Stardust.com, Inc.
1901 S. Bascom Ave, Suite 333
Campbell, California 95008
Phone: 408-879-8080
Fax: 408-879-8081

www.stardust.com

www.qosforum.com

White Paper -

QoS protocols & architectures

Quality of Service protocols use a variety of complementary mechanisms to enable deterministic end-to-end data delivery

---

QoS protocols & architectures

Quality of Service protocols use a variety of complementary mechanisms to enable deterministic end-to-end data delivery

QoS protocols & architectures

Quality of Service protocols use a variety of complementary mechanisms to enable deterministic end-to-end data delivery

Scope of this document

This purpose of this paper is to provide an introduction to and overview of the Quality of Service (QoS) protocols now available or under development for Internet Protocol (IP) based networks. After a brief introduction to the topic, we provide a high-level description of how each QoS protocol operates. We consider the many architectures in which the protocols work together along with policy management to provide end-to-end QoS for IP application traffic, and end by briefly describing the state of QoS support for IP multicast and explicit policy support.

Introduction

Standard Internet Protocol (IP)-based networks provide "best effort" data delivery by default. Best-effort IP allows the complexity to stay in the end-hosts, so the network can remain relatively simple [e2e]. This scales well, as evidenced by the ability of the Internet to support its phenomenal growth. As more hosts are connected, network service demands eventually exceed capacity, but service is not denied. Instead it degrades gracefully. Although the resulting variability in delivery delays (jitter) and packet loss do not adversely affect typical Internet applications--email, file transfer and Web applications—other applications cannot adapt to inconsistent service levels. Delivery delays cause problems for applications with real-time requirements, such as those that deliver multimedia, the most demanding of which are two-way applications like telephony.

Increasing bandwidth is a necessary first step for accommodating these real-time applications, but it is still not enough to avoid jitter during traffic bursts. Even on a relatively unloaded IP network, delivery delays can vary enough to continue to adversely affect real-time applications. To provide adequate service -- some level of quantitative or qualitative determinism -- IP services must be supplemented. This requires adding some "smarts" to the net to distinguish traffic with strict timing requirements from those that can tolerate delay, jitter and loss. That is what Quality of Service (QoS) protocols are designed to do. QoS does not create bandwidth, but manages it so it is used more effectively to meet the wide range or application requirements. The goal of QoS is to provide some level of predictability and control beyond the current IP "best-effort" service.

A number of QoS protocols have evolved to satisfy the variety of application needs. We describe these protocols individually, then describe how they fit together in various architectures with the end-to-end principle in mind. The challenge of these IP QoS technologies is to provide differentiated delivery services for individual flows or aggregates without breaking the Net in the process. Adding "smarts" to the Net and improving on "best effort" service represents a fundamental change to the design that made the Internet such a success. The prospect of such a potentially drastic change makes many of the Internet's architects very nervous.

To avoid these potential problems as QoS protocols are applied to the Net, the end-to-end principle is still the primary focus of QoS architects. As a result, the fundamental principle of *"Leave complexity at the 'edges' and keep the network 'core' simple"* is a central theme among QoS architecture designs. This is not as much a focus for individual QoS protocols, but in how they are used together to enable end-to-end QoS. We explore these architectures later in this paper after we give a brief overview of each of the key QoS protocols.

The QoS protocols

There is more than one way to characterize Quality of Service (QoS). Generally speaking, QoS is the ability of a network element (e.g. an application, a host or a router) to provide some level of assurance for consistent network data delivery. Some applications are more stringent about their QoS requirements than others, and for this reason (among others) we have two basic types of QoS available:

- ***Resource reservation*** (integrated services): network resources are apportioned according to an application's QoS request, and subject to bandwidth management policy.

- ***Prioritization*** (differentiated services): network traffic is classified and apportioned network resources according to bandwidth management policy criteria. To enable QoS, network elements give preferential treatment to classifications identified as having more demanding requirements.

These types of QoS can be applied to individual application "flows" or to flow aggregates, hence there are two other ways to characterize types of QoS:

- ***Per Flow***: A "flow" is defined as an individual, uni-directional, data stream between two applications (sender and receiver), uniquely identified by a 5-tuple (transport protocol, source address, source port number, destination address, and destination port number).

- ***Per Aggregate***: An aggregate is simply two or more flows. Typically the flows will have something in common (e.g. any one or more of the 5-tuple parameters, a label or a priority number, or perhaps some authentication information).

Applications, network topology and policy dictate which type of QoS is most appropriate for individual flows or aggregates. To accommodate the need for these different types of QoS, there are a number of different QoS protocols and algorithms:

- ***ReSerVation Protocol (RSVP)***: Provides the signaling to enable network resource reservation (otherwise known as Integrated Services). Although typically used on a per-flow basis, RSVP is also used to reserve resources for aggregates (as we describe in our examination of QoS architectures).

- ***Differentiated Services (DiffServ)***: Provides a coarse and simple way to categorize and prioritize network traffic (flow) aggregates.

- ***Multi Protocol Labeling Switching (MPLS)***: Provides bandwidth management for aggregates via network routing control according to labels in (encapsulating) packet headers.

- ***Subnet Bandwidth Management (SBM)***: Enables categorization and prioritization at Layer 2 (the data-link layer in the OSI model) on shared and switched IEEE 802 networks.

QoS	***Net***	***App***	***Description***
most	X		Provisioned resources end-to-end (e.g. private, low-traffic network)
	X	X	RSVP (Resource reSerVation Protocol) [IntServ Guaranteed] Service (provides feedback to application)
	X	X	RSVP [IntServ Controlled] Load Service (provides feedback to application)
	X		Multi-Protocol Label Switching [MPLS]
	X	X	Differentiated Services [DiffServ] applied at network core ingress appropriate to RSVP reservation service level for that flow. Prioritization using Subnet Bandwidth Manager [SBM] applied on the LAN would also fit this category.
	X	X	Diffserv or SBM applied on per-flow basis by source application
	X		Diffserv applied at network core ingress
	X		Fair queuing applied by network elements (e.g. CFQ, WFQ, RED)
least			Best effort service

Table 1. Shows the different bandwidth management algorithms and protocols, their relative QoS levels, and whether they are activated by network elements (Net) or applications (App), or both.

Table 1 compares the QoS protocols in terms of the level of QoS they provide and where the service and control are implemented -- in the Application (App) or in the

Network (Net). Notice that this table also refers to router queue management algorithms such as Fair Queuing (FQ), Random Early Drops (RED). Queue management—including the number of queues and their depth, as well as the algorithms used to manage them--is very important to QoS implementations. We refer to them here only to illustrate a full spectrum of QoS capabilities, but as they are largely transparent to applications and not explicitly QoS algorithms, we will not refer to them again. For more information see [Queuing].

The QoS protocols we are focused on in this paper vary, but they are not mutually exclusive of one another. On the contrary, they complement each other nicely. There is a variety of architectures in which these protocols work together to provide end-to-end QoS across multiple service providers. We will now describe each of these protocols in some more detail -- describing their essential mechanics and functionality -- and follow that with a description of the various architectures in which they can be used together to provide end-to-end QoS.

RSVP – Resource reservation

The ReSerVation Protocol (RSVP) is a signaling protocol that provides reservation setup and control to enable the integrated services [IntServ], which is intended to provide the closest thing to circuit emulation on IP networks. RSVP is the most complex of all the QoS technologies, for applications (hosts) and for network elements (routers and switches). As a result, it also represents the biggest departure from standard "best-effort" IP service and provides the highest level of QoS in terms of service guarantees, granularity of resource allocation and detail of feedback to QoS-enabled applications and users.

Here is a simplified overview of how the protocol works, as illustrated in Figure 1:

- Senders characterize outgoing traffic in terms of the upper and lower bounds of bandwidth, delay, and jitter. RSVP sends a **PATH message** from the sender that contains this *traffic specification* (TSpec) information to the (unicast or multicast receiver(s)) destination address. Each RSVP-enabled router along the downstream route establishes a "path-state" that includes the previous source address of the PATH message (i.e. the next hop "upstream" towards the sender).

- To make a resource reservation, receivers send a RESV (reservation request) message "upstream". In addition to the TSpec, the **RESV message** includes a *request specification* (Rspec) that indicates the type of Integrated Services required—either Controlled Load or Guaranteed--and a *filter specification* (filter spec) that characterizes the packets for which the reservation is being made (e.g. the transport protocol and port number). Together, the RSpec and filter spec represent a *flow-descriptor* that routers use to identify each reservation (a.k.a., a "flow" or a "session").

- When each RSVP router along the upstream path receives the RESV message, it uses the admission control process to authenticate the request and allocate the necessary resources. If the request cannot be satisfied (due to lack of resources or authorization failure), the router returns an error back to the receiver. If accepted, the router sends the RESV upstream to the next router.

- When the last router receives the RESV and accepts the request, it sends a confirmation message back to the receiver (note: the "last router" is either closest to the sender or at a reservation merge point for multicast flows).

- There is an explicit tear-down process for a reservation when sender or receiver ends an RSVP session.

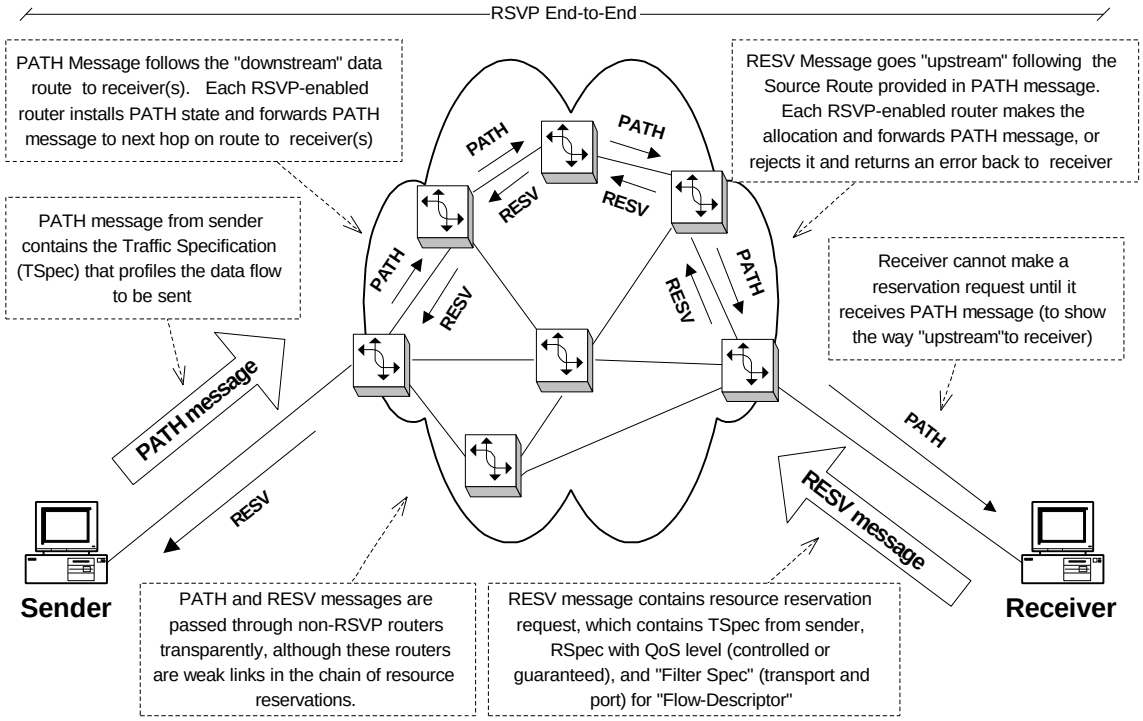

Figure 1: RSVP "PATH" and "RESV" messages are used to establish a resource reservation between a sender and receiver. There is an explicit tear-down of reservations also (not shown).

RSVP enables **Integrated Services**, of which there are two fundamentally different types:

- **Guaranteed**: This comes as close as possible to emulating a dedicated virtual circuit. It provides firm (mathematically provable) bounds on end-to-end queuing delays by combining the parameters from the various network elements in a path, in addition to ensuring bandwidth availability according to the TSpec parameters [IntServ Guaranteed].

- **Controlled Load**: This is equivalent to "best effort service under unloaded conditions." Hence, it is "better than best-effort," but cannot provide the strictly bounded service that Guaranteed service promises [IntServ Controlled].

Integrated Services use a token-bucket model to characterize its input/output queuing algorithm. A token-bucket is designed to smooth the flow of outgoing traffic, but unlike a leaky-bucket model (which also smoothes the out-flow), the token-bucket model allows for data bursts--higher send rates that last for short periods [Partridge].

Data flows for an RSVP session are characterized by senders in the TSpec (traffic specification) contained in PATH messages, and mirrored in the RSpec (reservation specification) sent by receivers in RESV messages. The token-bucket parameters—bucket rate, bucket depth, and peak rate--are part of the TSpec and RSpec. Here is a complete list of the parameter descriptions [RSVP IntServ, IntServ Parameters, IntServ Controlled]. For both Guaranteed and Controlled Load service, non-conforming (out-of-spec) traffic is treated like non-QoS best-effort traffic:

- **Token rate (r)**: The continually sustainable bandwidth (bytes/second) requirements for a flow. This reflects the *average* data rate *into* the bucket, and the target *shaped* data rate *out* of the bucket.

- **Token-bucket depth (b)**: The extent to which the data rate can exceed the sustainable average for short periods of time. More precisely, the amount of data sent cannot exceed rT+b (where T is any time period).

- **Peak Rate (p)**: This is set to the maximum send rate (bytes/second) if known and controlled, or to positive infinity if not known (floating point value represented by 255.000…0, as described in RFC 1832). For all time periods (T), the amount of data sent cannot exceed M+pT.

- **Minimum policed size (m)**: The (byte) size of the smallest packet (data payload only) generated by the sending application. This is not an absolute number, but in cases where the percentage of small packets is small, this number should be increased to reduce the overhead estimate for this flow (which can affect reservation acceptance). All packets smaller than m are treated as size m and policed accordingly.

- **Maximum packet size (M)**: The biggest packet (in bytes). This number should be considered an absolute, since any packets of larger size are considered out of spec and may not receive QoS-controlled service as a result.

In our description of the traffic and reservation specifications, we have omitted details about other RSVP and Integrated Service features such as:

1) ADSpec in a PATH message, which contains information (service, delay, bandwidth estimates, etc.) generated by the data source or any or all network nodes in the downstream path.

2) Reservation styles, which deal with how one reservation interacts with others.

3) Filter-spec which allow characterization of "sub-flows" that could be used in a hierarchically encoded signal for heterogeneous receivers, for example.

4) Policy data, which provides detailed condition information for use in resource reservation policy decisions.

Here is a summary of the more salient characteristics of the RSVP Protocol mechanisms:

- Reservations in each router are "soft," which means they need to be refreshed periodically by the receiver(s).

- RSVP is not a transport, but a network (control) protocol. As such, it does not carry data, but works in parallel with TCP or UDP data "flows."

- Applications require APIs to specify the flow requirements, initiate the reservation request, and receive notification of reservation success or failure after the initial request and throughout a session. To be useful, these APIs also need to include RSVP error information to describe a failure during reservation setup or anytime thereafter during the lifetime of a reservation as conditions change.

- Reservations are receiver-based, in order to efficiently accommodate large heterogeneous (multicast) receiver groups.

- Multicast reservations are "merged" at traffic replication points on their way upstream, which involves complex algorithms that are not well understood yet [RSVP Killers]. We discuss the topic of QoS support for multicast in more detail later in this paper.

- Although RSVP traffic can traverse non-RSVP routers, this creates a "weak-link" in the QoS chain where the service falls-back to "best effort" (i.e. there is no resource allocation across these links).

- There are two types of RSVP Protocols: *Native RSVP* has an IP Protocol number 46 (for the protocol field of an IP header), and the RSVP header and payload are encapsulated by the (raw) IP header itself. *UDP-encapsulated RSVP* has its header contained in a UDP

datagram. The 802 "Subnet Bandwidth Manager" that we describe later in this paper only supports *Native RSVP*.

As mentioned already, RSVP provides the highest level of IP QoS available. It allows an application to request QoS with a high level of granularity and with the best guarantees of service delivery possible. This sounds wonderful and leaves one wondering why we need anything else. The reason is that it comes at the price of complexity and overhead, thus is overkill for many applications and (as we describe later) for some portions of the network. Simpler, less fine-tuned methods are needed, and that is what DiffServ provides, as we describe now.

DiffServ – Prioritization

Differentiated Services [DiffServ] provides a simple and coarse method of classifying services of various applications. Although others are possible, there are currently two standard per hop behaviors (PHBs) defined that effectively represent two service levels (traffic classes):

- ***Expedited Forwarding*** (EF): Has a single *codepoint* (DiffServ value). EF minimizes delay and jitter and provides the highest level of aggregate quality of service. Any traffic that exceeds the traffic profile (which is defined by local policy) is discarded [DiffServ EF].

- ***Assured Forwarding*** (AF): Has four classes and three drop-precedences within each class (so a total of twelve *codepoints*). Excess AF traffic is not delivered with as high probability as the traffic "within profile," which means it may be demoted but not necessarily dropped [DiffServ AF].

As illustrated in Figure 2, PHBs are applied by the conditioner to traffic at a network ingress point (network border entry) according to pre-determined policy criteria. The traffic may be marked at this point, and routed according to the marking, then unmarked at the network egress (network border exit). Originating hosts can also apply the DiffServ marking, and there are a number of advantages in doing so [e2e-QoS, DiffServ Arch].

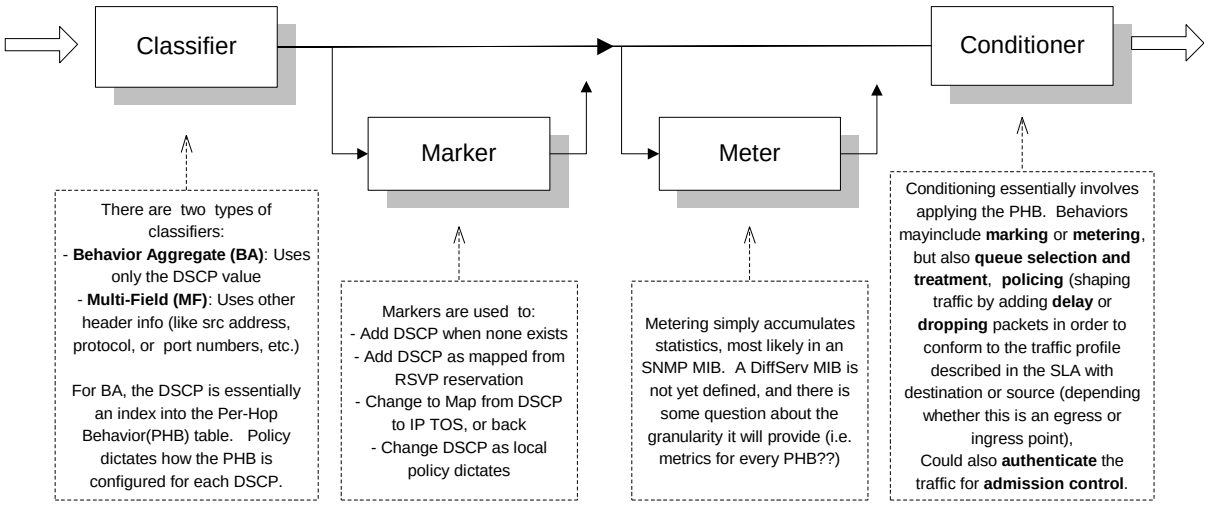

Figure 2: Differentiated Services Architecture, with a break out of some specifics. This functionality is enabled in every DiffServ enabled router, although not all functions are used all the time. Typically, border routers--at ingress and egress points--apply functions, but interior routers may also.

DiffServ assumes the existence of a service level agreement (SLA) between networks that share a border. The SLA establishes the policy criteria, and defines the traffic profile. It is expected that traffic will be policed and smoothed at egress points according to the SLA, and any traffic "out of profile" (i.e. above the upper-bounds of bandwidth usage stated in the SLA) at an ingress point have no guarantees (or may incur extra costs, according to the SLA). The policy criteria used can include time of day, source and destination addresses, transport, and/or port numbers (i.e. application Ids). Basically, any context or traffic content (including headers or data) can be used to apply policy.

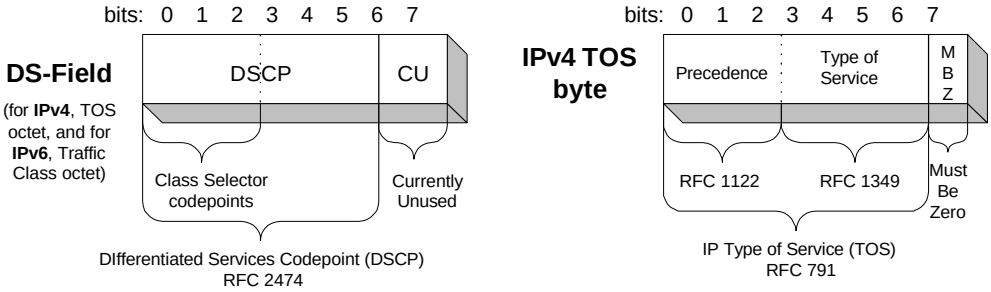

Figure 3: Differentiated Services Code Points (DSCP) redefine the IPv4 Type of Service byte. IP Precedence bits are preserved in class selector codepoints & PHBs, but TOS values are not.

When applied, the protocol mechanism that the service uses are bit patterns in the "DS-byte," which for IPv4 is Type-of-Service (TOS) octet and for IPv6 is the Traffic Class octet. As illustrated in Figure 3, although the DS field uses the IPv4 TOS byte [DiffServ Field], as defined in RFC 791 [IP], it does not preserve the original IPv4 TOS bit values as defined by RFC 1349 [TOS]. The IP Precedence bits (0-2) are preserved, however. And although it is possible to assign any PHB to the codepoints

in this range, the (required) default PHBs are equivalent to IP Precedence service descriptions, as described in detail in RFC 1812 [RouterReqs].

The simplicity of DiffServ to prioritize traffic belies its flexibility and power. When DiffServ uses RSVP parameters or specific application types to identify and classify constant-bit-rate (CBR) traffic, it will be possible to establish well-defined aggregate flows that may be directed to fixed bandwidth pipes. As a result, you could share resources efficiently and still provide guaranteed service. We will describe this type of usage later as we describe the various QoS architectures possible.

MPLS – Label Switching

Multi-Protocol Label Switching [MPLS] is similar to DiffServ in some respects, as it also marks traffic at ingress boundaries in a network, and un-marks at egress points. But unlike DiffServ, which uses the marking to determine priority within a router, MPLS markings (20-bit labels) are primarily designed to determine the next router hop. MPLS is not application controlled (no MPLS APIs exist), nor does it have an end-host protocol component. Unlike any of the other QoS protocols we describe in this paper, MPLS resides only on routers. And MPLS is protocol-independent (i.e., "multi-protocol"), so it can be used with network protocols other than IP (like IPX, ATM, PPP or Frame-Relay) or directly over data-link layer as well [MPLS Framework, MPLS Architecture].

MPLS is more of a "traffic engineering" protocol than a QoS protocol, per se. MPLS routing is used to establish "fixed bandwidth pipes" analogous to ATM or Frame Relay virtual circuits. The difference is arguable since the end-result is service improvement and increased service diversity with more flexible, policy-based network management control, all of which the other QoS protocols also provide.

MPLS simplifies the routing process (decreases overhead to increase performance) while it also increases flexibility with a layer of indirection. Here's a sketch of the process used by MPLS-enabled routers called a **Label Switching Router (LSR)**:

- At the first hop router in the MPLS network, the router makes a forwarding decision based on the destination address (or any other information in the header, as determined by local policy) then determines the appropriate label value -- which identifies the **Forwarding Equivalence Class (FEC)** -- attaches the label to the packet and forwards it to the next hop.

- At the next hop, the router uses the label value as an index into a table that specifies the next hop and a new label. The LSR attaches the new label, then forwards the packet to the next hop.

The route taken by an MPLS-labeled packet is called the **Label Switched Path (LSP)**. The idea behind MPLS is that by using a label to determine the next hop, routers have less work to do and can act more like simple switches. The label

represents the route and by using policy to assign the label, network managers have more control for more precise traffic engineering.

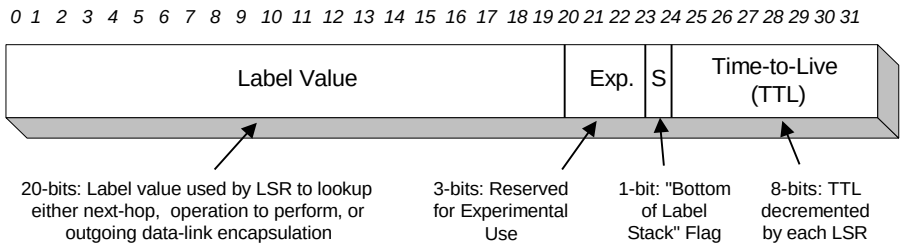

Figure 4: MPLS label stack entry used to "encapsulate" IP Header

Label processing is actually a bit more involved than described above, since labels can be "stacked" (to allow MPLS "routes within routes"), and labeled packets have a time-to-live value (TTL), as shown in Figure 4. The TTL works essentially the same way TTL in an IP header works: each router hop decrements the value by one until it hits zero. The difference is that when an MPLS TTL reaches zero, the action is label dependent (so unlike with IP, the datagram may not be discarded and an ICMP "TTL Exceeded" message may not be generated). Nonetheless, label processing is the relatively simple aspect of MPLS.

A more complex aspect of MPLS involves the distribution and management of labels among MPLS routers, to ensure they agree on the meaning of various labels. The Label Distribution Protocol (LDP) [MPLS LDP] is specifically designed for this purpose, but it is not the only possibility. There are proposals to use RSVP [MPLS LSPS], BGP [MPLS BGP], and PIM [MPLS PIM] possibly "piggy-backing" label management information, so the use of more than one protocol for label distribution is expected.

Although infrastructure details such as label distribution are important to mention, for most network managers they will be transparent. More relevant to MPLS for most network managers is the policy management that determines which labels to use where, and not how the labels are actually distributed.

SBM - Subnet Bandwidth Management

QoS assurances are only as good as their weakest link. The QoS "chain" is *end-to-end* between sender and receiver, which means every router along the route must have support for the QoS technology in use, as we have described with the previous QoS protocols. The QoS "chain" from *top-to-bottom* is also an important consideration, however, in two aspects:

- **Sender and receiver hosts** must enable QoS so applications can enable it explicitly or the system can enable it implicitly on behalf of the applications. Each OSI layer from the application down must also support QoS to assure that high-priority send and receive requests receive high priority treatment from the host's network system.

- **Local Area Network (LAN)** must enable QoS so high-priority frames receive high-priority treatment as they traverse the network media (e.g., host-to-host, host-to-router, and router-to-router). LANs are OSI Layer 2, the data-link layer, whereas the QoS technologies described previous to this have been Layer 3 (DiffServ) and above (RSVP & MPLS).

Some Layer 2 technologies have always been QoS-enabled, such as Asynchronous Transfer Mode (ATM). However, other more common LAN technologies such as Ethernet were not originally designed to be QoS-capable. As a shared broadcast medium or even in its switched form, Ethernet provides a service analogous to standard "best effort" IP Service, in which variable delays can affect real-time applications. However, the [IEEE] has "retro-fitted" Ethernet and other Layer 2 technologies to allow for QoS support by providing protocol mechanisms for traffic differentiation.

The IEEE 802.1p, 802.1Q and 802.1D standards define how Ethernet switches can classify frames in order to expedite delivery of time-critical traffic. The Internet Engineering Task Force [IETF] Integrated Services over Specific Link Layers [ISSLL] Working Group is chartered to define the mapping between upper-layer QoS protocols and services with those of Layer 2 technologies, like Ethernet. Among other things, this has resulted in the development of the "Subnet Bandwidth Manager" (SBM) for shared or switched 802 LANs such as Ethernet (also FDDI, Token Ring, etc.). SBM is a signaling protocol [SBM] that allows communication and coordination between network nodes and switches in the [SBM Framework] and enables mapping to higher-layer QoS protocols [SBM Mapping].

A fundamental requirement in the SBM framework is that all traffic must pass through at least one SBM-enabled switch. As shown in Figure 5, aside from the QoS-enabled application and Layer 2 (e.g., Ethernet), the primary (logical) components of the SBM system are:

- **Bandwidth Allocator (BA)**: Maintains state about allocation of resources on the subnet and performs admission control according to the resources available and other administrator-defined policy criteria.

- **Requestor Module (RM)**: Resides in every end-station and *not* in any switches. The RM maps between Layer 2 priority levels and the higher-layer QoS protocol parameters according to the administrator-defined policy. For example, if used with RSVP it could map based on the type of QoS (Guaranteed or Controlled Load) or specific Tspec, Rspec or Filter-spec values.

As illustrated in Figure 5, the location of the BA determines the type of SBM architecture in use: *Centralized* or *Distributed*. Whether there is only one or more than one BA per network *segment*, only one is the "Designated SBM" (DSBM) (Note:

there can be more segment per subnet). The DSBM may be statically configured or "elected" among the other Bas [SBM].

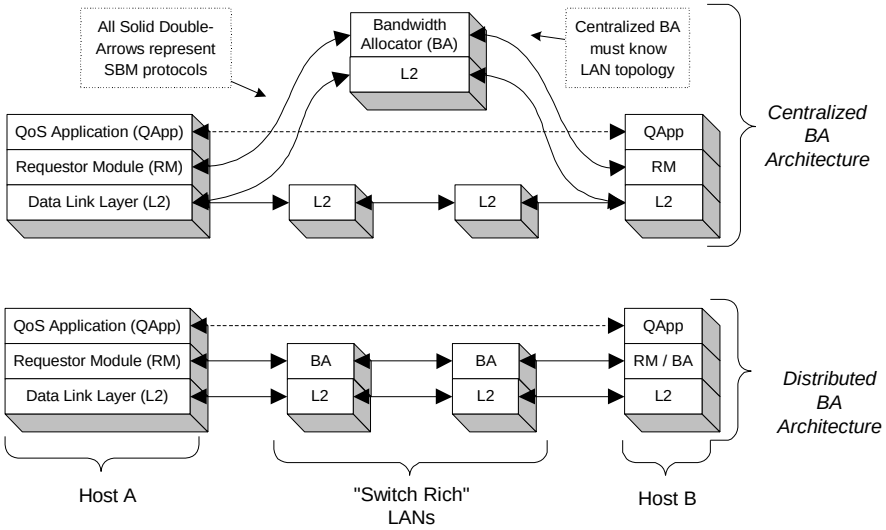

Figure 5: There are two forms of the Subnet Bandwidth Manager (SBM) architecture, in which the Bandwidth Allocator is either centralized or distributed [SBM Framework]

The SBM protocol provides an RM-to-BA or BA-to-BA signaling mechanism for initiating reservations, querying a BA about available resources, and changing or deleting reservations. The SBM protocol is also used between the QoS-enabled application (or its third-party agent) and the RM, but this involves use of a programming interface (API) rather than the protocol, so it simply shares the functional primitives. Although SBM protocol is designed to be QoS protocol-independent, so it is designed work with other QoS protocols such as ST-II, for example, the specifications use RSVP in their examples, as will we. Here is a simple summary of the admission control procedure of the SBM protocol:

0 DSBM initializes: gets resource limits (statically configured for now)

1 DSBM Client (any RSVP-capable host or router) looks for the DSBM on the segment attached to each interface (done by monitoring the "AllSBMAddress," the reserved IP Multicast address 224.0.0.17).

2 When sending a PATH message, a DSBM client sends it to the "DSBMLogicalAddress" (reserved IP Multicast address, 224.0.0.16) rather than to destination RSVP address.

3 Upon receiving a PATH message, a DSBM establishes PATH state in the switch, stores the Layer2 and Layer3 (L2/L3) addresses from which it came, and puts its own L2/L3 addresses in the message. DSBM then forwards the PATH message to next hop (which may be another DSBM on the next network segment).

4 When sending an RSVP RESV message, a host sends it to the first hop (as always), which would be the DSBM(s) in this case (taken from the PATH message).

5 DSBM evaluates the request and if sufficient resources are available, forwards to the next hop (else returns an error).

This sketch looks very much like standard RSVP processing in a router, however we omitted some significant details for the sake of simplicity. We will not attempt more detail here, but want to mention the TCLASS object that either a sender or any DSBM can add to a RSVP PATH or RESV message. It contains a preferred 802.1p priority setting and allows overriding a default setting, although any DSBM may change the value after receiving it. Routers must save the TCLASS in the PATH or RESV state, and remove it from the message to avoid forwarding it on the outgoing interface, but then they must put it back into incoming messages.

IEEE 802.1p uses a 3-bit value (part of an 802.1Q header) in which can represent an 8-level priority value. They are changeable and the specified bounds are only targets, but the default service-to-value mappings defined in [SBM Mapping] are:

0 Priority 0: Default, assumed to be best-effort service

1 Priority 1: Reserved, “less-than” best-effort service

2 Priority 2-3: Reserved

3 Priority 4: Delay Sensitive, no bound

4 Priority 5: Delay Sensitive, 100ms bound

5 Priority 6: Delay Sensitive, 10ms bound

6 Priority 7: Network Control

As with DiffServ, the simplicity of prioritization values belies the complexity that is possible. As we describe next in the QoS Architectures section, the flexibility that mapping provides allows for a wide variety of possibilities capable of supporting a wide range of QoS assurances and granularity.

QoS architectures

With the exception of the RSVP mapping we did to illustrate 802 SBM, the examples we’ve used in the descriptions of the QoS protocols described (RSVP, DiffServ and MPLS), have all shown each protocol used independently from end-to-end between sender and receiver. In real-world use, it is unlikely that these QoS protocols will be used independently, and in fact they are designed for use with other QoS technologies to provide top-to-bottom and end-to-end QoS between senders and receivers.

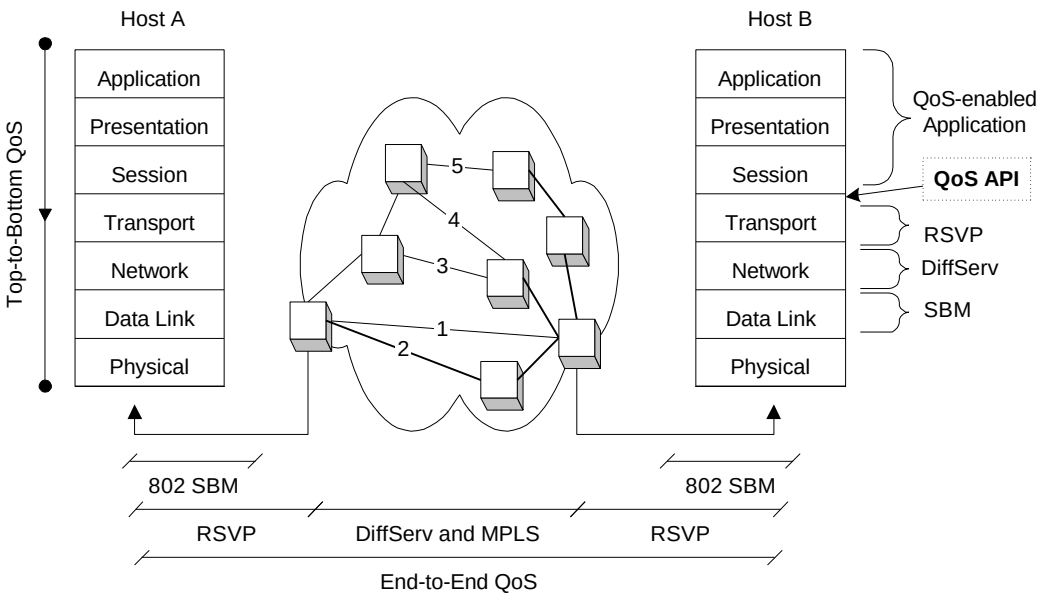

Figure 6: "End-to-end" and "top-to-bottom" in the real world means enjoying heterogeneity, and that includes QoS technologies, which were made to compliment each other end-to-end.

Most of the specifications for "gluing" these QoS pieces together are not standardized as yet, but work is well underway to define the various architectures that are possible—and necessary—to provide ubiquitous end-to-end QoS. In this section we describe a number of these architectures, highlight the issues and describe how they address them. Figure 6 provides a high-level view of how the pieces fit together, and Figure 7 provides another more detailed view of much the same idea. We reference both of these illustrations as we describe how the various protocols work together in concert to provide end-to-end and top-to-bottom QoS.

RSVP and DiffServ end-to-end model

Figure 7 shows a complete picture of how the QoS technologies can work together to provide "end-to-end QoS". [e2e-QoS]. Aside from the bandwidth broker -- which is still a new concept at this point in time -- this represents the model under development within the IETF community.

RSVP provisions resources for network traffic, whereas DiffServ simply marks and prioritizes traffic. RSVP is more complex and demanding than DiffServ in terms of router requirements, so can negatively impact backbone routers. This is why the "best common practice" says to limit RSVP's use on the backbone [RSVP Applicability], and why DiffServ *can* exist there.

DiffServ is a perfect compliment to RSVP as the combination can enable end-to-end quality of service (QoS). End hosts may use RSVP requests with high granularity (e.g. bandwidth, jitter threshold, etc.). Border routers at backbone ingress points can then map those RSVP "reservations" to a class of service indicated by a DS-byte (or source host may set the DS-byte accordingly also). At the backbone egress point, the RSVP provisioning may be honored again, to the final destination. Ingress points

essentially do traffic conditioning on a customer basis to assure that service level agreements (SLAs) are satisfied.

The architecture represented in Figure 7—RSVP at the "edges" of the network, and DiffServ in the "core"—has momentum and support. Work within the IETF DiffServ work group is progressing quickly, although initial tests have shown mixed results.

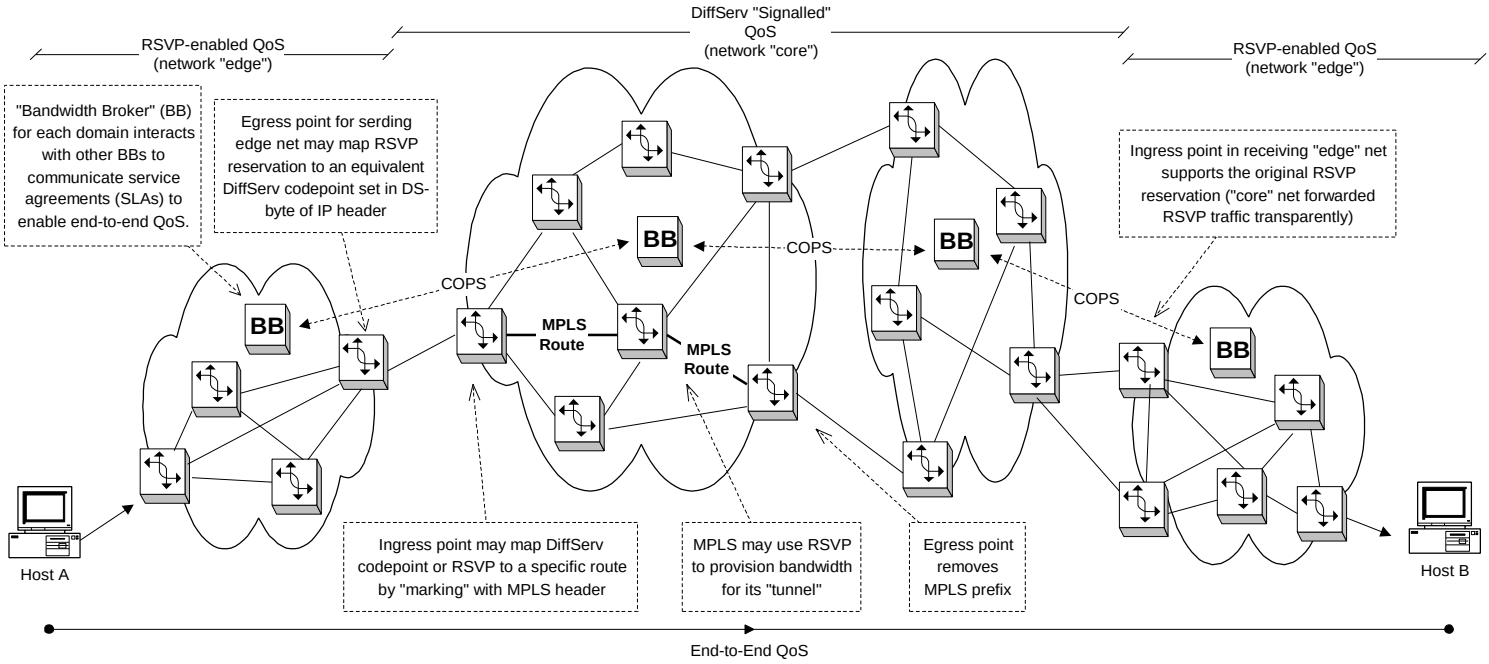

Figure 7: Illustrates the possible use of different QoS technologies under development--RSVP, DiffServ, MPLS, COPS and Bandwidth Brokers"--working cooperatively in various stratagies to enable end-to-end QoS

RSVP DCLASS object

There are default DiffServ settings defined to map from RSVP reservation values. However, it is also possible by use of the DCLASS object for an RSVP reservation message to carry along with it a preferred DiffServ codepoint value (this is analogous to the TCLASS object used with SBM). The DCLASS object may be added by the sender, or added along the way by any RSVP router [RSVP DCLASS].

The downside with DCLASS settings determined by the receiver or an intermediate hop is that these network elements may not know the best setting for use somewhere else in the network. RSVP reservations may be rejected if the DCLASS setting is deemed inappropriate, or any intermediate network element may simply ignore the DCLASS "suggestion" and mark packets according to local policy.

RSVP provisioning for aggregates

As we described, by classifying traffic flows, DiffServ and MPLS create what are effectively "pipes" for these aggregates. For these pipes to provide any "service quality" better than standard best effort, traffic on these virtual pipes must not exceed capacity. The problem is that neither DiffServ nor MPLS have specific protocol

mechanics for detecting how much bandwidth they need and then allocating the necessary resources for dedicated usage. Only RSVP is designed to do that.

Hence, although RSVP was originally designed to allocate bandwidth for individual application flows, it is very important for allocating bandwidth to accommodate the needs of traffic aggregates as well [RSVP MPLS]. This need highlights the challenge, however, for network engineers using DiffServ or MPLS to know the bandwidth demands to anticipate, so they can make the appropriate resource reservation request. Additionally, senders and receivers at both ends of the virtual pipes must make these reservation requests so the appropriate PATH and RESV messages can be sent from and to the appropriate unicast locations.

"A key problem in the design of RSVP version 1 is, as noted in its applicability statement, that it lacks facilities for aggregation of individual reserved sessions into a common class. The use of such aggregation is required for scalability" [RSVP Aggregation]. So in addition to using RSVP to provision for QoS aggregates, another consideration is using RSVP to provision for RSVP aggregates.

MPLS for RSVP

As described in [MPLS RSVP], there's a proposal to use an EXPLICIT_ROUTE object in RSVP to pre-determine paths taken by label-switched RSVP flows. These flows use virtual pipes established through MPLS-enabled routers (LSRs), as illustrated in Figure 7. Even without the EXPLICIT_ROUTE object in RSVP reservations, it is possible for MPLS to assign labels according to the RSVP flowspecs.

In either case, the effect is a significant simplification of RSVP support on the MPLS routers. By referencing MPLS labels, LSRs need not manage RSVP state [MPLS Architecture].

MPLS for DiffServ

As might be expected, because DiffServ and MPLS are similar with respect to the qualitative QoS they enable (i.e. classification), mapping DiffServ traffic on MPLS "pipes" (LSPs) is relatively simple. It is, but there are still DiffServ-specific considerations.

To support DiffServ's per-hop model, an MPLS network operator needs to allocate a set of aggregate forwarding resources for each DiffServ forwarding class in each MPLS router (LSR) and assign labels. Additionally an LSR may need to associate the packet with a particular drop-precedence (which could be stored in the "experimental" (Exp) field of the MPLS header) [MPLS DiffServ].

QoS support for Multicast

IP Multicast is a requirement, not an option, if the Internet is going to scale. It is a natural compliment to QoS for support of one-to-many audio and video "broadcasts" over the Internet, so support for multicast has always been a fundamental requirement

in the design of QoS protocols. Although allowances have always been made in the initial designs of QoS protocols, full support of QoS for multicast is still not standardized or fully understood yet. There are a number of issues involved with multicast support that we describe here, as we summarize the current state of support of QoS for multicast for each of the QoS protocols we've focused on in this paper.

RSVP support for Multicast

As we mentioned earlier in this paper, the initial design for RSVP and Integrated Services took IP Multicast support into consideration by making the reservations receiver-based. One aspect of multicast that makes it a challenge to support is that the receivers that comprise a multicast group may vary widely in their capabilities with regard to the downstream bandwidth available to them. This heterogeneous receivership is likely to have a wide variety of reservation requests, specific to the path their data will flow downstream. Hence, it is essential that each receiver be allowed to specify a different reservation according to its needs.

Another aspect of the Integrated Services design relevant to multicast in general and heterogeneous receivers specifically is the ability to set filter specifications. By allowing this, hierarchical data may be possible. Hierarchically encoded data streams are designed so that when less bandwidth is available, receivers can still get a usable signal, though with lower fidelity. Filter specifications could reserve bandwidth for the portion of the stream a lower-bandwidth receiver is capable of receiving.

The great challenge that RSVP presents and which is not yet fully understood deals with ordering and merging reservations [IntServ Service Spec]. As yet no standards are published, but there is at least one simulation reference [RSVP Multicast] and an examination of some of the problems possible with multicast reservation mergers [RSVP Killers].

DiffServ support for Multicast

The relative simplicity of Differentiated Services makes it a better (easier and more scalable) fit for multicast support, but there are still challenges involved. Specifically, estimating the traffic is a challenge due to the dynamic nature of group memberships and to the fact that although a multicast distribution tree may have a single ingress point, it will often have multiple egress points (which can change as membership changes). Work is still underway in this area.

MPLS support for Multicast

MPLS support for Multicast is a subject of intense development effort, but no standards have emerged as yet. There are a number of relevant Internet Drafts on the subject of IP Multicast support in MPLS networks and multicast traffic engineering [MPLS Multicast].

SBM support for Multicast

SBM has explicit support for multicast, and as described previously, SBM utilizes IP Multicast as part of the protocols. There are no issues with SBM multicast support

assuming support for IGMP in SBM-enabled switches, so multicast traffic is only forwarded to segments where group members reside.

Policy-enabled QoS

QoS provides differentiation of traffic and the services provided to that traffic. This means that some traffic gets improved service and (inevitably) other traffic gets degraded service. Naturally, everyone would want the improved service for most of their traffic, but everyone can't have it (or at least not for free). Thus, QoS has a need for policy (the decision about which flows are entitled to which service) and policy creates a need for user authentication (to verify user identification).

Among the QoS protocols, only RSVP has explicit provisions for policy support, which we describe next. With other QoS protocols, policy is applied at network border location, which may be located either at a layer transition in a TCP/IP stack implementation (for example, as a layer 3 IP packet is passed to a layer 2 network driver) based on identifiable characteristics of the packet. We describe these border locations and their use of policies to define varying services in other QoS Forum papers on Policy.

RSVP policy object

The draft [RSVP Policy] describes the protocol mechanisms and algorithms for policy support in RSVP, and is meant to update the policy object description in RFC 2205 [RSVP]. Any policy-enabled RSVP router can generate, modify or remove POLICY_DATA objects.

Policy objects contain an option list and policy element list. The options are either a FILTER_SPEC object to preserve the original flow/policy association or a SCOPE object to prevent "policy loops". The policy elements are opaque and understood only by the RSVP routers that use them; the Internet Assigned Numbers Authority (IANA) will maintain a registry of policy element values and their meaning.

Conclusion

Until now, IP has provided a "best-effort" service in which network resources are shared equitably. Adding quality of service support (QoS) to the Internet raises significant concerns, since it enables differentiated services that represent a significant departure from the fundamental and simple design principles that made the Internet a success. Nonetheless, there is a significant need for IP QoS and protocols have evolved to address this need.

Since applications have a range of QoS requirements in terms of the granularity of determinism and the level of guarantee, there are also a variety of services and protocols available. In addition to having application and host involvement, some of these protocols are designed for use transparently within the network.

These varied protocols and mechanisms and services are all designed to work together. By mixing and matching their capabilities in a variety of possible architectures, the goal of end-to-end and top-to-bottom QoS-enabled communications is getting closer to reality every day. The standards are not fully developed yet, and there are still some important considerations such as multicast support that require further attention, but deployment is already underway on many IP networks.

References

[DiffServ] IETF "Differentiated Services" working group. See http://www.ietf.org/html.charters/diffserv-charter.html and http://www.ietf.org/ids.by.wg/diffserv.html

Also see http://www.ietf.org/internet-drafts/1id-index.txt and search for "diff" to see the large number of diffserv-relevant Internet Draft submissions from individuals.

[DiffServ AF] J. Heinanen, F. Baker, W. Weiss, J. Wroclawski, "Assured Forwarding PHB Group", RFC 2597, June 1999

[DiffServ Arch] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, "An Architecture for Differentiated Services", RFC 2475, December 1998

[DiffServ EF] V. Jacobson, K. Nichols, K. Poduri, "An Expedited Forwarding PHB", RFC 2598, June 1999

[DiffServ Field] K. Nichols, S. Blake, F. Baker, D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, December 1998

[e2e] J. Saltzer, D. Reed, D. Clark, *End to End Arguments in System Design*, ACM Transactions in Computer Systems, November 1984. See http://www.reed.com/Papers/EndtoEnd.html

[e2e-QoS] Y.Bernet, R.Yavatkar, P.Ford, F.Baker, L.Zhang, K.Nichols, M.Speer, R. Braden, *Interoperation of RSVP/Int-Serv and Diff-Serv Networks*, February 1999, <draft-ietf-diffserv-rsvp-02.txt>, Work in Progress

[IEEE] The Institute of Electrical and Electronics Engineers (IEEE) is a formal body in which network technologies such as Ethernet (among many other things) are specified and standardized.

For information on the 802 LAN standards such as 802.1p, 802.1Q and 802.1D, see http://standards.ieee.org/catalog/IEEE802.1.html

[IETF] The Internet Engineering Task Force is a loose confederacy of volunteers from the network industry and academia that uses "running code and rough consensus" to establish protocol standards for the Internet. See http://www.ietf.org

[IntServ] IETF "Integrated Services" working group. See http://www.ietf.org/html.charters/intserv-charter.html and http://www.ietf.org/ids.by.wg/intserv.html

[IntServ Controlled] J. Wroclawski, *Specification of the Controlled-Load Network Element Service*, RFC 2211, September 1997

[IntServ Guaranteed] S. Shenker, C. Partridge, R. Guerin, *Specification of Guaranteed Quality of Service*, RFC 2212, Sept 1997

[IntServ Parameters] S. Shenker, J. Wroclawski, "General Characterization Parameters for Integrated Service Network Elements", RFC 2215, September 1997

[IntServ Service Spec] S. Shenker, J. Wroclawski, *Network Element Service Specification Template*, RFC 2216, Sept 1997

[IP] J. Postel, "Internet Protocol – DARPA Internet Program Protocol Specification", RFC 791, September 1981

[ISSLL] Integrated Services over Specific Link Layers, see http://www.ietf.org/html.charters/issll-charter.html

[MPLS] IETF "Multiprotocol Label Switching" working group. See http://www.ietf.org/html.charters/mpls-charter.html and http://www.ietf.org/ids.by.wg/mpls.html

[MPLS Architecture] E. Rosen, A. Viswanathan, R. Callon, "Multiprotocol Label Switching Architecture", April 1999, <draft-ietf-mpls-arch-05.txt>, Work in Progress

[MPLS BGP] Y. Rekhter, E. Rosen, "Carrying Label Information in BGP-4", February 1999, <draft-ietf-mpls-bgp4-mpls-02.txt>, Work in Progress

[MPLS DiffServ] J. Heinanen, "Differentiated Services in MPLS Networks", June 1999, <draft-heinenen-diffserv-mpls-00.txt>, Work in Progress

[MPLS Framework] R. Callon, N. Freedman, A. Fredette, G. Swallow, A. Viswanathan, "A Framework for Multiprotocol Label Switching", June 1999, <draft-ietf-mpls-framework-03.txt>, Work in Progress

[MPLS LDP] B. Thomas, N. Feldman, P. Doolan, L. Andersson, A. Fredette, "LDP Specification", June 1999, <draft-ietf-mpls-ldp-05.txt>, Work in Progress

[MPLS LSPS] D-H Gan, T. Li, G. Swallow, L. Berger, V. Srinivasan, D. Awduche, "Extensions to RSVP for LSP Tunnels", March 1999, <draft-ietf-mpls-rsvp-lsp-tunnel-02.txt>, Work in Progress

[MPLS Multicast] D. Ooms, W. Livens, B. Sales, M. Ramalho, A. Acharya, F. Griffoul, "Framework for IP Multicast in MPLS", June 1999, <draft-ietf-mpls-multicast-00.txt>, Work in Progress

L. Wu, D. Ooms, A. Mondrus, "MPLS Multicast Traffic Engineering", June 1999, <draft-wu-mpls-multicast-te-00.txt>, Work in Progress

C-Y. Lee, K. Carlberg, B. Akyol, "Engineering Paths for Multicast Traffic using MPLS", June 1999, <draft-leecy-multicast-te-00.txt>, Work in Progress

H. Hummel, S. Loke, "Explicit Tree Routing", June 1999, <draft-hummel-mpls-explicit-tree-01.txt>, Work in Progress

A. Acharya, F. Griffoul, F. Ansari, "IP Multicast Support in MPLS Networks", February 1999, <draft-acharya-ipsofacto-mpls-mcast-00.txt>, Work in Progress

[MPLS PIM] D. Farinacci, Y. Rekhter, E. Rosen, "Using PIM to Distribute MPLS Labels for Multicast Routes", June 1999, <draft-farinnacci-mpls-multicast-00.txt>, Work in Progress

[MPLS RSVP] D.Awduche, D.Gan, T.Li, G.Swallow, V. Srinivasan, *Extensions to RSVP for Traffic Engineering*, August 1998, <draft-swallow-mpls-rsvp-trafeng-00.txt>, Work in Progress

[Partridge] C. Partridge, Gigabit Networking, Addison-Wesley, February 1994, ISBN 0-201-563339

[Queuing] Len Kleinrock has an extensive bibliography on traffic queuing and buffering at http://millennium.cs.ucla.edu/LK/Bib/

Sally Floyd has information on queue management and her research on Class-based Queuing (CBQ) at http://www.aciri.org/floyd/cbq.html and on RED at ../red.html

[RouterReqs] F. Baker, "Requirements for IP Version 4 Routers", RFC 1812, June 1995

[RSVP] R. Braden, L. Zhang, S. Berson, S. Herzog, S. Jamin, "Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification", RFC 2205, September 1997

[RSVP Aggregation] F. Baker, "Aggregation of RSVP for IPv4 and IPv6 Reservations", June 1999, <draft-baker-rsvp-aggregation-01.txt>, Work in Progress

[RSVP Applicability] A. Mankin, et al, "Resource ReSerVation Protocol (RSVP) Version 1 Applicability Statement – Some Guidelines on Deployment", RFC 2208, September 1997

[RSVP DCLASS] Y. Bernet, "Usage and Format of the DCLASS Object with RSVP Signalling", February 1999, <draft-bernet-dclass-00.txt>, Work in Progress

[RSVP IntServ] J. Wroclawski, "The Use of RSVP with IETF Integrated Services," RFC 2210, September 1997

[RSVP Killers] M. Talwar, "RSVP Killer Applications", January 1999, <draft-talwar-rsvp-kr-01.txt>, Work in Progress

[RSVP MPLS] D. Awduche, L. Berger, D. Gan, T. Li, G. Swallow, V. Srinivasan, "Extensions to RSVP for LSP Tunnels", March 1999, Work in Progress

[RSVP Multicast] M. Pullen, R. Malghan, L. Lavu, G. Duan, J. Ma, RFC 2490

[RSVP Policy] S. Herzog, "RSVP Extensions for Policy Control", April 1999, <draft-ietf-rap-rsvp-ext-05.txt>, Work in Progress

[SBM] R. Yavatkar, D. Hoffman, Y. Bernet, F. Baker, "SBM (Subnet Bandwidth Manager): A Protocol for RSVP-based Admission Control over IEEE 802-style networks", May 1999, <draft-ietf-issll-is802-sbm-08.txt>, Work in Progress

[SBM Framework] A. Ghanwani, J. W. Pace, V. Srinivasan, A. Smith, M. Seaman, "A Framework for Integrated Services Over Shared and Switched IEEE 802 LAN Technologies", June 1999, <draft-ietf-issll-is802-framework-07.txt>, Work in Progress

[SBM Mapping] H. Seaman, A. Smith, E. Crawley, J. Wroclawski, "Integrated Services Mappings on IEEE 802 Networks", June 1999, <draft-issll-is802-svc-mapping-04.txt>, Work in Progress

[TOS] Almquist, P. “Type of Service in the Internet Protocol Suite”, July 1992, RFC 1349