpec rate because higher rates will reduce queueing delay. The slack term signifies the difference between the desired delay and the delay obtained by using a reservation level R. This slack term can be utilized by the service element to reduce its resource reservation for this flow. When a service element chooses to utilize some of the slack in the RSpec, it must follow specific rules in updating the R and S fields of the RSpec; these rules are specified in the Ordering and Merging section. If at the time of service invocation no slack is specified, the slack term, S, is set to zero. No buffer specification is included in the RSpec because the service element is expected to derive the required buffer space to ensure no queueing loss from the token bucket and peak rate in the TSpec, the reserved rate and slack in the RSpec, combined with internal information about how the element manages its traffic.

The TSpec can be represented by three floating point numbers in single-precision IEEE floating point format followed by two 32-bit integers in network byte order. The first value is the rate (r), the second value is the bucket size (b), the third is the peak rate (p), the fourth is the minimum policed unit (m), and the fifth is the

maximum packet size (M).

The RSpec rate, R, and the slack term, S, can also be represented using single-precision IEEE floating point.

For all IEEE floating point values, the sign bit must be zero. (All values must be positive). Exponents less than 127 (i.e., 0) are prohibited. Exponents greater than 162 (i.e., positive 35) are discouraged, except for specifying a peak rate of infinity.

Exported Information

Each guaranteed service module must export at least the following information. All of the parameters described below are

characterization parameters.

A network elements implementation of guaranteed service is characterized by two error terms, C and D, which represent how the element's implementation of the guaranteed service deviates from the fluid model. These two parameters have an additive composition rule.

If the composition function is applied along the entire path to compute the end-to-end sums of C and D (Ctot and Dtot) and the resulting values are then provided to the end nodes (by presumably the setup protocol), the end nodes can compute the maximal packet delays. Moreover, if the partial sums (Csum and Dsum) from the most recent reshaping point (reshaping points are defined below) downstream towards receivers are handed to each network element then these network elements can compute the buffer allocations necessary to achieve no packet loss, as detailed in the section Guidelines for Implementors. The proper use and provision of this service requires that the quantities Ctot and Dtot, and the quantities Csum and Dsum be computed. Therefore, we assume that usage of guaranteed service will be primarily in contexts where these quantities are made available to end nodes and network elements.

The error term C is measured in units of bytes. An individual element can advertise a C value between 1 and 2**28 (a little over 250 megabytes) and the total added over all elements can range as high as (2**32)-1. Should the sum of the different elements delay exceed (2**32)-1, the end-to-end error term should be (2**32)-1.

The error term D is measured in units of one microsecond. An individual element can advertise a delay value between 1 and 2**28 (somewhat over two minutes) and the total delay added all elements can range as high as (2**32)-1. Should the sum of the different elements delay exceed (2**32)-1, the end-to-end delay should be

(2**32)-1.

The guaranteed service is service_name 2.

Error characterization parameter C is numbered 1 and parameter D is numbered 2.

The end-to-end composed value (Ctot) for C is numbered 3 and the

end-to-end composed value for D (Dtot) is numbered 4.

The since-last-reshaping point composed value (Csum) for C is numbered 5 and the since-last-reshaping point composed value for D (Dsum) is numbered 6.

No other exported data is required by this specification.

Policing

Policing is done at the edge of the network, at all heterogeneous source branch points and at all source merge points. A heterogeneous source branch point is a spot where the multicast distribution tree from a source branches to multiple distinct paths, and the TSpec's of the reservations on the various outgoing links are not all the same. Policing need only be done if the TSpec on the outgoing link is "less than" (in the sense described in the Ordering section) the TSpec reserved on the immediately upstream link. A source merge point is where the multicast distribution trees from two different sources (sharing the same reservation) merge. It is the responsibility of the invoker of the service (a setup protocol, local configuration tool, or similar mechanism) to identify points where policing is required. Policing may be done at other points as well as those described above.

The token bucket and peak rate parameters require that traffic must obey the rule that over all time periods, the amount of data sent cannot exceed M+min[pT, rT+b-M], where r and b are the token bucket parameters, M is the maximum packet size, and T is the length of the time period (note that when p is infinite this reduces to the standard token bucket requirement). For the purposes of this accounting, links must count packets which are smaller than the minimal policing unit to be of size m. Packets which arrive at an element and cause a violation of the the M+min[pT, rT+b-M] bound are considered non-conformant. Policing to conformance with this token bucket is done in two different ways.

At the edge of the network, non-conforming packets are treated as best-effort datagrams. [If and when a marking ability becomes

available, these non-conformant packets should be ''marked'' as being

non-compliant and then treated as best effort packets at all subsequent routers.]

> NOTE: There may be situations outside the scope of this document, such as when a service module's implementation of guaranteed service is being used to implement traffic sharing rather than a quality of service, where the desired action is to discard non-conforming packets. To allow for such uses, implementors should ensure that the action to be taken for non-conforming packets is configurable.

Inside the network, this approach does not produce the desired results, because queueing effects will occasionally cause a flow's traffic that entered the network as conformant to be no longer conformant at some downstream network element. Therefore, inside the network, service elements must reshape traffic before applying the token bucket test. Reshaping entails delaying packets until they are within conformance of the TSpec.

Reshaping is done by combining a buffer with a token bucket and peak rate regulator and buffering data until it can be sent in conformance with the token bucket and peak rate parameters. (The token bucket regulator should start with its token bucket full of tokens). Under guaranteed service, the amount of buffering required to reshape any conforming traffic back to its original token bucket shape is b+Csum+(Dsum*r), where Csum and Dsum are the sums of the parameters C and D between the last reshaping point and the current reshaping point. Note that the above buffer requirement is an upper bound that can be significantly reduced if the cumulative latency [7] from the last reshaping point is known. More precisely, in the above formula Dsum can be replaced by Dsum - (cumulative latency). In addition, the knowledge of the peak rate at the reshapers can also be used to further reduce the buffer requirements. A network element must provide the necessary buffers to ensure that conforming traffic is not lost at the reshaper.

If a datagram arrives to discover the reshaping buffer is full, then the datagram is non-conforming. Observe this means that a reshaper is effectively policing too. As with a policer, the reshaper should relegate non-conforming datagrams to best effort. [If marking is available, the non-conforming datagrams should be marked]

> NOTE: As with policers, it should be possible to configure how reshapers handle non-conforming packets.

Note that while the large buffer makes it appear that reshapers add considerable delay, this is not the case. Given a valid TSpec that

accurately describes the traffic, reshaping will cause little extra delay at the reshaping point. However, if the TSpec is smaller than the actual traffic, reshaping will cause a large queue to develop at the reshaping point, which both causes substantial additional delays and forces some datagrams to be treated as non-conforming. This scenario makes an unpleasant denial of service attack possible, in which a receiver who is successfully receiving a flow's traffic via best effort service is pre-empted by a new receiver who requests a reservation for the flow, but with an inadequate TSpec and RSpec. The flow's traffic will now be policed and possibly reshaped. If the policing function was chosen to discard datagrams, the best-effort receiver would stop receiving traffic. For this reason, in the normal case, policers are simply to mark packets as best effort. While this protects against denial of service, it is still true that the bad TSpec may cause queueing delays to increase.

> NOTE: To minimize problems of reordering datagrams, reshaping points may wish to forward a best-effort datagram from the front of the reshaping queue when a new datagram arrives and the reshaping buffer is full.
>
> Readers should also observe that reclassifying datagrams as best effort also makes support for elastic flows easier. They can reserve a modest token bucket and when their traffic exceeds the token bucket, the excess traffic will be sent best effort.

A related issue is that at all network elements, packets bigger than the MTU of the network element must be considered non-conformant and should be classified as best effort (and will then either be fragmented or dropped according to the element's handling of best effort traffic). [Again, if marking is available, these reclassified packets should be marked.]

Ordering and Merging

TSpec's are ordered according to the following rule: TSpec A is a substitute ("as good or better than") for TSpec B if (1) both the token rate r and bucket depth b for TSpec A are greater than or equal to those of TSpec B, (2) the peak rate p is at least as large in TSpec A as it is in TSpec B. (3) the minimum policed unit m is at least as small for TSpec A as it is for TSpec B, and (4) the maximum packet size M is at least as large for TSpec A as it is for TSpec B.

A merged TSpec may be calculated over a set of TSpecs by taking the largest token bucket rate, largest bucket size, largest peak rate,

smallest minimal policed unit, and largest maximum packet size across all members of the set. This use of the word "merging" is similar to

that in the RSVP protocol; a merged TSpec is one which is adequate to describe the traffic from any one of a number of flows.

The RSpec's are merged in a similar manner as the TSpecs, i.e. a set of RSpecs is merged onto a single RSpec by taking the largest rate R, and the smallest slack S. More precisely, RSpec A is a substitute for RSpec B if the value of reserved service rate, R, in RSpec A is greater than or equal to the value in RSpec B, and the value of the slack, S, in RSpec A is smaller than or equal to that in RSpec B.

Each network element receives a service request of the form (TSpec, RSpec), where the RSpec is of the form (Rin, Sin). The network element processes this request and performs one of two actions:

a. it accepts the request and returns a new Rspec of the form (Rout, Sout);
b. it rejects the request.

The processing rules for generating the new RSpec are governed by the delay constraint:

$$Sout + b/Rout + Ctoti/Rout <= Sin + b/Rin + Ctoti/Rin,$$

where Ctoti is the cumulative sum of the error terms, C, for all the network elements that are upstream of the current element, i. In other words, this element consumes (Sin - Sout) of slack and can use it to reduce its reservation level, provided that the above inequality is satisfied. Rin and Rout must also satisfy the constraint:

$$r <= Rout <= Rin.$$

When several RSpec's, each with rate Rj, j=1,2..., are to be merged at a split point, the value of Rout is the maximum over all the rates Rj, and the value of Sout is the minimum over all the slack terms Sj.

Guidelines for Implementors

This section discusses a number of important implementation issues in no particular order.

It is important to note that individual subnetworks are service elements and both routers and subnetworks must support the guaranteed service model to achieve guaranteed service. Since subnetworks typically are not capable of negotiating service using IP-based protocols, as part of providing guaranteed service, routers will have to act as proxies for the subnetworks they are attached to.

In some cases, this proxy service will be easy. For instance, on leased line, the proxy need simply ensure that the sum of all the flows' RSpec rates does not exceed the bandwidth of the line, and needs to advertise the serialization and transmission delays of the link as the values of C and D.

In other cases, this proxy service will be complex. In an ATM network, for example, it may require establishing an ATM VC for the flow and computing the C and D terms for that VC. Readers may observe that the token bucket and peak rate used by guaranteed service map directly to the Sustained Cell Rate, Burst Size, and Peak Cell Rate of ATM's Q.2931 QoS parameters for Variable Bit Rate traffic.

The assurance that packets will not be lost is obtained by setting the router buffer space B to be equal to the token bucket b plus some error term (described below).

Another issue related to subnetworks is that the TSpec's token bucket rates measure IP traffic and do not (and cannot) account for link level headers. So the subnetwork service elements must adjust the rate and possibly the bucket size to account for adding link level headers. Tunnels must also account for the additional IP headers that they add.

For packet networks, a maximum header rate can usually be computed by dividing the rate and bucket sizes by the minimum policed unit. For networks that do internal fragmentation, such as ATM, the computation may be more complex, since one must account for both per-fragment overhead and any wastage (padding bytes transmitted) due to mismatches between packet sizes and fragment sizes. For instance, a conservative estimate of the additional data rate imposed by ATM AAL5

plus ATM segmentation and reassembly is

$$((r/48)*5)+((r/m)*(8+52))$$

which represents the rate divided into 48-byte cells multiplied by the 5-byte ATM header, plus the maximum packet rate (r/m) multiplied by the cost of the 8-byte AAL5 header plus the maximum space that can be wasted by ATM segmentation of a packet (which is the 52 bytes wasted in a cell that contains one byte). But this estimate is likely to be wildly high, especially if m is small, since ATM wastage is usually much less than 52 bytes. (ATM implementors should be warned that the token bucket may also have to be scaled when setting the VC parameters for call setup and that this example does not account for overhead incurred by encapsulations such as those specified in RFC 1483).

To ensure no loss, service elements will have to allocate some buffering for bursts. If every hop implemented the fluid model perfectly, this buffering would simply be b (the token bucket size). However, as noted in the discussion of reshaping earlier, implementations are approximations and we expect that traffic will become more bursty as it goes through the network. However, as with shaping the amount of buffering required to handle the burstiness is bounded by b+Csum+Dsum*R. If one accounts for the peak rate, this can be further reduced to

$$M + (b-M)(p-X)/(p-r) + (Csum/R + Dsum)X$$

where X is set to r if (b-M)/(p-r) is less than Csum/R+Dsum and X is R if (b-M)/(p-r) is greater than or equal to Csum/R+Dsum and p>R; otherwise, X is set to p. This reduction comes from the fact that the peak rate limits the rate at which the burst, b, can be placed in the network. As before, the buffer requirements can be lowered by subtracting from Dsum the propagation delay since the last reshaping point, if it is known. Conversely, if a non-zero slack term, Sout, is returned by the network element, the buffer requirements are increased by adding Sout to Dsum.

While sending applications are encouraged to set the peak rate parameter and reshaping points are required to conform to it, it is always acceptable to ignore the peak rate for the purposes of

computing buffer requirements and end-to-end delays. The result is simply an overestimate of the buffering and delay. As noted above, if the peak rate is unknown (and thus potentially infinite), the buffering required is b+Csum+Dsum*R. The end-to-end delay without the peak rate is b/R+Ctot/R+Dtot.

The parameter D at each service element should be set to the maximum packet transfer delay (independent of bucket size) through the service element. For instance, in a simple router, one might compute the worst case amount of time it make take for a datagram to get through the input interface to the processor, and how long it would take to get from the processor to the outbound interface (assuming the queueing schemes work correctly). For an Ethernet, it might represent the worst case delay if the maximum number of collisions is experienced.

The parameter C is the data backlog resulting from the vagaries of how a specific implementation deviates from a strict bit-by-bit service. So, for instance, for packetized weighted fair queueing, C is set to M.

If a network element uses a certain amount of slack, Si, to reduce the amount of resources that it has reserved for a particular flow,

i, the value Si should be stored at the network element. Subsequently, if reservation refreshes are received for flow i, the network element must use the same slack Si without any further computation. This guarantees consistency in the reservation process.

As an example for the use of the slack term, consider the case where the required end-to-end delay, Dreq, is larger than the maximum delay of the fluid flow system. The latter is obtained by setting R=r in the fluid delay formula, and is given by

$$b/r + Ctot/r + Dtot.$$

In this case the slack term is

$$S = Dreq - (b/r + Ctot/r + Dtot).$$

The slack term may be used by the network elements to adjust their local reservations, so that they can admit flows that would otherwise

have been rejected. A service element at an intermediate network element that can internally differentiate between delay and rate guarantees can now take advantage of this information to lower the amount of resources allocated to this flow. For example, by taking an amount of slack s <= S, an RCSD scheduler [5] can increase the local delay bound, d, assigned to the flow, to d+s. Given an RSpec, (Rin, Sin), it would do so by setting Rout = Rin and Sout = Sin - s.

Similarly, a network element using a WFQ scheduler can decrease its local reservation from Rin to Rout by using some of the slack in the RSpec. This can be accomplished by using the transformation rules given in the previous section, that ensure that the reduced reservation level will not increase the overall end-to-end delay.

Evaluation Criteria

The scheduling algorithm and admission control algorithm of the element must ensure that the delay bounds are never violated. Furthermore, the element must ensure that misbehaving flows do not affect the service given to other flows. Vendors are encouraged to formally prove that their implementation is an approximation of the fluid model.

Examples of Implementation

Several algorithms and implementations exist that approximate the fluid model. They include Weighted Fair Queueing (WFQ) [2], Jitter-EDD [3], Virtual Clock [4] and a scheme proposed by IBM [5]. A nice theoretical presentation that shows these schemes are part of a large

class of algorithms can be found in [6].

Examples of Use

Consider an application that is intolerant of any lost or late packets. It uses the advertised values Ctot and Dtot and the TSpec of the flow, to compute the resulting delay bound from a service request with rate R. Assuming R < p, it then sets its playback point to [(b-M)/R*(p-R)/(p-r)]+(M+Ctot)/R+Dtot.

Security Considerations

This memo discusses how this service could be abused to permit denial of service attacks. The service, as defined, does not allow denial of service (although service may degrade under certain circumstances).

Acknowledgements

The authors would like to gratefully acknowledge the help of the INT SERV working group. We would also like to expressly acknowledge the help of several people who helped us ensure the mathematics of this document were correct [TBA].

References

[1] S. Shenker and J. Wroclawski. "Network Element Service Specification Template", Internet Draft, June 1995, <draft-ietf-intserv-svc-template-01.txt>

[2] A. Demers, S. Keshav and S. Shenker, "Analysis and Simulation of a Fair Queueing Algorithm," in Internetworking: Research and Experience, Vol 1, No. 1., pp. 3-26.

[3] L. Zhang, "Virtual Clock: A New Traffic Control Algorithm for Packet Switching Networks," in Proc. ACM SIGCOMM '90, pp. 19-29.

[4] D. Verma, H. Zhang, and D. Ferrari, "Guaranteeing Delay Jitter Bounds in Packet Switching Networks," in Proc. Tricomm '91.

[5] L. Georgiadis, R. Guerin, V. Peris, and K. N. Sivarajan, "Efficient Network QoS Provisioning Based on per Node Traffic Shaping," IBM Research Report No. RC-20064.

[6] P. Goyal, S.S. Lam and H.M. Vin, "Determining End-to-End Delay Bounds in Heterogeneous Networks," in Proc. 5th Intl. Workshop on

Network and Operating System Support for Digital Audio and Video, April 1995.

[7] S. Shenker. "Specification of General Characterization

Parameters", Internet Draft, November 1995, <draft-ietf-intserv-charac-01.txt>

Authors' Address:

Scott Shenker
Xerox PARC
3333 Coyote Hill Road
Palo Alto, CA 94304-1314
shenker@parc.xerox.com
415-812-4840
415-812-4471 (FAX)

Craig Partridge
BBN
2370 Amherst St
Palo Alto CA 94306
craig@bbn.com