

Tutorial on Internet Services for SURFnet4 Infrastructure Project 1998

P. F. Chimento *

Abstract

This document contains a tutorial on the recent work in the area of offering services in the Internet. We cover the principles and some of the details of Integrated Services and Differentiated Services. We also explain the relationship to RSVP of each of these architectures. Finally, we draw some conclusions about the relationship of these architectures to SURFnet.

1 Introduction

The IETF is working on services that go beyond the basic best-effort service of the current Internet. These services require more sophisticated handling of packets, in addition to (possibly) admission control procedures and policy control applications that work together to determine who may ask for the advanced services and when, and who is granted the service.

There are several different approaches to providing these advanced services, and the IETF is currently exploring two of them. In this document, we will discuss integrated services, which require some protocol to communicate with the routers along the path of the intended flow and reserve the resources necessary to provide the service and differentiated services, a weaker method of distinguishing different flows of packets and requires some policing, but does not necessarily require the state information associated with the *flows* defined by integrated services. Differentiated services works on the TOS field (IPv4) or the Traffic Class field (IPv6) in the packet headers. It can thus work more or less on a packet by packet basis, rather than keep extensive state information about flows.

2 Integrated Services

Integrated services has currently defined in two RFCs two levels of service for IP, in addition to the normal best-effort service that is available today. These are called *Controlled-Load Service* and *Guaranteed Service*. Both of these kinds of service require specification by the endpoint of the characteristics of the traffic that will be produced, and further require special handling of the packets to which the service is applied by the network elements (routers, e.g.).

In the following sections, we will discuss the definitions and characteristics of these services and how they could be implemented in the Internet.

2.1 Architecture

As is usual in the Internet, there are a number of options for implementing a function. For integrated services, some information has to be communicated between

*This research was funded by a contract from SURFnet b.v.

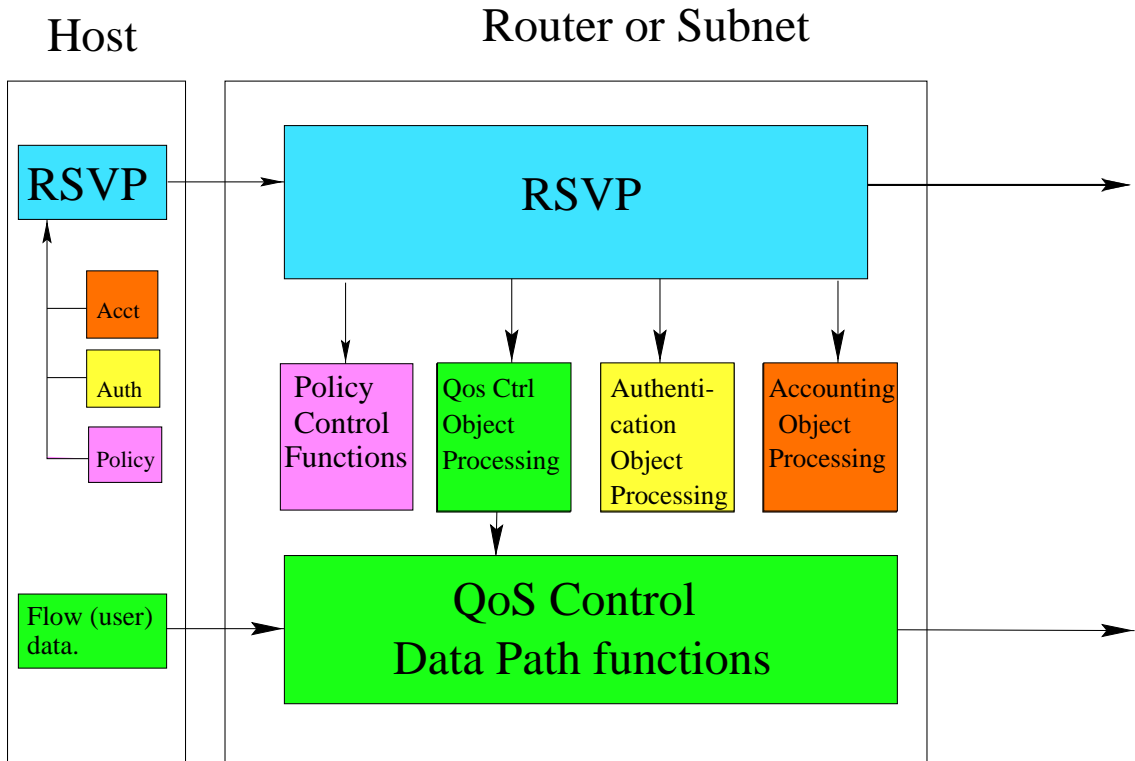


Figure 1: High level architecture for integrated services

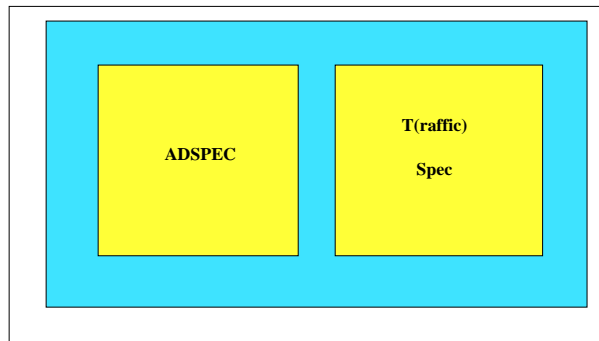
the end systems and the network elements (routers and subnets). A logical choice for this communication is RSVP, but that is not the only possibility. Some other set up protocol (equivalent in functionality) or management procedures may be used to perform the same functions. In this document, we will assume that this communication takes place via RSVP, but architecturally, RSVP and integrated services do not depend on one another.

Figure 1 shows part of the architecture of an integrated services system. Note that the RSVP is only a protocol that carries (among other things) the invocation of some QoS functions. This is communicated to the functions within the router/subnet which actually deliver the QoS. RSVP itself is not aware of the semantics of the objects that it carries. Note that other relationships exist between these types of information than are shown in this picture. For example, the admission control function would use not only policy objects carried by RSVP, but also the objects used to invoke the QoS functions and possibly even the authentication and accounting objects.

The main point is that the invocation and provision of the QoS functions are different, and further, that RSVP itself does not provide any functions, but only communication services. So, the provision of integrated services (and differentiated services) is independent of RSVP itself.

While acknowledging that integrated services can be set up in various ways, it is nonetheless handy to describe the information needed for the various kinds of service in terms of sets of parameters that (possibly) will flow in RSVP messages.

We begin by looking at Figure 2 which shows some of the data objects found in the PATH message. In RSVP, the PATH message flows from the sender to the receiver (i.e. downstream) and contains information about the traffic stream coming



Objects in the PATH message

Figure 2: Objects in the sender's PATH message

from the sender.

The traffic stream itself is described by the TSpec of the sender. The TSpec does not change, but signals to all intermediate nodes and the destination what the characteristics of the stream are. The ADSPEC can be originated by the sending host or the first router in the path, and contains a number of general parameters characterizing the path which are determined by hop-by-hop computations, and also parameters characterizing particular QoS services supported by the network elements. Examples of the general parameters are: Path MTU size, path bandwidth, path minimum latency, etc.[17].

Figure 3 shows the format of a flow descriptor in RSVP. A flow descriptor is made up of two parts: the flow spec and the filter spec. The flow descriptor is found in the RESV message which flows from the receiver to the sender (i.e. upstream).

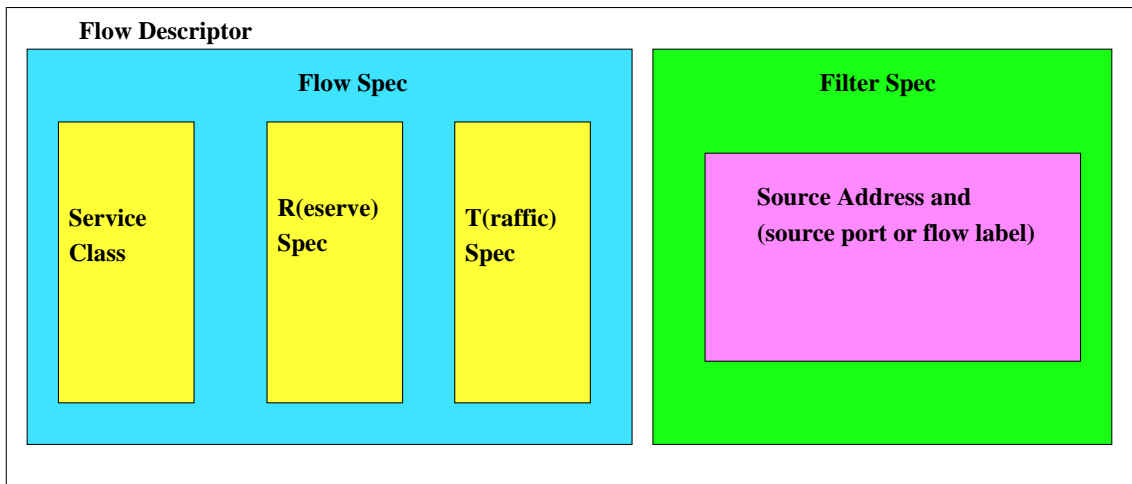
The filter spec is input to the packet classifier which selects substreams of the packet stream seen by a particular link which have requested a particular kind of service. The flow spec is made up of three parts which state: the type of service requested, the parameters of the service (the RSpec) and the parameters of the flow requesting the service (the TSpec). We will discuss these parameters more in some detail when we discuss specific kinds of service offered by IP.

The filter spec serves a purely logical packet discrimination function. Based either on the flow label (in an IPv6 packet) or the combination source address and source port (in an IPv4 packet) a substream of packets can be selected to which the service is applied.

The flow spec contains the specific values of the parameters that make up the traffic characteristics of the flow and the specific values of the parameters of the service being requested.

There is more detail in the QoS control data path functions, as shown in Figure 4. Here we see the information from RSVP being passed to the various functions that must exist in the router in order for QoS to be provided. We see, first of all, packet classifier and packet scheduler functions. The packet classifier determines how the packets will be handled. The packet scheduler applies the particular mechanisms to the packets in order to provide the quality of service requested.

Other important functional blocks play a role. The policy control and admission control functions work together to determine whether the particular flow is permitted to request this particular service at this particular time (policy) and whether there exist enough resources in the network element to support the service requested by this flow (admission). The results of the application of these functions will of



Objects in an RSVP RESV message

Figure 3: Format of a Flow Descriptor

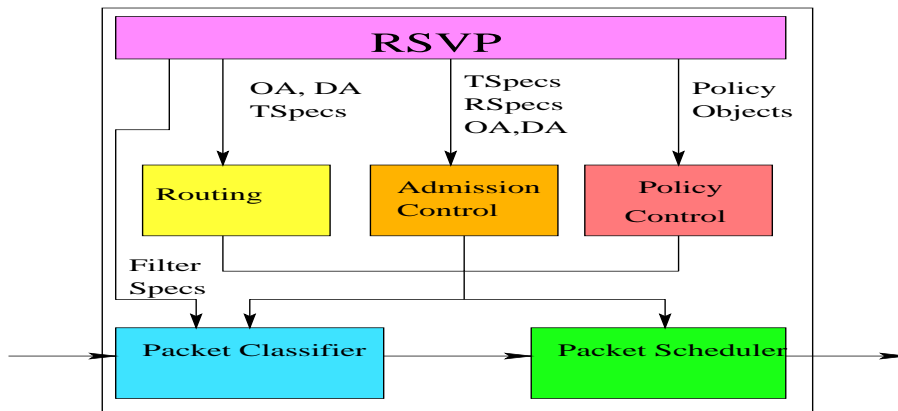


Figure 4: Relationship of RSVP information to router packet handling

course influence the operation of the packet classifier. In addition, the RSVP information can be used to influence the routing algorithms that are running in the network element (e.g. QoS routing).

2.2 Controlled-Load Service

2.2.1 Service Description

Controlled load service was created for a certain class of applications which have a “real time” character, but can adapt somewhat to network conditions, but which tend to perform badly on loaded networks. Controlled load service attempts to create “unloaded” network conditions for such applications, but does not provide service guarantees. Such applications could be audio and video for example.

Controlled load service addresses specifically two QoS parameters: packet loss, and delay. The goals of this service are that the packet loss rate “closely approximates” that of the medium and that the transit delay through network elements is close to the minimum possible transit delay (i.e. no queues).

The QoS goals of the service are deliberately non-specific. The assumption of the RFC [16] is that disruptions of service last only as long as a *burst length* of the flow that is receiving the service. The RFC assumes that if the network elements fail to provide service for a longer period than that, then there is a problem with the resource allocation scheme used by the network element.

2.2.2 Architecture

Figure 5 shows a network element which is providing controlled load service. Specifically, what is depicted is the packet handling function for the normal data path. This is one possible design out of many. Note that here all packet streams flow through the packet classifier function in the router. This determines which packets are eligible for controlled load service and which are not. At the exit of the packet classifier, there is a token bucket. The token bucket mechanism is described in [14]. The token bucket mechanism is used to determine which of the packets of the flow identified by the packet classifier are eligible to receive the controlled load service.

In the design shown in Figure 5 there are two queues. The upper queue has priority and packets which conform to the token bucket for this flow are put in this higher priority queue. The lower queue is shared between packets which do not conform to the token bucket for the flow and packets which receive only best-effort service. Note that the lower queue can be (conceptually, at least) divided into two queues.

This brings up a number of issues treated in the [16]. First, is the issue of how non-conforming traffic is treated. The RFC says that non-conforming controlled load service traffic should be treated as best-effort traffic. So in Figure 5 we see the excess going into the best-effort queue. However, this excess traffic should not adversely affect the normal best effort service; for example, it should not cause significant degradation of normal best-effort service. The immediate problem is that if there is a lot of non-conformant controlled load traffic, it could cause best effort traffic (e.g. TCP) to back off because of packet losses from the best-effort queue. So there has to be some mechanism in place to handle excess traffic (i.e. non-conforming traffic) from controlled load service.

There are a lot of possibilities for such mechanisms, including having another level of priority queue for the excess controlled load traffic, so that best-effort traffic has preferential treatment or using weighted fair queueing or class based queueing to ensure that best effort traffic is not disturbed by this excess traffic.

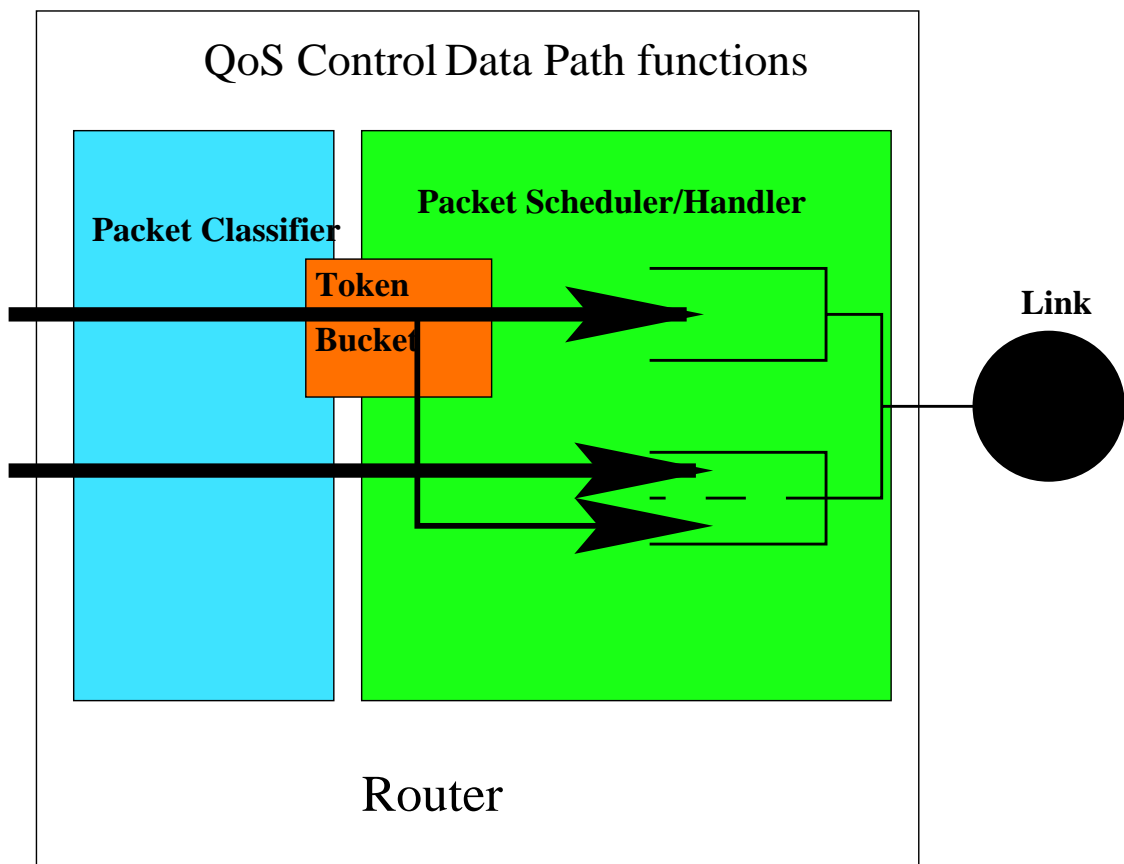


Figure 5: Controlled Load Service

Another issue is the allocation of resources in the high-priority (controlled load traffic) queue; that is to say, how do multiple controlled load flows share the high priority queue and how does one make admission control decisions about these flows.

2.2.3 Mechanisms

The controlled load service is invoked for a flow by the application causing RSVP to send a SENDER_TSPEC with a request for controlled load service in an RSVP PATH message. RSVP also sends an ADSPEC in the PATH message which contains, among other things, an indication of whether controlled load service is implemented at every hop on the path, the minimum MTU size and contains other parameters which characterize the path taken by the flow's data packets. The SENDER_TSPEC is not modified by any intermediate nodes, but the ADSPEC may be. When the receiver sees the PATH message, it processes it and initiates a RESERVE message. When intermediate nodes process the PATH message, for controlled load service, they only indicate whether or not the service is supported by setting the so-called 'break bit'. The 'break bit' lets everyone know that there is at least one node in the path which does not support the service.

It is important to note that there is no precise (quantitative) definition of controlled load service. Each network element then must decide (independently) whether it can provide this service given its current load and the degree of risk (multiplexing) it is willing to take. When we say *risk* here, we mean that by over-committing (beyond the sum of the TSPECs of the controlled load flows), the network element may find itself in a situation where it cannot provide the controlled load service that it has agreed to provide.

Once the flow has been admitted, the data path functions take over. The data path functions take care of the handling and scheduling of packets. Here we should note an aspect of the behaviour of the token bucket ([14], [16]): Suppose that the flow bursts up to the number of packets allowed by the token bucket, and thereafter sends at exactly the token rate specified in the TSPEC. If the node chooses to service the flow at exactly the token rate, there will be a "permanent" backlog of packets in the queue at that node. Therefore the RFC recommends that there be some mechanism in the node to allow such a backlog to clear thereby reducing the queueing time for that flow. In a similar way, because we know that traffic flowing through multiple nodes does not retain its original shape and characteristics, if a node sets the maximum buffer size to exactly the burst size specified in the TSPEC, there will likely be (unnecessary) losses due to the fact that traffic is not reshaped at each node in the network. The RFC here also recommends flexibility in the buffer allocation scheme so that extra buffers are available (at least on a statistical basis) for bursts.

The node determines the conformance of the flow via the token bucket mechanism parameterized by the TSPEC. The handling of non-conforming packets has some latitude, but is not permitted to degrade the service provided to other controlled load flows and should not (but possibly could) degrade best-effort service. However, one option for treating flows with non-conformant traffic is to degrade the service for the entire flow (not just for the non-conforming packets). Another possibility is to reshape the flow so that non-conforming packets wait either until they conform or up to some predefined time limit. Another way of handling non-conforming packets is to queue them with the best effort traffic (or behind the best effort traffic). This, of course, runs the risk of having packets for the flow delivered out of order causing further delays for the higher layer protocols (e.g. TCP).

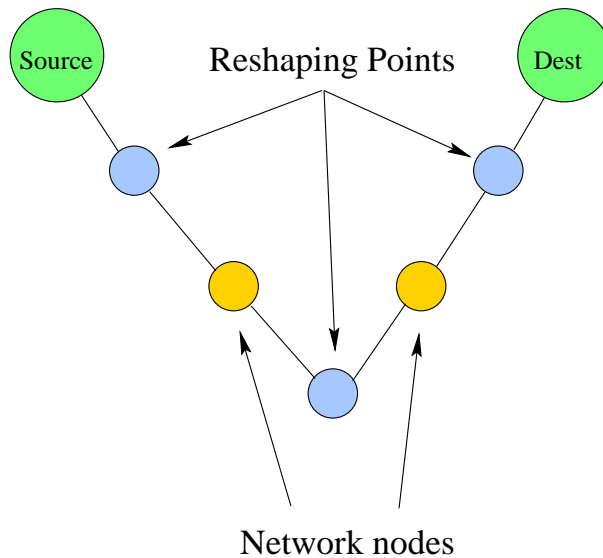


Figure 6: Nodes providing guaranteed service

2.3 Guaranteed Service

Guaranteed service in IP is described in RFC 2212 ([15]). The purpose of guaranteed service is to provide bounded delay and assured delivery of all packets which fall within a given specification for applications that need such service. This section discusses the parameters of guaranteed service, how it is invoked and the mechanisms used to provide it. An essential part of guaranteed service is that the parameters from all the network elements in a particular path have to be able to be combined in a simple way to predict (bounds on) the QoS that will be experienced by the packets in that flow.

The active elements in guaranteed service are shown in Figure 6.

The reservations are uni-directional, so for bidirectional communication, two reservations must take place. Figure 6 shows one direction, where the source produces a traffic stream with certain characteristics, a number of network elements which may or may not reshape the flow forward the stream, and the destination determines (in the RSVP model) the reservation to be made. As in earlier sections, though RSVP is not essential for Guaranteed Service, we use it as a convenient mechanism for showing the parameters necessary to provide guaranteed service in an IP network.

2.3.1 Models

Since guaranteed service is quantitative - that is, it delivers specific delay bounds - it has to be based on both a model of source behaviour and a model of how the network elements handle the flow. Note that these models are for the purpose of bounding queueing delay. The other component of end-to-end delay is propagation delay which is controlled by the routing algorithms in use when the flow is set up. The models are shown in Figure 7.

The model on the left is a *token bucket* [14] which is an envelope for the source. That is, the source stream must at all times stay within the bounds set up by the token bucket. The model on the right is the model of a link in a router. It shows the resources (buffers and bandwidth) that a router must allocate for guaranteed flows.

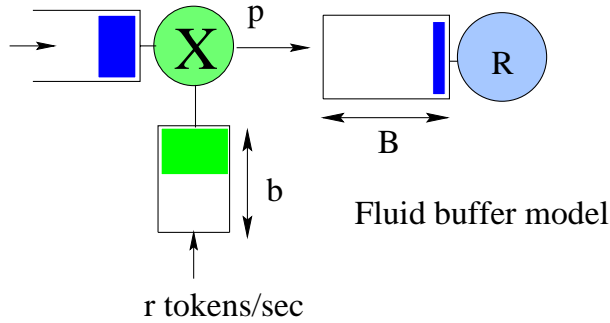


Figure 7: Guaranteed Service models

The model used is a “fluid model”. The source releases fluid at a rate r (which is the rate that tokens are produced) but it is possible for the source to produce (temporarily) fluid faster than rate r . This excess is stored at the source and released at rate r . If the source has not transmitted anything for some time, tokens keep accumulating up to a limit, b . Any fluid that arrives at the source can be released instantaneously, as long as there are tokens available. The rate at which fluid can be transmitted is p .

This is an approximation of how packets are produced and transmitted, where the actual transmission rate of a given packet may be much higher than the rate at which packets (or bytes) are produced.

The fluid flow arrives at the network element whose resources are modelled by a server which serves the fluid at rate R and a buffer of size B which the flow may use. The buffer is necessary because the transmission rate p may be much greater than R .

The theoretical result that the fluid delay at the network is bounded by $\frac{b}{R}$ (given that $r \leq R$) gives the basis for guaranteed service. One can easily see that this bound holds, by assuming that $p = \infty$ (that is, the fluid is “instantaneously” transferred from the source to the buffer), and by assuming that the token pool is full (that is, there are b tokens available). If the source were to produce an amount of fluid greater than or equal to b , then the buffer at the network element would be filled with b units of fluid and the largest delay experienced would be the $\frac{b}{R}$, determined by the rate at which the network element serves the fluid.

Real network elements do not implement the fluid model perfectly, and so the additional delays due to the implementation are captured in two additional terms: C is the term that is intended to capture delay that is dependent on the flow transmission rate. D is the term that captures non-rate-dependent delay. So the adjusted delay is:

$$\text{Delay} = \frac{b}{R} + \frac{C}{R} + D \quad (1)$$

The delay bounds built up by guaranteed service are computed along the path of the flow by Equation 1. We describe in the next section how this is done.

2.3.2 Parameters and invocation

To outline the general flow of information: The sender sends the TSpec and AD-SPEC in a PATH message in the direction of the receiver of the flow. The TSpec

is not changed as it flows through the network elements, but the parameters in the ADSPEC are composed, in this case, accumulated, and used by the intermediate network elements in the path of the flow. The receiver uses the information provided by the sender TSpec and the ADSPEC and computes a TSpec and RSpec which comprises part of the FLOWSPEC which is sent upstream in the RESERVE message.

From the models outlined in the previous section, we can derive a number of parameters that are necessary to specify in order to invoke guaranteed service. The first set of parameters are part of the sender's TSpec:

(r, b) These are the parameters of the token bucket. r denotes the rate at which bytes will be produced and b is the depth of the token pool (in bytes).

p This is the peak rate at which packets will be sent.

m This is the minimum policed unit. Any IP packet that is less than this size is counted by the token bucket at m bytes for purposes of determining whether the flow is conforming or not.

M This is the maximum datagram size that will be sent by the flow. (Note that this may be negotiated downwards if intermediate (sub)networks have a smaller MTU size.)

Note that the TSpec itself does not change as it is sent through the network.

The RSpec sent by the receiver contains the following:

R The rate in bytes per second. (Note this is the server rate of the fluid model above).

S A slack term that may be used by intermediate nodes to compute a lower level of resource reservation. This is the difference between the delay guarantee needed by the application and the delay guarantee that can be guaranteed by reserving bandwidth R . If this difference is positive (i.e. there is slack) then intermediate nodes may use some or all of it.

The parameters that are built up along the path, as the flow is set up include the following list. These parameters are carried in the ADSPEC and are processed by each RSVP/Intserv/Guaranteed Service capable router along the path. These parameters include:

C_{tot} The total rate-dependent delay computed end-to-end on the path from the sender to the receiver. This parameter is computed locally by each network element in the path and then added to the total. The units are bytes.

C_{sum} The sum of the rate-dependent delay since the last reshaping point in the network. The units are bytes.

D_{tot} The total rate-independent delay computed end-to-end on the path from sender to receiver. This parameter is computed locally by each network element in the path and then added to the total. The units are microseconds.

D_{sum} The sum of the rate-independent delay since the last reshaping point in the network. The units are microseconds.

Use of the parameters: C_{tot} and D_{tot} are used by the endpoints to compute the end-to-end delay bounds achievable on the path according to Equation (1). By using Equation (1), substituting $R = r$ and the target delay for the flow, the receiver can compute the slack term of the RSpec, S , by taking $\max\{\text{Desired queueing delay} - \text{Total queueing delay}, 0\}$. Intermediate network elements can then use this slack to reduce their reservations so that while the targets are still met, the local delay can be increased and the rate reservation decreased¹. This is done by using (a portion of) the slack contained in the RSpec. The slack term is then reduced by the network element and forwarded on.

In addition, C_{sum} and D_{sum} are used by the network elements in order to compute buffer requirements needed to achieve (near) 0-loss at intermediate nodes. These parameters are, in effect, used for network element-to-network element communication. The reason that this computation is needed is that traffic can be distorted from the original token bucket envelope simply by flowing through queueing points in the network. Although some reshaping points are dictated by the RFC², reshaping points may be placed anywhere in the path of the flow (and would most likely occur at domain boundaries). The presence of reshaping points can be used to reduce the buffer resource requirements at intermediate network elements. Specifically, the buffer size needed to handle bursts is given by:

$$\begin{cases} M + (b - M)\left(\frac{C_{sum}}{R} + D_{sum}\right)r & \text{if } \frac{b-M}{p-r} < \frac{C_{sum}}{R} + D_{sum} \\ M + (b - M)\frac{p-R}{p-r} + C_{sum} + D_{sum}R & \text{if } \frac{b-M}{p-r} \geq \frac{C_{sum}}{R} + D_{sum} \text{ and } p > R \\ M + \left(\frac{C_{sum}}{R} + D_{sum}\right)p & \text{otherwise} \end{cases} \quad (2)$$

3 Differentiated Services

The **diffserv** working group in the IETF is developing a new view of how to provide services in the Internet. *Differentiated Services* is the term used to describe this work, but it includes (in some sense) the work on *Integrated Services* as well as the work on RSVP and the work on Policy Control. While Integrated Services (and to some extent, RSVP) is designed to work with individual traffic streams (flows, now sometimes called *microflows*), Differentiated Services is designed to work with traffic aggregates (also, confusingly, sometimes referred to as flows, or even *macroflows*).

What is emerging from this working group is a new network model, in effect. The origin of the workgroup was the perceived scalability problem of Integrated Services and RSVP, but the result has been a more encompassing view of service provisioning in the Internet.

In Figure 8 we see an example of this new network model. **Integrated services**, such as guaranteed service and controlled load service, are provided by routers in the outer regions of the network where, it is assumed, they will be dealing with rather fewer flows and so the scalability problems do not arise. In the core of the network, routers will see large aggregations of traffic, perhaps thousands or tens of thousands of flows. These routers do not keep track of flows individually, but rather, provide different services to different classes of flows, and so treat only aggregates.

Figure 8 is of course just an example. Hosts may be connected directly to *diffserv* (DS) subnets. Further, DS subnets may be customers (i.e. traffic sources) for one another.

The routers in the interior of the core can be somewhat simpler than the routers at the border of the *diffserv* (DS) cloud. The routers at the border of a *diffserv* cloud are responsible for conditioning traffic according to a *traffic conditioning agreement*

¹Note that this is at the cost of additional buffer space.

²These points are heterogeneous branch points for multicast flows and source merge points where flows from two different sources share a reservation.

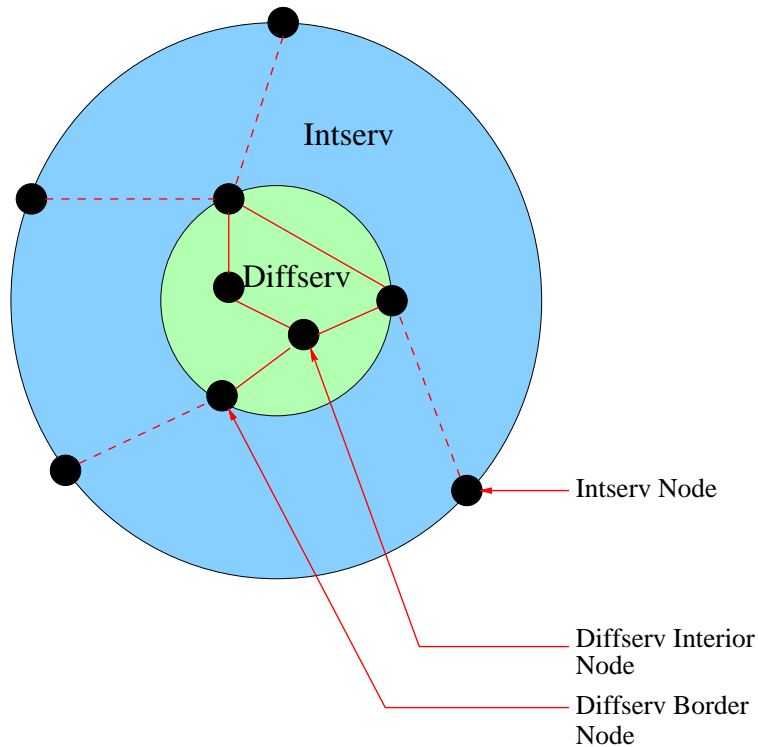


Figure 8: A Possible Network View of Differentiated Services

or *TCA* which may be in effect between administrations owning connecting peer diffserv subnets. The *TCA* itself may be part of a *Service Level Agreement* or *SLA* which defines exactly what service the diffserv subnet will offer and how much capacity (for example using an arrangement such as the link sharing scheme of Floyd and Jacobson [7]) is allowed to a traffic source (which could itself be an aggregate traffic source). Note, however, that some of this DS border functionality can be implemented in hosts that are attached to (and request DS services from) a DS subnet.

The border routers of the diffserv subnet since they implement SLAs in a sense, must also implement policy. This can include link sharing as well as the mapping of services from another domain to the service set offered by this domain and the degradation or promotion of packets to (respectively) lower or higher service classes.

3.1 Architecture

The idea of differentiated services is simple: A unidirectional flow enters the network through an ingress border router where it is conditioned and forwarded using a *per-hop behaviour (PHB)* which provides the service intended for the flow. At each subsequent router, the PHB is applied to the flow until it reaches an egress border router which (possibly) shapes and conditions the flow before sending it on to another DS or non-DS domain or host.

The architecture of a DS node will differ somewhat based on the position of the node in the network (border or interior) and based on the choices made by the administration of the network for implementing policy and service level agreements. The architecture of differentiated services is described in detail in two RFCs [2, 3].

The boundary node in general has more functions than interior nodes and may

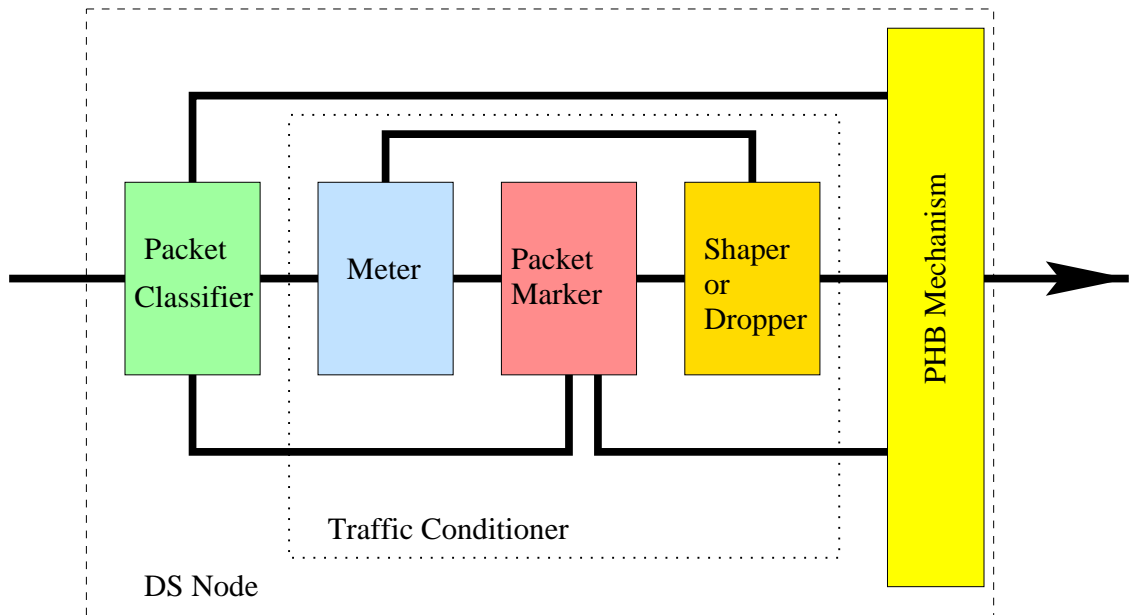


Figure 9: Nodal Architecture of Differentiated Services

extend services such as initial packet marking and shaping to hosts or domains that are incapable of providing those functions themselves. Interior nodes can assume that traffic has already passed through a DS border node and has therefore already been marked properly and conditioned. These functions may be omitted from an interior DS node (though they may, of course, be present).

The SLA is a key concept in DS. The SLA determines how a downstream DS domain will handle an upstream domain's packets. The SLA typically contains a *Traffic Conditioning Agreement (TCA)* which describes the classification of flows, how packets belonging to those flows are marked or remarked, possibly token bucket parameters, and agreements on how to handle packets which are out-of-profile (i.e. which do not meet the agreed-upon characteristics as measured by some metering device). There are a number of logical 'devices' which are used to provide this traffic conditioning function. They are illustrated in Figure 9.

A traffic profile specifies some of the properties of the flow which is to receive a certain level of service. It is some specification or device, such as a token bucket, which allows the router to determine unambiguously whether a particular packet is to be regarded as in- or out-of-profile. In addition, on the basis of the SLA/TCA or on the basis of domain policy, out-of-profile packets may be treated differently than in-profile packets. For example, they may receive different PHBs or they may be conditioned differently or they may incur different charges.

The first device is a packet classifier. This is a device which is used essentially to select which flows are to receive a particular service. There are two kinds of packet classifiers: one type is based on using **only** the DS field in the IP packet header. This type is called a *behaviour aggregate (BA)* classifier. The advantage of this type of classification is that it is simple (1 byte field) and there are only a limited number of states (for example, token buckets) which have to be maintained by the router or interface. The other type of classifier is called a *multi-field (MF)* classifier and it may use multiple fields in the packet header to identify the packet as part of a flow. One advantage of this type of classification is that (e.g.) packets belonging to flows arriving on the same interface, but which are covered by separate SLAs

may be identified and distinguished. A disadvantage is that state information is potentially very large, if this kind of classification is used to excess.

Figure 9 shows a number of logical devices grouped together as a *Traffic Conditioner*. Traffic conditioners are most useful as logical devices residing in ingress or egress DS border nodes. However, they can exist as well at traffic sources and at interior DS nodes. In a particular node or on a particular interface in a DS domain, some (or even all) of the functions shown in the traffic conditioner may be null.

To take the logical devices in turn, the *meter* monitors the substream identified by the packet classifier. It consists of the logical equivalent of the token bucket and should be able to identify packets as in-profile or out-of-profile. Depending on the SLA and domain policy, the information from the traffic meter can be used by the marking function or the shaper/dropper or both.

The *packet marker* essentially causes the packet to be treated according to the SLA/TCA. In effect, the relationship between services in different DS domains, as well as the relationship between services in a non-DS domain (such as an IntServ domain) and the services provided in the DS domain is contained in the marking function of the packet marker. The packet marker sets the value of the DS byte in the IP packet header, possibly based on the results of the packet classifier function and the metering function. The DS byte value selects the PHB to be received by all the packets bearing that marking (also known as a *behaviour aggregate*). So, issues spelled out in the SLA, such as which flows are to receive what services, and what is to be done with out-of-profile packets, are implemented by the boundary router marking the DS field with a particular value. This DS byte value determines the treatment that the packet receives in the domain.

A *shaper* ensures that a flow conforms exactly with the parameters given by a particular traffic profile. This may cause some packets to be delayed. If the actual packet stream is very different from the traffic profile, this could result in dropped packets just because the shaper is not required to have an infinite (or even an extremely large) buffer.

A *dropper* discards packets from a flow in order to force conformance with a traffic profile. Whether a dropper is in operation for all or part of a flow (say, for example, for out-of-profile packets) or even at all, is determined by the SLA/TCA or by domain policy. The dropper may use information from the classifier, meter and marker in making its decision about whether to drop a packet.

3.2 Service Definitions

There have been several Internet Drafts which describe service definitions for DS subnets. We will first describe the basic PHBs that are designed to be compatible with the use of the IP Precedence Field (formerly part of the TOS field) and then we will describe the *Expedited Forwarding* PHB proposed in [9] and the most recently defined *Assured Forwarding* PHB [8] as examples of “value add” PHBs.

3.2.1 The DS field and PHBs

The DS Byte is a redefinition of the TOS byte in the IPv4 header and the Traffic Class Octet in the IPv6 header. This byte is divided into two fields: a 6-bit DS codepoint and a 2-bit currently unused field. The relationship between the 6-bit codepoint and PHBs implemented in a node and domain, is configurable. In order to maintain some backward compatibility with current IP Precedence Field usage, a set of codepoints are defined, called *Class Selector codepoints* which are mapped to a standard set of minimum behaviours which are compatible with already deployed IP Precedence behaviour. Note that the IP Precedence field is 3 bits.

The basic PHB which is supported by DS nodes is called the *default* PHB and it is the normal best-effort forwarding behaviour of IP ([1]). That is to say, subject to resource constraints dictated by the network administration, packets are forwarded at every opportunity which is not required to serve another class of traffic.

The codepoint which selects the default PHB is B'000000'. Other codepoints may, of course, be mapped to this PHB also, and an unrecognized codepoint may select this PHB as a default way of handling it. This PHB is compatible with the way packets with that TOS byte value are treated today.

There are 8 *class selector codepoints*, of which one is the code point selecting the default behaviour. The PHBs selected by the other seven codepoints must meet the following requirements: There is an order imposed on class selector codepoints by their numerical values. The general rule for class selector PHBs is that, under equivalent loads³, the PHBs associated with a higher-valued class selector codepoint should yield **better** behaviour than PHBs associated with lower-valued class selector codepoints. To meet the minimum requirements, there must be at least two distinct forwarding behaviours discernable. One of these behaviours must give preference to packets bearing a DS codepoint of B'11x000' over packets receiving the default PHB, for compatibility with IP Precedence classes B'111' and B'110' which are used for routing traffic. The class selector PHBs provide a basis for a set of standard PHBs which can be provided by DS nodes.

The 64 codepoints of the space are divided into 3 parts:

B'xxxxx0' Used for standard PHBs. This consists of 32 codepoints which may be assigned to standard PHBs.

B'xxxx11' Used for Experimental or Local Use PHBs. These are defined and used only within a single DS domain.

B'xxxx01' Used for Experimental or Local Use PHBs. However, these 16 codepoints may be taken in the future for use in the standards pool if the 32 codepoints there should prove insufficient.

3.2.2 Expedited Forwarding PHB

There are Internet Drafts proposing a number of PHBs, some in a more or less preliminary state. One of the more complete, and which follows the guidelines in [3] is the description of the *Expedited Forwarding (EF)* PHB [9]. The purpose of the EF PHB is to build a 'virtual leased line' type of service (the same type of service attempted by the CBR transfer mode in ATM). The characteristics of this service are: Small packet loss ratio, low delay and delay variation (jitter), and what is termed 'assured bandwidth' but which means a guaranteed (more or less) throughput. The definition of the EF PHB focusses on the mechanisms needed in the forwarding path. It assumed, for correct operation, that traffic is already conditioned at the border nodes of the DS domain and, of course, it implicitly assumes that admission control functions are in place in the domain.

The key idea of the DS PHB is that for the EF aggregate flow, the departure rate is held to be at a given (configurable) rate or higher, regardless of the load on the node at any given time, and also regardless of the characteristics of any other traffic stream seen by the node. The reason for these (stringent) requirements is that in order to provide the service with the characteristics listed above, the queues must be kept consistently low. The obvious way to achieve this is by holding the

³Actually, this is a very problematic point. The meaning of *equivalent* here is left (intentionally, I think) very vague. It is problematic because it is not clear how you would define equivalent loads, because the influence of flows on one another and the second-order properties of the flows (such as variability) could make a great deal of difference, even when the average load applied is the same.

input rate (possibly significantly) lower than the output rate. In order to do that, the output rate must be fixed, or at least must be constrained not to fall below a given level.

There are several mechanisms that could be used to implement the EF PHB, including simple priority queueing, weighted round robin scheduling (WRR) and others. Although these mechanisms can implement the basic function of EF, they do not all have the same characteristics. Some properties of the service, such as jitter, may be significantly different for different implementations of the same service. Also how the concatenation of different implementations of the EF PHB will affect these properties is not known at this time.

3.2.3 Assured Forwarding PHB Group

Assured Forwarding (AF)[8] is an example of the definition of a PHB group, that is a related set of per-hop behaviours. The purpose of this service definition is to allow a domain to provide different levels of delivery assurance, and possibly different levels of delay and jitter. However, these last two qualities are not the primary focus of AF. There are no explicit goals set for AF with respect to jitter and delay, though under certain circumstances (i.e. dependent on the resource allocation made and the particular mechanisms used to implement this PHB group) the macroflow assigned a higher AF PHB may indeed see better delay and jitter characteristics.

The codepoints to be assigned to the AF PHB group are given in Table 1.

	AF 1	AF 2	AF 3	AF 4
Low drop preference	b'010000'	b'011000'	b'100000'	b'101000'
Medium drop preference	b'010010'	b'011010'	b'100010'	b'101010'
High drop preference	b'010100'	b'011100'	b'100100'	b'101100'

Table 1: DS byte codepoints for AF

The requirements of the PHB group are stated in terms of forwarding behaviour, resource allocation and reordering. There are 4 AF classes and 3 drop preference levels within each class. The six DS byte bits are divided into two 'fields': The first field (3 bits) is the AF class number (decimal 2,3,4 or 5) and the second field (3 bits) is the drop precedence (decimal 0, 2 or 4). The numerical ordering of the DS byte (AF class number) implies an ordering of the service delivered. If $codepoint(x) < codepoint(y)$ then class(y) has at least as many forwarding resources (buffers and bandwidth) allocated to it as class(x). Likewise, the ordering of the of the drop precedence fields implies an ordering of drop ratios: if $codepoint(p) < codepoint(q)$ then the drop ratio of packets within an AF class with drop precedence p must be at least as small as the drop ratio of packets marked with drop precedence q . Finally, within an AF class, the node is not allowed to reorder the packets of a microflow, no matter what the drop precedence.

The intention is that nodes implement all four AF classes and that these classes are not aggregated; that is, packets belonging to different classes must be forwarded independently of one another. For the three levels of drop precedence, there must be at least two distinguishable levels of drop ratio. If the network (or domain) provides only two levels of drop ratio, then the two higher drop precedence values are mapped into the higher drop ratio and the lowest drop precedence value is mapped into the lower drop ratio.

Finally, traffic conditioning can be used to limit the traffic volume of AF traffic entering a domain. Packets can be dropped by traffic conditioning actions, but they can also be upgraded or downgraded in drop precedence, or even assigned to another

AF class (it is not stated whether packets can be downgraded to best effort). If packets are assigned to another AF class, because of ordering considerations, packets from the same microflow are not allowed to be assigned to different AF classes.

4 Conclusions

It is clear from the activity in the IETF that there is a new picture of the IP network architecture emerging as a result of the work in the Intserv and Diffserv groups. These two approaches to providing differentiated services are not at all incompatible, and, to a certain extent, can be made to work together. However, there are quite a number of difficult and serious questions that have to be answered, including the following:

- How do concatenated diffserv networks behave when the PHBs defined in each network have different implementations ?
- Can diffserv networks effectively provide transparency to intserv flows given the PHBs defined so far ?
- What mechanisms are suitable for implementing PHBs and what are the (quantitative) characteristics of different mechanisms used for implementing the same PHBs. Do these characteristics make a significant difference in how the service is perceived ?

References

- [1] Fred Baker. Requirements for IPv4 routers. RFC 1812, June 1995.
- [2] Yoram Bernet, James Binder, Steven Blake, Mark Carlson, Elwyn Davies, Borje Ohlman, Dinesh Verma, Zheng Wang, and Walter Weiss. A framework for differentiated services. Internet Draft `draft-ietf-diffserv-framework-00.txt`, May 1998. Expires November 1998.
- [3] Steven Blake, David Black, Mark Carlson, Elwyn Davies, Zheng Wang, and Walter Weiss. An architecture for differentiated services. Internet Draft `draft-ietf-diffserv-arch-01.txt`, August 1998. Expires February 1999.
- [4] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource reservation protocol (RSVP)- version 1 functional specification. Internet Draft, November 1996. Expires: 4/29/97.
- [5] L. Breslau and S. Shenker. Partial service deployment in the integrated services architecture. Internet Draft, 23. April 1997. Expires: 10/23/97.
- [6] Alan Demers, Srinivasan Keshav, and Scott Shenker. Analysis and simulation of a fair queueing algorithm. *Internetworking: Research and Experience*, 1:3–26, 1990.
- [7] Sally Floyd and Van Jacobson. Link-sharing and resource management models for packet networks. *IEEE/ACM Transactions on Networking*, 3(4):365–386, August 1995.
- [8] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski. Assured forwarding phb group. Internet Draft `draft-ietf-diffserv-af-01.txt`, October 1998. Expires April 1999.

- [9] Van Jacobson, Kathleen Nichols, and Kedarnath Poduri. An expedited forwarding PHB. Internet Draft `{draft-ietf-diffserv-phb-ef-00.txt}`, August 1998. Expires February 1999.
- [10] K. Nichols, V. Jacobson, and L. Zhang. A two-bit differentiated services architecture for the Internet. Internet Draft, 1997.
- [11] Kathleen Nichols and Steven Blake. Differentiated services: Operational model and definitions. Internet Draft, February 1998. Expired August 1998.
- [12] Kathleen Nichols, Steven Blake, Fred Baker, and David Black. Definition of the differentiated services (DS Field) in the IPv4 and IPv6 headers. Internet Draft `{draft-ietf-diffserv-header-02.txt}`, August 1998. Expires February 1999.
- [13] A.K. Parekh and R.G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: The single node case. *IEEE/ACM Transactions on Networking*, 1(3):344–357, June 1993.
- [14] Craig Partridge. *Gigabit Networking*. Professional Computing Series. Addison-Wesley, Reading, Massachusetts, 1994.
- [15] S. Shenker, C. Partridge, and R. Guerin. Specification of Guaranteed Quality of Service. Request for Comments 2212, September 1997.
- [16] J. Wroclawski. Specification of the Controlled-Load Network Element Service. Request for Comments 2211, September 1997.
- [17] J. Wroclawski. The use of RSVP with IETF integrated services. Request for Comments 2210, September 1997.