Internet QoS: Past, Present, and Future

Daniel Reid and Michael Katchabaw
Department of Computer Science
University of Western Ontario
London, Ontario, N6A 5B7, Canada
Email: dreid28@csd.uwo.ca, katchab@csd.uwo.ca

Technical Report
Department of Computer Science,
The University of Western Ontario
June 2004

Table of Contents

Abstract: In the past, numerous attempts at providing widespread Quality of Service (QoS) have failed. The two most popular architectures proposed, Integrated Services and Differentiated Services, suffer from scalability and flexibility concerns respectively. Newer QoS-enabling technologies such as MPLS have emerged, and look very promising in providing widespread QoS support. To provide a more general and multi-purpose QoS signaling protocol, the Next Steps in Signaling Working Group was formed recently. This paper will provide a broad overview of previous and more currently proposed architectures, along with a brief discussion of the future direction of QoS.

1. Introduction

Originally designed as a best-effort service, the Internet has grown significantly, and so have its needs. Today, many applications require a service level better than best-effort can offer. QoS can provide these enhanced services.

QoS is an expression used to depict the overall experience a user or application receives over a network [1]. Since the inception of computer networks, numerous approaches have been attempted to provide improved QoS to end users. Over the Internet, QoS is typically measured in terms of bandwidth, loss, delay, jitter, and availability.

The traffic flowing through today's networks is extremely diverse, with each type requiring differing levels of QoS. The diversity of such data flows are the result of numerous data networks which have migrated to IP for transport over the past decade. The IP protocol, which now encompasses the majority of data flows, has since conquered reliability issues, but it was not originally designed for the strict requirements needed by some applications today. Little consideration has been placed to account for delay and bandwidth requirements of these applications. The connectionless properties of IP networks result in unpredictable best-effort services, causing significant problems with new QoS sensitive applications such as teleconferencing and IP telephony. QoS models strive to address these issues, taking a best-effort network and transforming it into one which can provide bandwidth and delay assurances to its applications.

Since IP does not intrinsically support any preferential treatment of traffic, the onus has been placed on service providers and network managers to make their components QoS-aware. Large intranets are often subject to the same policies and consequently the QoS procedures are easier to deploy. While it may be relatively easier compared to the Internet, the installation of such infrastructures are currently not practical in a large number of cases due to the high costs associated with specialty routers and the capable network engineers to manage them. Many current networks are simply pushing for greater network capacity to alleviate congestion problems. While this option is currently cheaper than deploying a QoS infrastructure, it does not allow any prioritization of flows. Allowing traffic to be treated equally is not desirable since the unpredictable nature of network traffic will usually not yield QoS requirements regardless of the available bandwidth; unless of course the network has infinite resources. Throwing more bandwidth at the problem is merely a short term fix, and will not address any long term issues.

The Internet, on the other hand, originally designed for a best-effort service, has made little progress in respect to providing QoS because of its heterogeneity. Most current and emerging standards are still in their early stages, but soon the rising need for QoS will propel them to the forefront.

There have been several fields of thought on providing QoS to the end users. By far, the two most popular and accepted philosophies are the Integrated Services Model (IntServ) and Differentiated Services Model (DiffServ). More recently, a QoS-enabling technology named Multi Protocol Label Switching (MPLS) has emerged and looks very promising. Section two, three, and four will examine each of these models in detail. Section five will introduce several relatively newer models which have attempted to alleviate the problems these models have. Finally, the last section will discuss future work in the area of QoS; in particular the Next Steps in Signaling Working Group.

2. Integrated Services

The Integrated Services model [2] is primarily differentiated from others through the use of resource reservation. Deployment of QoS is done on a per-flow basis, with applications performing the reservation requests. The Integrated Services Working Group has defined several different service classes that can commit an arbitrary QoS level to a uniquely identified session of packets; also referred to as a *data flow*.

Since reservations are made on a per-flow basis, there is a need for per-flow state to be installed in routers participating in the reservation. Along the control plane, there is need for per-flow signaling. On the data plane, there is a need for per-flow identifiers and scheduling algorithm parameters.

To implement the IntServ model a signaling protocol is needed, along with a traffic controller. The traffic controller is comprised of an admission control routine, classifier, and packet scheduler. The following subsections will provide details of each, with a discussion concerning flow descriptors included in-between.

2.1 Signaling Protocol

In order to facilitate a reserved flow, a signaling protocol must be used to contact participating nodes for admission control, allocation and de-allocation of resources, and in the case of a soft-state protocol, to refresh the flow periodically. The IntServ model provides thorough constraints on how this should take place, but intentionally does not specify which method to use, or how the flows are to be identified. A signaling protocol does not perform any resource reservation itself, but rather is used as a means of carrying the information needed to do so. While more basic approaches simply utilize network management procedures such as SNMP or manual configuration of the routers, there have been numerous more flexible protocols developed. Currently, the most commonly recognized and only IETF standardized reservation protocol for IP networks is the Resource Reservation Protocol (RSVP). Citing the complexity and scalability concerns of RSVP, several other lightweight approaches have been developed.

Short descriptions and comparative analysis of existing end-to-end signaling protocol solutions follow. Table 1 can be found at the end of the section summarizing several key aspects of each protocol.

2.1.1 ST-II/+

As early as 1979, with the development of the ST Stream Protocol [3], thought has been put into providing guaranteed services over IP networks. Over a decade later in 1990, the second version, ST-II [4], was being developed. ST-II is a simplex, homogeneous, sender-initiated resource reservation system which uses distribution trees for multicasting[5]. In 1995, a new specification, referred to as ST-II+ [6], was released that has further extensions such as allowing either sender or receiver reservations. Unlike the other protocols to be discussed, it is a fully functional inter-networking protocol meant to replace IP. The simplex properties allow it to only reserve resources in one direction; that is, there is a distinct sender and receiver. To provide a reserved bidirectional flow, two separate reservations must be made. It is a connection-oriented protocol, requiring state information for each connection to be held in participating nodes with no timeout period, also referred to as a hard-state connection. Reliability is accomplished through hop-by-hop acknowledgements, and the ability to retransmit any lost control messages.

Operation

To make a reservation, a *Connect* message containing the set of recipients and *flow descriptor*, detailing QoS requirements, is sent out. Each intermediary node receiving this Connect message determines via its routing protocol the next node(s), records this forwarding state, and attempts to make the resource reservation. Each node, including the receiver, can reduce the flow specification if it can not allocate the requested resources.

After receiving a Connect message, each receiver returns either an *Accept* or *Refuse* message. When an Accept message is traversing backwards through the network towards the sender, if the flow specifications have since been reduced, the node changes the resource allocation accordingly. The initial sender waits for this reply (or several replies if multicasting) before sending any data. When received, if the Accept message contains reduced flow specifications, the sender can either continue, or send a *Disconnect* message to the receiver; thus, ending the flow.

When multicasting, the membership of receivers can be dynamically changed if hosts need to be included or removed. A host wishing to join a multicasting group is typically required to send an unspecified message type to the sender, informing them to send a Connect message. Similar to the initial setup, the sender must take a look at the Accept message and decide to accept or reject it. The resources allocated for the stream must remain homogeneous.

To be removed from a multicast or unicast session, the source will either send a disconnect message that specifies the individual receivers, or will set a tear down flag to end the entire flow. Individual receivers may opt to remove themselves by sending a Refuse message

Advantages

Since ST-II is a functional inter-networking protocol, the added complexity of being able to handle both data and control messages allows it to combine the knowledge of resource reservation with routing information. Thus, it can make routing decisions based on resource availability in neighbouring nodes [7]. This property, combined with its hard-state, can allow guaranteed QoS even in the case of route changes. Due to its hard-state, there is also less network overhead required.

Disadvantages

While a functional inter-networking protocol provides far more complex operations, a tradeoff exists, in that it is less modular and cannot be used with different routing and transmission protocols [7]. Its incapability to use different transmission and routing protocols severely limits its ability to become widely accepted due to its inflexibility, and incredible complexity since it is responsible for reliably maintaining router state. In addition, costly mechanisms must be present to handle errors due to its hard-state.

The homogeneous nature of ST-II does not allow custom resource reservations made per sender receiver pair. This nature makes the assumption that all receivers and links have the same capability, which is far from true in the majority of cases. Respectively, receivers and links have differing abilities and capacities for processing data. Further reducing flexibility is the fact that differing end points often require, or request, distinct QoS levels.

In the case of multiple senders, separate reservations are needed. This results in resources being reserved along multiple trees, even when many branches share common links. In some cases, such as distributed voice conversations when there is typically only one speaker at a time, this can lead to a significant waste of resources. In general, ST-II has been designed for a small number of members when multicasting, and has been shown to not scale well in large groups [5].

2.1.2 Resource Reservation Protocol

In 1993, the Resource Reservation Protocol (RSVP) was introduced [8]. It is currently the only IETF standardized resource reservation signaling protocol. It is similar to ST-II in that it is a simplex protocol and contains mechanisms to provide robustness to changing network dynamics. RSVP distinguishes itself from other protocols by allowing more flexibility and scalability when dealing with multicasting. Unlike the sender-initiated ST-II, RSVP achieves this scalability through its receiver oriented design. The receiver is responsible to provide the flow specifications, and is also responsible for periodically refreshing the soft-state reservation to keep it active. Alternatively, ST-II does not require refreshing, but instead moves this complexity into the network itself resulting in a much more complex protocol.

One common misconception is that RSVP is a routing protocol. Rather than replicating complex and costly functions, RSVP processes consult local existing routing protocol(s). It has also been designed to handle future unicast and multicast routing protocols as well. Due to the reliance on other routing protocols, it is known that routes will sporadically change for a number of reasons. RSVP has been designed to automatically refresh reservations in these cases provided that the resources are available on the new path.

When multicasting, RSVP has been designed to allow the end users to specify custom QoS needs. By doing so, the reserved aggregate resources can accurately reflect the resources actually needed, resulting in less waste. The joining of a multicast group is done through out-of-band messages; normally through the Internet Group Management Protocol (IGMP).

RSVP is unique in its support of *channel changing* for multicasting sessions. This feature reduces waste (and perhaps bandwidth charges) by allowing receivers to specify what particular data they wish to be delivered. These parameters are left at the nodes, where the upstream filtering occurs. Quite often during a multicasting session there are times when a receiver doesn’t wish to receive all the data, such as choosing individual audio streams from a larger feed. This finer grained control is met with much greater complexity in the networks.

Operation

To make a reservation, it is assumed that a sender has first sent the receiver a *PATH* message. The PATH message traverses down the data path, and at each node sets up the path state including the address of the last hop. As mentioned previously, RSVP has a soft-state, so periodic refreshes are required from the end systems to keep the state fresh. Without a refresh, the state in RSVP nodes will time out and be removed.

Upon receiving a PATH message, the reservation request is made and sent from the receiver to the sender in the form of a *RESV* message. Using the path state stored in the routers, this reservation message travels back through the reverse path. At each node, two actions are taken; first, the reservation request is put through admission and policy control. If either fails, an appropriate error message is returned to the receiver. Upon success, the flow specifications are extracted from the message and the resources are then allocated. The reservation request is then forwarded further upstream to the next hop. When multicasting, reservations being sent to the same sender are merged as they travel upstream, by only forwarding the largest resource reservation. Receivers who wish a confirmation may do so, but must also realize that the receipt of such a confirmation is not guaranteed.

Reservation Styles

To dictate how resource reservation requests from heterogeneous receivers in a multicast group should be aggregated in the network efficiently, RSVP introduces *reservation styles*. There are two attributes which specify the reservation style; the sharing attribute and sender selection attribute.

The sharing attribute chooses whether the resource reservation will be *shared* among the receivers, or if there will be *distinct* reservations. Shared reservations are useful in audio conferencing where there will normally be one person speaking at a time.

The sender selection attribute determines how the senders are selected. With *explicit*, a filter spec is used to determine a set of senders. With *wildcard*, there is no filter spec, and the whole set is used.

Names are given to each filter type; *Fixed Filter* (distinct, explicit), *Shared Explicit* (shared, explicit), and *Wildcard Filter* (shared, wildcard).

Advantages

The design of RSVP is to modularize as much as possible; therefore, it only transmits control messages containing resource reservation information, and not the data itself (as would ST-II in comparison). By doing so, RSVP can be used in conjunction with many different routing and data transfer protocols [7].

A receiver oriented soft-state design allows heterogeneity in the data flows, such as the capability to reserve differing amounts of resources per receiver. It also allows hosts to receive differing data streams sent to the same multicast group, and the ability to change streams without having to change its reservation [8]. Each receiver will make its own reservation with RSVP resolving any differences. If multiple receivers opt for differing levels of QoS, RSVP will merge these requests by taking the maximum value. To achieve better use of network resources, there are also different reservation styles that permit applications in the same multicast group to stipulate how the flows should be aggregated. These are discussed shortly.

Soft-state also provides better dynamic adaptability and robustness, such as the automatic adaptation of routing changes and the ability for on-the-fly membership changes in large multicast groups to take place seamlessly; unlike ST-II, which was designed for much smaller multicasting groups. It has also been shown that this receiver oriented design and the merging capabilities in RSVP reduce load as one gets closer to the source, limits interactions between end points, and generally reduces network protocol overhead [5].

The scalability in allowing large multicast groups is due to the receiver initiated approach, which does not require resource reservation requests to progress all the way to the source; only to join as they reach a branch in the multicast tree.

Disadvantages

As mentioned previously, RSVP has been designed to run independent from routing protocols, using any protocol available. The drawback to such modularization is that separating the knowledge of resource reservation from the routing information, results in the inability to make routing decisions based on the resource availability in neighbouring nodes [7].

That aside, there are two major problems: complexity and scalability. The complexity of the protocol results in large overhead when processing messages, while the lack of general scalability results in too much bandwidth and storage being consumed as the number of flows increase. Both of these limitations result in serious problems for the IntServ model, since RSVP is the standardized signaling protocol.

2.1.3 Yet Another Sender Session Internet Reservation Protocol

In 1998, the YEt another Sender Session Internet Reservation protocol (YESSIR) [9] was developed to simplify the complexity RSVP, yet still keeping many of its features. It is a sender initiated, simplex, soft-state protocol that is run over an in-band protocol, the Real Time Control Protocol (RTCP). It was noted by the designers that a large chunk of applications requiring QoS were multimedia oriented using the Real-time Transport Protocol (RTP), and thus decided to extend this increasingly popular protocol with QoS support.

RTP is used for actual data transport, while RTCP is the control protocol for RTP sessions. RTCP is also used periodically for sender and receiver reports, indicating session characteristics such as packets transmitted, packet loss, delay, etc. While RTP alone does not include any resource reservation abilities, YESSIR embeds these abilities by taking advantage of the periodic RTCP sender reports.

It has been shown that YESSIR resource reservation is three times faster, and the processing overhead on refreshes are half that of RSVP [10]. Bandwidth consumption is also lower, since YESSIR does not require additional IP or transport headers.

Operation

Reservation requests from the sender are sent through RTCP messages, and arrive at routers through the use of the router alert option. YESSIR-aware routers, and those that support the router alert option, process the message, otherwise ignore and forward them. The resource reservations are made through the use of a flow descriptor, existing RTCP information, or a combination of both.

If a reservation cannot be made, the reason for failure is optionally attached to the sender report. Upon receipt of the sender report, the receiver attaches any failure messages to the receiver report. Upon receiving a report with errors, the sender can stop the session, continue the session if using partial reservations (discussed shortly), or transmit and request less bandwidth.

Like RSVP, the soft-state of YESSIR requires periodic refreshes of sender reports to keep reservations from being deleted and to adapt to routing changes. To teardown a resource reservation properly, an RTCP *BYE* message is used, releasing the reservation and any state associated with it.

Advantages

YESSIR messages are periodically transported by RTCP through the use of the router alert option. By doing so, YESSIR does not require a new protocol to be developed by taking advantage of piggybacking on existing messages. This technique has been shown to reduce both processing and protocol overhead costs at the router [9].

One distinct advantage YESSIR has over other signaling protocols is the option to allow partial reservations. In other reservation systems, requests are either accepted or refused. Upon refusal end hosts will typically resend the same request (or one with lesser QoS requirements) quite often resulting in added network cost. If specified, YESSIR does not require all nodes to accept a reservation. Nodes which cannot make the reservation will classify the flow as best effort, and forward the request without any error. During periodic refreshes, these nodes will again have the opportunity to make the resource reservation. With this approach, a partial reservation can be made with the hopes that more nodes will be added, and a full reservation will eventually be the end result.

Routers often use the added YESSIR flow descriptor to make a reservation, but can also make a reservation without it. In *measurement mode*, routers can use the typical data commonly found in the RTCP sender reports, including transmission statistics such as byte counts and timestamps, to make a reservation. Since RTCP already includes these details, there is no burden on the router to estimate rates through counting packets, as other measurement based admission controls do.

Alternatively, it has been suggested that RTP packets be periodically marked with the router alert option, so as to extract payload identification to make the appropriate reservations.

While not providing all the reservation styles RSVP is capable of, YESSIR uses a simplified approach, using only *individual* and *shared* styles and controlling them from the sender side rather than receiver. Individual style requires each sender to make its own reservation, while shared style allows the senders in an RTP session to share the reservation.

Disadvantages

The most obvious disadvantage to YESSIR is that it can only be used with RTP sessions. While multimedia applications using RTP are becoming increasingly more popular, other QoS sensitive applications would not be able to take advantage of this protocol. Additionally, there is extra support required in the applications.

Although partial reservations are a handy for some applications, they can also lead to potential problems. It is possible that multiple reservations being made at the same time will result in each being given only partial reservations. This is referred to as fragmentation and can result with many reservations being given undesirable quality. This issue is currently being investigated. Additionally, when using partial reservations an advantage is given to hosts with higher bandwidth, since they can more frequently send their RTCP requests. This results in certain hosts being able to claim new resources faster as they become available.

The sender initiated property of YESSIR does not allow receivers from choosing their own QoS level, and does not allow channel switching or multiple reservation styles. The lack of error mechanisms also slows down any recoveries due to a route change.

2.1.4 The Ticket Signaling Protocol

The Ticket Signaling Protocol (TSP) [11] is a lightweight, simplex, sender initiated resource reservation protocol developed in 1998. Citing the scalability issues of connection oriented reservation protocols, TSP allows resources to be reserved without the need for connection states in the network, only requiring link states. This results in a highly scalable connectionless protocol.

All information required for resource reservations are stored at the end hosts, with it being delivered to intermediate nodes through the signaling protocol. There is no need to store any per-flow state in the routers, only to process it. It is currently at the prototype level.

Operation

To make a reservation, a request is sent containing a *traffic contract*; or also known as a flow descriptor. Included in this contract are the source, destination, priority, bandwidth, and timing specifics for the requested reservation. This request is then sent using existing routing protocols to the destination, with each node performing admission control along the way. If at any point admission control fails, the request is dropped. Upon the request reaching the receiver, an acknowledgement (containing the request) is returned to the *access router*; that is, the sender's edge router. The access router then creates a *ticket* containing the contract information, and is then sent to the source.

Reservations are made on a time slot basis, which is included in the original request. Tickets are then placed inside the data flow once per time slot, not requiring nodes to store any connection state, to confirm the reservation. To end a reservation, a release message can be sent, having nodes de-allocate any reserved resources.

Policing is conducted at the access router to insure proper use by adding digital signatures to insure tickets cannot be modified, and that old tickets are useless due to expiration dates.

If a ticket is lost, there is an ability to send the previous ticket as a *NACK* ticket, indicating that there was a problem receiving the expected ticket. If two tickets are lost in a row, the resources are released, and the reservation ends.

Advantages

The greatest advantage TSP has over other protocols is the lack of connection state needed in the nodes, leading to scalability and low complexity overall.

Disadvantages

Since no connection state is maintained in the routers, there is no multicasting support with this protocol. Also, additional mechanisms are required in the routers to prevent the immediate rerouting of traffic with reserved resources. Complex *switch-over* state tables are needed within the routers to prevent tickets from changing routes, causing over provisioning and other failures. Upon a route change, flows with resource reservations would fall into a best effort category, and would require another request to regain its reservation.

Access routers have the option to use connection state for security and policing, but do not require so. Either way, the access routers provide a single point of policy failure if they misbehave or are compromised. Bad tickets can also have dire consequences.

2.1.5 The Dynamic Reservation Protocol

The Dynamic Reservation Protocol (DRP) [12] is a sender-initiated, soft-state, simplex, reservation protocol designed for multicasting, which was developed in 1998. It has been modeled after RSVP providing several distinct differences.

Operation

To make a reservation, no prior setup is required. Reservations are created on-the-fly by sending reservation (RES) messages ahead of the data. After the RES message has been sent, data can follow immediately. If the resources cannot be reserved, the routers will reserve as much as possible; thus, allowing partial reservations. Included in the RES message is the sender's *ceiling reservation type* (CRT) which specifies the greatest QoS level it is willing to transmit.

Return (RTN) packets carrying path and feedback information from the receiver are sent back, with routers processing the data. Similar to RSVP's RESV message, RTN messages are also merged, allowing for greater scalability in large multicasting groups. Also included is a receiver ceiling reservation type (CRTr) which is similar to the CRT, specifying the greatest QoS level the receiver is willing to take. The sender can then address heterogeneity by using the minimum QoS level specified in the CRT and CRTr fields. A receiver can change its CRTr by sending a RTN packet containing the new value.

Advantages

Not requiring reservation setup allows applications to gain instant QoS, (given they pass admission control) along with the ability to modify flows instantaneously. This is also helpful for applications with on/off traffic by letting them free resources during times of inactivity, and immediately regaining their reservation when needed.

DRP also allows heterogeneity of receivers in the same multicast group, requiring little complexity on the receiver's part. The ability to merge RTN messages also allows for better multicasting scalability, especially in large groups.

Disadvantages

While DRP has several unique abilities that RSVP does not, generally speaking it has the same problems, such as overall scalability concerns and unattractive unicast delivery.

2.1.6 Boomerang

The complexity of RSVP has resulted in a poorly scalable protocol when deployed across large networks with numerous data flows. The aim of the Boomerang protocol [13] is to provide a much simpler solution, and consequently provide a more scalable one at the same time. Boomerang does not aim at replacing RSVP all together, only providing a much simpler alternative to a subset of potential uses; particularly the unicast flows. Multicast resource reservations are best left to RSVP.

Questions have been raised recently as to the soundness of multicast reservation systems due to their complexity, and more importantly that the needs for unicast reservations (such as VoIP) are becoming increasingly more popular. Boomerang, developed in 1999, is a non-simplex, soft state resource reservation system that is geared to these unicast reservations. It is the only non-simplex lightweight signaling protocol, which allows reservations to be made in both directions.

Tests conducted have shown that Linux-based Boomerang routers are able to handle upwards of 120,000 concurrent reservations, and up to 6800 requests per second without any noticeable impact on performance [13].

Operation

Since resource reservations using Boomerang are bi-directional, the end nodes are not labeled as sender or receiver. To make a reservation, the *initiating node* (IN) sends the resource reservation request through the network to the *far end node* (FEN). These requests include both the forward and reverse flow descriptor, and follow standard routing protocols. The resource reservation is done on a per-hop basis at Boomerang-aware nodes, with the caveat that all non-aware nodes are able to blindly pass the request through. This request is then bounced back from the FEN to the IN. As the request traverses the loop, each node examines the flow descriptor and compares it to available resources. If the resource reservation cannot be made, the flow descriptor is updated to the lower of these two, with the first node rejecting the reservation setting the NACK flag. While traversing, if the refresh interval requested is too high, the field is updated to the minimum acceptable.

Upon the request making a full loop, the IN checks the NACK flag to see if the reservation was successful, and the refresh interval to see if it needs changing. If the NACK flag is set, the request has been denied. The IN can then re-request with the same levels, or with the new specification attained.

After establishing a resource reservation, the IN is responsible for maintaining this reservation through periodic refreshes. Similar to RSVP, if not refreshed, the reservations are removed. This allows routing changes on the fly with only temporary effects.

Table 1. Signaling Protocol Comparisons

	RSVP	ST-II/+	YESSIR	TSP	DRP	Boomerang
Year	1993	1990/95	1998	1998	1998	1998
Reservation Initiation	Receiver	Sender	Sender	Sender	Sender	-
Reservation State	Soft	Hard	Soft	Hard	Soft	Soft
Direction	Simplex	Simplex	Simplex	Simplex	Simplex	Non-Simplex
Reservation Types	Hetero	Homo	Hetero	-	Hetero	Homo
Signaling Band	O-Band	O-Band	O-Band	O-Band	O-Band	O-Band

Advantages

Boomerang's current prototype uses ICMP echo messages to carry reservation requests. Deployment for such an approach is desirable since non-Boomerang nodes are still able to forward requests. Additionally, since the FEN is only required to bounce a request back, no alterations are needed except for the ability to respond to echo requests.

All complexity is performed within the IN, which is responsible for the creation and maintenance of the reservations. Consequently, Boomerang does not require any significant participation from the FEN, only requiring it to bounce the request back. As mentioned above, this can be done trivially with ICMP echoes. The far end host does not need to be modified.

Unlike RSVP, where reservation and path messages are separated, Boomerang uses one message resulting in short reservation setup times. When establishing bi-directional flows, it has been shown that in cases where available resources are in demand, Boomerang has a much lower blocking probability than RSVP [14]. Due to the use of a single message, the return path also need not be the same.

Unlike RSVP and ST-II, both senders and receivers can act as the IN and make the reservations. By using the sender as the IN, greater control can be kept over policy and billing issues, while allowing the receiver to act as the IN allows greater flexibility on the receiver's part.

Boomerang also has the ability for the looped messages, in case of failure, to act as a query returning with the minimum available resources.

Disadvantages

Boomerang only has support for one-to-many multicasting, and several drawbacks. These include the sender oriented design drawbacks discussed with the ST-II protocol earlier, such as the lack of custom reservations and channel switching.

The current use of ICMP echo messages is not necessarily meant as a permanent transport. While advantages are gained initially, a more traditional approach would be required eventually by either defining a new ICMP message, or a completely new protocol. Similar to other signaling protocols, this would require changes to infrastructure.

Unlike RSVP, rejected reservation requests are not immediately returned to the IN, and must follow the full loop before returning, creating added network traffic. The designers have opted to keep this issue in return for a simple protocol and no need for active nodes. There are currently no security mechanisms in place, and problems will arise by firewalls blocking ICMP messages. Generally speaking, there is a lack of functionality.

2.2 Flow Descriptor

While the signaling protocol can be considered a vessel, the flow descriptor can be considered the cargo carried within. The descriptor has been defined by the IntServ model, and is broken into two parts; the *flowspec* and the *filterspec.*

The flowspec describes the requested level of QoS, and provides other vital information needed for a resource reservation to be completed. The IntServ model has set out guidelines for what a possible flowspec format would look like. This includes both a service class and two other parameters: an Rspec and Tspec. An Rspec defines the desired QoS, while a Tspec describes the data flow, such as traffic flow and patterns.

The filterspec, coupled with a session specification, is used to define the subset of data packets which will receive the requested QoS found in the flowspec. The flowspecs could be defined as senders themselves (i.e. addresses and ports), protocols, or any fields contained within protocol headers.

2.3 Traffic Control

QoS provided for each data flow is done so through traffic control, which is comprised of three parts; admission control, a packet classifier, and a packet scheduler. Since the focus of this paper is primarily on the mechanisms, architectures, and protocols used to provide QoS, traffic control will only be discussed at an introductory level.

Admission Control

Admission control is used to determine whether or not a new resource reservation can be granted [15]. The admission control in each router is independent, and as such, no particular algorithm needs to be used in every router. Each node will apply this decision procedure to the request, and will return an error if admission fails. Upon a successful admission, the flow descriptor contained within the request is forwarded to the packet classifier and scheduler.

Packet Classifier

Using filterspecs, the classifier maps all packets into some class, with each of these classes receiving the same QoS level. The class of a packet is typically determined through either a classification number, or from the data contained within the headers. The use of classification numbers typically takes the approach in replace IP with a virtual circuit, and using circuit identifiers. This is the approach taken with ATM, and protocols such as ST-II [4]. Along somewhat similar lines, MPLS (which will be discussed in section four) encapsulates messages with a label, and uses these for classification. When using a connectionless approach, data such as the source address, protocol numbers, port numbers, or even application layer information can be used to classify. Consequently, classifier implementations are very complex since there is much processing required.

Packet Scheduler

When incoming data packets reach a node capable of resource reservation, the class (if any) is determined by the classifier, and then sent to the packet scheduler to be queued appropriately [16]. Using queues and timers, the packet scheduler controls the forwarding of data streams. A more generic description would be that its function is to reorder the output queue. Packets which do not fit into any special service class are automatically handled as best-effort. Several different techniques have been developed such as priority queues, round-robin variants, or Weighted Fair Queuing which can splice bandwidth into specific shares. The dropping of packets is also an important aspect, since careful consideration must be taken. The Integrated Services model has proposed a preemptable packet dropping service, where hosts can willfully mark their packets as droppable if delay bounds can not be met. Typically this includes delay sensitive data, and can potentially help alleviate congestion.

2.4 Service Classes

The IntServ model has prepared several services classes to meet the needs of applications. Each will be discussed briefly.

Guaranteed Service

Guaranteed Service [17] is one in which provides solid bounds to delay and bandwidth on a network flow. Guaranteed Service flows can expect all packets to be transported and delivered within the predetermined bounds with no loss, given the path does not change.

Packet delay over the Internet is comprised of two parts. The first factor is the fixed delay which primarily comes from transmission delays, and as a result is uncontrollable. Fixed delays are related to the chosen path, which is a result of the setup mechanism, not the Guaranteed Service. The second factor is queuing delay which can be controlled by Guaranteed Service. Guaranteed Service is not concerned with the median or minimal delay of packets, only the maximum delay.

Predictive Service

Predictive Service [18] (also known as Controlled Load Service) is one in which provides low loss and a fairly reliable probabilistic delay bound. Flows that use this service can expect that the majority of packets to be transported will be delivered within the requested delay bound, and reserved flow rates will be mostly honoured. This service is primarily used for applications which require an upper bound on delay for performance, but can still function properly with the odd late or lost packet.

Increased use of Predictive Service will permit more reserved flows since the relaxed commitments allow higher utilization of network resources. Since delay bounds will be broken infrequently, there is no attempt in providing jitter control. These services are provided assuming no failures in the network infrastructure or routing changes.

Similar to Guaranteed Services, Predictive Services must use admission control and deny any flows which will cause any hindrance to the current flows.

Controlled Delay Service

Controlled Delay Service [19] is comparable to Predictive Service, with the primary difference being that Controlled Delay does not offer any bounds on delay. Similar to Predictive Services, it does not provide any jitter guarantees. This service is primarily designed for applications that are delay sensitive, yet are still able to adapt to delay levels through other application specific means, or are willing to upgrade to a higher level service.

Similar to Guaranteed Services and Predictive Services, Controlled Delay Services must use admission control.

Best Effort Service

Best effort service encompasses all other traffic which does not belong to the above service classes. No admission control is required. There are no bandwidth, delay, or jitter guarantees.

2.5 IntServ Advantages

While models such as IntServ would require massive restructuring of Internet infrastructure for widespread use, it can still provide benefits when partially deployed, particularly in intranets or ISP backbones. Stateful solutions are able to provide better assurance levels, and flexibility. IntServ can provide per-flow guarantees with firm bounds on bandwidth and delay.

2.6 IntServ Disadvantages

The amount of state required, particularly in core routers, increases with the number of flows. Since router performance is linked with its ability to maintain these flows, and the Internet is still growing at a phenomenal rate, there are serious scalability concerns with the IntServ model. With the advent of lightweight signaling protocols, scaling concerns have been partially alleviated, but unfortunately are still unable to address large multicasting groups effectively. Regardless of signaling protocols, the router requirements are extremely high due to the complex nature of the IntServ model.

With few predefined services classes, IntServ is not flexible as some would like. It would also require ubiquitous deployment to reach home users. There is also a lack of policy control mechanisms.

3. Differentiated Services

Differentiated Services (DiffServ) [20] is a model which provides QoS through a relative priority scheme, with network devices handling traffic at aggregate levels rather than the IntServ approach of handling individual flows. Most complexity has been pushed out to the edge routers, with core routers simply forwarding and scheduling the already classified data. Traffic is classified into behaviour aggregates (BA) as it enters the network, with routers treating each aggregate in a unique manner. No connection setup is needed.

Classifying aggregates is not done at the end host, but is rather done through the use of Service Level Agreements (SLA). They are used to provide differentiated services between a user and a provider, and contain the rules for packet classification and conditioning. The following sections will discuss some key aspects of the DiffServ architecture.

3.1 Per-Hop Behaviours

Aggregates are grouped into per-hop behaviours (PHBs), which are marked in the DiffServ code point (DSCP). The DSCP is located in the first six bits of the IP Type of Service field. There have been several PHB’s proposed, but two have gained much attention; Expedited Forwarding, and Assured Forwarding.

Expedited Forwarding PHB

Expedited Forwarding (EF) [21], also known as premium service, is primarily for applications which produce fixed rate traffic, requiring an assured (but not guaranteed) bound on delay and jitter.

EF does not deal with individual user flows, but rather the aggregates of them. Therefore, no bounds are placed on individual flows. The aggregate receives its predetermined rate regardless of any other traffic on the node, with any EF traffic exceeding the set rate being discarded.

To achieve this service, the departure rate of the aggregate on each outgoing link must be greater than or equal to the sum of maximum arrival rates on incoming links. In the time it takes to send a maximum MTU packet at each outgoing link, the overall outgoing service rate should average, or exceed, the predetermined rate. There are a number of queuing methods which can be applied, including simple priority queues (PQ), weighted round robin queue scheduling (WRR), and class based queues (CBQ).

An SLA is used when classifying packets into the aggregate, and is typically meant for long term provisioning, not on-demand connections. The aggregate is expected to use only a small share of bandwidth, and operates as a *Virtual Leased Line* (VLL). When not using its predetermined rate, excess bandwidth is used by other PHBs. Admission control is frequently conducted offline, with less emphasis, if any, put on signaling protocols.

Assured Forwarding PHB

Assured Forwarding (AF) [22] is primarily for applications which require reliability better than best-effort service. From the SLA, profiles for aggregates are derived with predefined rates. Each aggregate is given a high probability of timely delivery as long as it does not exceed the predetermined rate. Packets which conform to this rate are called *in-profile*. It is also possible to send at a rate beyond the defined rate in the profile; these packets are called *out-of-profile*, and it is understood that this traffic will not be given as high priority as in-profile packets would. Regardless, neither type of packet will be reordered.

There are four Assured Forwarding classes defined, with each class receiving differing amounts of resources. Each class is further marked with three drop precedence values, where in the case of congestion, these values determine which packet will be dropped based on significance. In-profile packets will typically be marked at low drop precedence, while others will be marked at the other two levels.

QoS for each AF class is determined by the resources allocated for the class, current load of the class, and drop precedence of each packet. To achieve this service, assured queues (AQ) are used and managed through Random Early Discard with In and Out (RIO).

3.2 Differentiated Services Components

From the SLA, a Traffic Conditioning Agreement (TCA) is derived. To adhere to this agreement, both classifiers and conditioners are needed. BA classifiers are used to sort packets into their PHB class using the DSCP, while Multi-Field (MF) classifiers can also use any other header information including interface information.

Conditioners are the control functions applied to each of these classes to make them perform appropriately. Conditioners are typically comprised of metering, policing, shaping, and packet marking functions.

Metering is used to measure the temporal attributes of classes, compare them to the TCA, and determine whether packets are in or out of profile.

Markers are used to set the DSCP and add the packet to an aggregate, based on the TCA. The meter may affect which aggregate it is added to, through remarking, depending on the packet profile it derived.

Shapers are used to smooth traffic to a configured rate based on the traffic profile. Packets will be dropped if the finite buffer is overrun.

Policing is done to restrict classes of traffic to certain rates so out-of-profile packets can be remarked or dropped.

3.3 DiffServ Proposals

Generally speaking, there have been two fields of thought in deploying DiffServ. Performing admission control and policing at the edges only, requiring no control in the core network, is the easiest. Such approaches [23, 24] require no state in the core network, but require substantial bandwidth in the core to prevent unfair degradation of individual flows from the aggregated class.

Other proposals [25] suggest the placement of moderate controls within the core to provide proportional fair sharing and flow protection. Unfortunately, scalability concerns arise due to the added complexity and state.

There have been several different pricing and increased functionality proposals put forward. By far the simplest pricing scheme proposed is Paris Metro Pricing [26], which proposes to separate the network into a number of equal logical channels. Each channel is assigned a different price, assuming that higher priced channels will naturally be less congested than others.

The Proportional Differentiation Model [27] is a way to improve DiffServ by allowing operators to fine tune quality spacing between aggregate classes, independent of class loads. Thus, the quality of each class differs under load, but the quality ratio between classes remains static. Alternatively, research [28] in providing relative proportional DiffServ uses feedback from a metering component to dynamically adjust traffic conditioning.

3.4 DiffServ Advantages

The differentiated services model is intended to address several problems that have plagued the Integrated Services model, including scalability and complexity. By not requiring per-flow state to be stored in the routers, and no complex signaling protocols, DiffServ is highly scalable and relatively less complex.

DiffServ also aims to provide qualitative and flexible service classes, where classes can be relative to one another; i.e. Platinum, Gold, Silver, and Bronze. In some cases, this is preferred over quantitative approaches.

3.5 DiffServ Disadvantages

Since DiffServ treats packets in the same class identically, it is difficult to provide quantitative QoS to individual flows. It is strong on simplicity, but weak on guarantees. It is primarily designed for and used by ISPs, and is not too useful (or even intended) for end users. Additionally, network management techniques such as bandwidth brokers must be in place to provide resource control.

While the aggregation of smaller flows suits the model well, other flows such as elongated or bandwidth intensive flows often require per-flow guarantees. If routes change, existing guarantees can change, leading to a degradation of service.

While SLAs can be dynamic, they are typically designed to be long term, yet both network traffic and topology are dynamic in nature. Lastly, DiffServ does not offer any receiver control.

4. Multiprotocol Label Switching

The original purpose of Multiprotocol Label Switching (MPLS) [29] was to provide high speed Layer 2 switching at Layer 3 through the creation of switched paths which use labels for routing decisions, rather than having to use complex route lookup mechanisms.

Since the advancement of high speed Layer 3 switching technology, the performance gain has since lost standing for the motivation of MPLS. Due to the connection oriented design, Traffic Engineering (TE) is possible, and has since become the most important motivation behind MPLS deployment. Another important motivation is that paths have the ability to cross many different Layer 2 transports, such as Ethernet, ATM, and Frame Relay, without the need for any other mechanisms present. TE enables many other abilities in the network, including the capability to provide various routing procedures for load balancing and congestion avoidance. It is also possible to assure differing levels of service to each path, or to create virtual tunnels for VPNs.

When packets enter an MPLS domain, a *label edge router* (LER) assigns each a short fixed-length label. The packets are then forwarded through a series of *label switched routers* (LSR), the entire path being referred to as a *label switched path* (LSP). The group of packets that use the same LSP and receive equal forwarding treatment is referred to as a Forward Equivalency Class (FEC). Data packets entering a LSR are forwarded based on their label. Once a label has been used to make a routing decision, it is then replaced with another label to be used at the next hop.

Labels are comprised of four fields, totaling 32 bits, and are inserted between the Layer 2 and Layer 3 header. The 20-bit label field contains the actual MPLS label value, while the 3-bit Class of Service (CoS) field is used for providing differing levels of service. A single bit Stack field is used to support a hierarchical label stack, and an 8-bit Time to Live (TTL) field provides conventional TTL functionality.

4.1 Label Distribution

A labeled switched path is created through the use of a label distribution protocol, which establishes paths through an MPLS network by distributing the appropriate labels, and reserving appropriate resources if requested. They are also required to provide a mechanism for the discovery of other LSRs.

To setup LSPs using existing IP routing information, the Label Distribution Protocol (LDP) can be used. There are, however, more complex protocols required to bypass existing routing protocols, or to provide resource reservations. There are two proposed LDPs for these which incorporate traffic engineering and reservation abilities; CR-LDP and RSVP-TE.

Constraint-based Routing over Label Distribution Protocol

CR-LDP was created through the modification of the Label Distribution Protocol (LDP), by adding traffic engineering capabilities. CR-LDP is a hard state protocol, requiring no periodic refreshes, and is transported by TCP sessions between LSRs. Reservations are requested by the sender.

To generate a new LSP, a LABEL_REQUEST is sent from the ingress to egress LER through either traditional routing protocols, explicitly stating the path, or a partial path. Included in the request is an optional flow descriptor if a reservation is required. At the ingress LER, and each intermediate LSR, the resource reservation is made before forwarding the request. After the resource reservation is made at the egress LER, a LABEL_MAPPING message is sent back towards the ingress LER containing a new LSP label and information regarding the reservation just made. At intermediate LSRs, any pending reservation is finalized, a new LSP label is created, and the forwarding table is updated for the new LSP.

Resource Reservation Protocol with Tunneling Extensions

RSVP-TE is an extension to RSVP that includes mechanisms for MPLS traffic engineering. Since it runs over a raw IP transport, it has mechanisms present to account for message loss. Additionally, its soft state nature requires periodic refreshes to keep reservations from being removed. Similar to RSVP, reservations requests are receiver oriented.

To make a new LSP, a PATH message is sent from the ingress to egress LER similar to that of CR-LDP. The PATH message is traversed through all intermediate nodes using existing routing protocols, explicitly specified paths, or partial paths. Upon reaching the egress LER, a RESV message is formulated containing a flow descriptor describing the requested reservation. After making its own reservation, a new LSP label is attached to the RESV message and returned back through the reverse path. Intermediate LSRs will attempt to make any reservations, update the forwarding table with the label received, and attach a new LSP label. The ingress LER will do the same, without the need to attach a new label.

Table 2 provides a comparative look at CR-LDP and RSVP-TE.

Table 2. CR-LDP and RSVP-TE Comparison

	CR-LDP	**RSVP-TE**
Transport Mechanism	TCP	Raw IP
State Management	Hard	Soft
LSP Refresh	No	Yes
Resource Request	Sender	Receiver
Strict Routing	Yes	Yes
Loose Routing	Yes	Yes
Shared Reservations	No	Yes

4.2 QoS

One of the largest misconceptions is that MPLS is a QoS technology in itself. Rather, it introduces a networking environment that is capable of transporting different traffic over a common infrastructure, while being able to enable QoS effectively. Thus, it is a QoS-enabling technology, and provides a flexible solution for QoS deployment and management [30].

MPLS with DiffServ

The abilities of MPLS to force packets to specific paths and to guarantee bandwidth to forward equivalency classes, combined with the ability of DiffServ to specify differentiated treatment of aggregates, results in QoS.

To support this, two types of LSPs are defined. E-LSP use labels as FEC destinations, and the CoS field to carry the class of the flow. By preserving labels, and using the CoS field for DiffServ, E-LSPs are easier to manage and far more scalable. Alternatively, they do not carry scheduling information, so there is the possibility bandwidth will be lacking in the queue it is placed in.

L-LSP uses labels as both the FEC destination and scheduling priority, with the CoS field used for drop priority. They are harder to manage, but there is no concern over bandwidth since scheduling information is included.

The merging of MPLS and DiffServ is more scalable than IntServ alone since routers require no per-flow state, only aggregate information. While LSPs can be dedicated to one flow, many flows can also aggregate into one, requiring less signaling.

Thus, MPLS technology forces application flows into connection-oriented paths, providing bandwidth guarantees to the flows. The addition of DiffServ provides additional service to these flows, including class based admission, differentiated queue servicing, preemption priority. Depending on which type of LSP is being used, scheduling information can be included to provide firmer QoS bounds.

MPLS Extensions

There has been numerous extension proposals intended to add further functionality to MPLS. While not directly affecting any QoS mechanisms, the increased functionality makes it a more attractive overall solution. Some proposed extensions have included modifying the RSVP-TE and LDP signaling protocols to support enhanced multicasting functionality [31] and fast reroute abilities [32], as well as modifying ICMP to allow LSRs to append MPLS information for greater messaging flexibility [33].

5. Other Architectures

Since neither IntServ nor DiffServ have gained far spread acceptance due to scalability and inflexibility concerns respectively, several new architectural models have been proposed which promise to remove these hindrances causing them from being deployed. The two most common approaches to accomplish this are to make both IntServ and DiffServ interoperable, or simplifying the signaling protocol. Some of these approaches will be discussed, with an architectural comparison table (Table 3) found at the end of the section.

5.1 Scalable Resource Reservation Protocol

The Scalable Resource Reservation Protocol (SRP) [34], introduced in 1998, is one which aggregates flows on links in the network, without the need for a signaling protocol. It requires no per-flow state in the routers, only at the network edge. The only additional overhead required in this architecture is the addition of two bits in each packet. These represent the values *reserved, requested*, and *best effort.*

SRP provides a service similar to IntServ's Controlled Load service, and cannot provide any bounds on delay or jitter, only bandwidth.

Operation

To make a resource reservation, an application does not require any previous signaling. Instead, the application starts sending data packets to the receiver with a request flag set. These packets are subjected to admission control in each intermediate node, with those accepted being forwarded with no changes. Rejected packets will have their flags changed to best-effort before being forwarded. Routers which have accepted these request packets will then reserve appropriate resources, simply through the additive properties of the aggregate reservation.

The receiver, after a short period of time, can then estimate the rate of the reservation which has been accepted. This value is then returned to the sender through the use of a feedback protocol. Based on the feedback, the sender can then transmit packets at the reserved rate marked with a reserved flag. These marked packets are treated as part of the reserved aggregate in the routers, receiving the equivalent of a Controlled Load service. The sender can maintain its reservation as long as there is some level of activity, with the reservation being removed after a certain period of inactivity. All other traffic is by default marked as best-effort.

Estimator

To maintain reservations, an estimator is needed in several areas to calculate the aggregate reservation required. This is done by measuring the number of packets currently marked as requested or reserved. Estimators are used by senders, receivers, and intermediate routers.

Senders can make *optimistic* predictions as to the reservation a network will allow. By default it assumes all requests are accepted. Routers use estimators for admission control, and receivers can generate *conservative* predictions for feedback. Sources use both the optimistic and conservative estimations to generate their output rate.

The specific implementation of estimators has been left independent, as several algorithms exist, and are still the focus of continuing work.

Advantages

SRP, while being highly scalable from lack of per-flow state, can be transparently tunneled through non-aware routers. If these routers are not susceptible to congestion, no service degradation will occur.

In addition, the protocol processing required is overly simplified by attaching necessary control information to data packets themselves instead of using a separate signaling protocol.

SRP also allows extensions to provide limited multicasting abilities.

Disadvantages
To implement SRP, routers must trust end hosts with traffic control decisions, such as not to exceed their allocated reservation. To circumvent this, complex mechanisms must be present to meter and police at the network edges. It is also suggested to handle reservations in the kernel to keep applications from misbehaving.

As with other protocols that allow partial reservations, there is the potential for resource starvation when multiple sources attempt to make reservations simultaneously. The negative feedback properties of this can potentially lead to devastating results.

Route changes in the network can also cause serious issues, since it can lead to overloading of reserved packets on the rerouted link, leading to degradation of service. *Route pinning*, which is the ability to force flows to follow a certain path, has been suggested, but still cannot account for router and link failures.

5.2 Flow Initiation and Reservation Tree Protocol

The Flow Initiation and ReServation Tree (FIRST) [35] protocol, introduced in 1999, is similar to RSVP in that it is geared towards multicasting and is receiver oriented. Its aim is to provide all the advantages of RSVP, ST-II+, and YESSIR. It does however require that routes do not change and keeps hard state in the routers, requiring no refresh messages. A reservation setup is similar to RSVP in that PATH and RESV messages are sent. Similarly, it can allow heterogeneous receivers who can request differing reservations.

Similar to SRP discussed above, FIRST reserves resources based on aggregates. It is assumed that intermediate nodes receiving reservation requests will be able to decide if the flow can be admitted or not using the same methods as SRP, namely estimators.

Each router contains flow and routing tables. Flow tables are used to maintain flow information, contain source and destination pairs, forward and reverse data paths, and the service level. The forward and reverse data is derived from reservation messages, and is used to pin the routes. A flow session is represented by the source and destination pair. Forward and reverse path values cannot change during the flow session.

Routing tables contain the same fields as flow tables, with the exception that the service level is always best effort. This routing table maintains normal routing data, and updates accordingly as routes change. This table is meant for best effort traffic which does not require any route pinning.

A termination message is required to teardown a reservation. Work on the protocol has ceased, with no analysis having been conducted.

Advantages
Since FIRST aggregates flows into several different service levels, a scalable architecture is the result. Being receiver oriented, it is also able to handle heterogeneous receivers and aggregate flows as it traverses up multicasting trees. It also can handle large multicasting groups in a scalable manner by merging reservation requests.

Disadvantages
Since work on the protocol has stopped, there has been little quantitative test results. The hard-state attributes result in poor robustness in the case of failure. Documentation is scarce, and there are many areas which need further explanation.

5.3 IntServ over DiffServ Networks

Seeing IntServ and DiffServ as complementary models, IntServ over DiffServ [36], introduced in 2000, attempts to overcome the limitations in both models by placing IntServ at the edge of the network, and DiffServ in the core. This requires no per-flow state to be stored in core routers, with only the edge routers being responsible for the interface between the domains. To IntServ capable nodes, DiffServ domains are seen as “virtual links” which connect them. Since no state is required in core routers, this approach is highly scalable.

While other signaling protocols may be used over the IntServ domain, current work being done uses RSVP. To create a per-flow guarantee, an RSVP PATH message is sent from the host. Upon reaching the ingress edge router, the message is packaged and shipped transparently across the DiffServ domain, being unpacked at the egress edge router. From there, the PATH message is delivered to the intended recipient, with a RESV message being returned. As with the PATH message, it is transported transparently across the DiffServ domain. Upon the egress edge router receiving a successful RESV message, it performs admission control to the DiffServ domain. If this is successful, the RESV message is allowed to continue to the receiver. If not, a RSVP error message is sent.

An alternative exists to the above example by allowing nodes within the DiffServ domain to be IntServ aware. That is, additional per-flow information can be used during DiffServ admission control, and during packet scheduling.

Service Mapping

Services provided by IntServ must be mapped into the DiffServ domain to keep end-to-end QoS from being broken. This is done by selecting one or more appropriate PHBs depending on the resource request, and by performing policing at the edge. There are two types of mapping. *Default mapping* encompasses all well-known IntServ to DSCP mappings. In *network driven* mapping, RSVP capable routers in the DiffServ domain can remark the DSCPs.

Resource Management in DiffServ Domains

There are several ways to provision resources in the DiffServ domains. The simplest, but least flexible, is statically allocating resources based on SLAs, or the use of bandwidth brokers. The most flexible option is to dynamically adjust them through the use of a signaling protocol such as RSVP.

Advantages

The use of DiffServ in core routers allows greater scalability, while the use of IntServ at the edges allows better flexibility, particularly with per-flow control. It is possible to allocate resources to certain applications, rather than large aggregates.

Disadvantages

One of the largest challenges to this architecture is resource management. When statically allocating resources within the DiffServ domain, or using other long term approaches such as bandwidth brokers, it is difficult to notify edge routers of the traffic load within the DiffServ domain. Using a dynamic approach such as RSVP signaling inside the DiffServ domain can adequately address this issue, but reverts to an unscalable model with the need for per-flow state in core routers. As well, the need for the service mapping ability in edge routers results in further added complexity.

Another large challenge is deployment. With neither IntServ nor DiffServ having any large scale deployment as it stands, IntServ over DiffServ would suffer just as much, if not more, difficulty.

Since DiffServ networks cannot provide firm bounds on delay, this architecture cannot deliver the IntServ Guaranteed Services class; only predictive and controlled delay classes, along with DiffServ PHBs.

5.4 Edge-assisted Quality of Service

The Edge-assisted Quality of Service (EQOS) [37] architecture, developed in 2000, requires only edge routers of a domain to be modified, with core routers requiring no changes. It is interoperable with both DiffServ and IntServ networks, and is comprised of a signaling protocol, a distributed admission control mechanism, and route pinning mechanism.

EQOS allows two types of flows; reserved and best effort. Reserved flows can provide bandwidth guarantees, but are unable to provide bounds on either delay or jitter.

Reservations are made through both a signaling protocol, and a token passing mechanism that maintains reservation consistency around the edge routers.

Signaling Protocol

EQOS uses RSVP for signaling, but does not use it in the traditional manner. PATH messages are transparently sent through the domain with only edge routers stamping their addresses and updating per-flow path states. RESV messages are also transparently transported back over the EQOS domain through the use of a route pinning mechanism, with only the two edge routers examining and making the reservation. Thus, no core routers are required to be RSVP-aware.

EQOS also takes advantage of the soft-state properties of RSVP to make sure flows which do not end properly are removed.

Distributed Admission Control

The EQOS architecture uses a distributed system which assumes the admission control responsibility of the entire domain. This is done at the flow's ingress edge router. To perform an admission decision for the entire domain, a single control token is circulated among the edge routers, which contains bandwidth information for all the links in the domain. Using information contained within this control packet, and network topology information, a reservation route is selected. While numerous algorithms can be used, the "shortest wide-enough" path is the most popular, which selects the shortest reservable route. Tokens are circulated dynamically by the shortest path, with links preempting the processing of other packets when a control token arrives to minimize circulation time.

Upon a successful reservation the route is pinned, and information contained within the control token is updated and forwarded to the next edge router. The reservation is not final until this control token has made one full circulation. This is done for two reasons; to keep edge routers from over provisioning a link, and to allow edge routers to potentially reduce best effort rates using the same links.

Information regarding best effort flows must be maintained as well, since there is no way within the domain to distinguish them from the reserved flows. Thus, the rate of the best effort flows must be limited at the edge routers using a system of fair queuing schedulers.

Route pinning can be done through either MPLS, or IP source routing. Since MPLS requires all routers in the domain to be MPLS-aware, IP source routing would be the most popular approach. Transparently, ingress edge routers insert the routing information with egress edge routers removing it.

To combat lost tokens, it has been suggested to have one router as a token monitor which keeps a copy of the token, releasing it only when the real token does not return after a timeout period. This replacement will be noticed by other edge routers, who will then reissue reservations or update appropriate flow counts.

If a link or router within the domain fails, the token is updated from an edge router who will receive the new link state through updates. After one full circulation, all edge routers will have the new topology information.

Advantages

The largest advantage to EQOS is the ease of deployment and low cost, due to the core routers not requiring any modifications. This architecture provides excellent scalability as well.

Disadvantages

Unfortunately, EQOS cannot provide true Guaranteed Services to reserved traffic since there is no means to provide bounds on delay or jitter. Thus, it would act similarly to the Controlled Load class of the IntServ model.

Additionally, both reserved and best effort flows require route pinning, which leads to lack of robustness and added overhead. While there are mechanisms in place to account for link and router failures, the temporary disorder can cause overloading of links resulting in packet loss.

Since a fair queuing scheduler is required for every edge to edge route, if the number of active reservations between two edge routers is large, there will be large computational demand on the router.

5.5 Endpoint Admission Control

Since attempts at deploying real time services have been hindered in the past, primarily due to scalability concerns and the massive restructuring and standardization needed, Endpoint Admission Control (EAC) [38] examines whether or not it is possible to provide such services with little support from core routers.

With this infrastructure, end hosts make their own admission control decisions by sending probe packets at the rate to be reserved into the network used to report back bandwidth, delay, and loss results. Since no participation is required from the network, this architecture can be run over DiffServ networks. A signaling protocol such as RSVP is not required. Based on the probe results, the decision to admit the flow is made. Since the results are not capable of providing exact numbers, the Guaranteed Services class can not be duplicated. Endpoint Admission Control aims at providing Controlled Load services, where strict guarantees are not made.

Packet loss is typically the measurement from which an admission decision is made. To prevent starvation while probing a network, probing must be done incrementally. During a given time period, if the packet loss exceeds the point where the total packet loss goes over the acceptable threshold, the probing is stopped.

Advantages

The lack of participation from the core routers makes this solution scalable, and cheaper to deploy.

Disadvantages

Besides the fairly high bandwidth wastage of probing, which takes anywhere between two to five seconds, it also causes a substantial delay which is not appropriate for most real time applications. It is also assumed that end users will cooperate, sending only if the resources are available. Since admission control is done at the end hosts, this is very difficult to police, leading to lack of verifiability.

Changes in routes will also cause congestion, as there is no mechanism to provide route pinning.

5.6 Stateless Core

To provide the flexibility of IntServ while achieving high scalability, the Stateless Core architecture (SCORE) [39] was introduced in 2000. It combines both service differentiation from DiffServ and guaranteed services from IntServ into one scalable package.

Since scalability concerns arise from flow state being stored in the network, SCORE remedies this through the removal of state from core routers; thus, only requiring it in the edge routers. Information required by routers can be found within the packets themselves. A specific signaling protocol has not suggested, as has been left independent of the SCORE architecture.

SCORE aims to push complexity out to the network edge, resulting in the need for edge routers to perform special operations. To operate, a transparent SCORE domain is needed.

To achieve a stateless core, both the stateful data path and stateful control path need to be redesigned. Along the data path, a mechanism must be present to provide bounds on delay and jitter without the need for state. Along the control path, a mechanism must also be present to allow admission control without the need for state.

Dynamic Packet State

Dynamic Packet State (DPS) is the fundamental technique used to implement the SCORE architecture. By using DPS, flow state is carried in the packet, rather than stored in the router. DPS is inserted into each packet at ingress routers and removed at egress routers, with intermediate nodes updating appropriate fields. This results in a transparent operation. Since edge routers handle less traffic and typically operate at lower speeds, scalability is the result.

Core-Jitter Virtual Clock

To provide delay bounds in the data path without the need for per-flow state, an extension to the Jitter Virtual Clock (JVC) is introduced, called Core-Jitter Virtual Clock (CJVC).

In a stateful network, JVC guarantees that no packet will miss its deadline. In it, each packet is assigned an *eligible time* and *deadline*. Packets are not released until it becomes eligible, with packets being sent out in order of their deadlines. To assign the eligible time of a packet, the maximum value of the following properties is taken:

- The arrival time of the packet
- The packets deadline at the previous node plus propagation delay
- The previous packets deadline at the current node

The deadline is computed as:

- Eligible Time + Packet Length / Reserved Rate

When assigning an eligible time, having to look at the deadline of the previous per-flow packet requires state. Thus, the concept of CJVC has been proposed to eliminate this dependence. A slack variable is computed at the ingress router based on the lengths of current and previous packets, the slack variable of the previous packet, and the number of hops. The variable is then sent with the packets, and is computed such that the eligible times and deadlines at the last router are the same as they would be if JVC were being used. Thus, CJVC can provide the same bounds as JVC can without the need for state in intermediate routers.

Collectively, three variables are carried with each packet using the DPS approach mentioned above; the slack variable, the reserved rate for the flow, and an *ahead of schedule* variable which identifies how far ahead of the deadline the packet was sent. Therefore, the new eligible times and deadlines are computed without the need for state as follows:

- Eligible Time = Arrival Time + Ahead of Schedule + Slack Variable
- Deadline = Eligible Time + Packet Length / Reserved Rate

As such, state is been removed from the data path and placed in the packets, requiring no per-flow storage in the routers.

Admission Control

Current admission control techniques are broken into two classes; distributed and centralized. Distributed approaches through protocols such as RSVP can provide short lived per-flow reservations, but lack scalability. Centralized approaches using bandwidth brokers are simple to implement, but are not appropriate for per-flow reservations. The SCORE architecture uses a distributed approach, but removes all per-flow state, thus making it scalable.

Admission control measures requested reservation rates against current aggregate rates to make a decision. Due to packet loss, partial reservation failures, and under utilization of reservations, maintaining a proper reservation aggregate for each outgoing link without per-flow state is very difficult. In the absence of these issues, the reservation aggregate could simply be measured by current outgoing rates.

Since under utilization of reservations happens frequently, a *virtual length* is assigned to each packet. This value is set so that if the packet length were equal to its virtual length, the flow would send at its reserved rate. From this the value of the unused reservation rate (since the last packet) can be made. These values are calculated and inserted by the ingress router, and used by core routers to estimate reservation aggregates. These estimations are then used for admission control.

Flow Protection

In the Internet, there is a reliance on self imposed congestion control; the most popular example being the Transmission Control Protocol (TCP). Since this relies on participation from the end hosts, there is no way to trust everyone to impose this type of control.

Flow protection is used to shelter well-behaved traffic from ill-behaved, by using fair bandwidth allocation in the routers. Typically, fair bandwidth allocation requires state to perform per-flow management. The SCORE architecture uses DPS to achieve this, giving it the name Core-Stateless Fair Queuing (CSFQ). Edge routers estimate the incoming flow rate and attach this value the packet using DPS. When received by core routers, the probability of the packet being dropped is calculated as a function of the rate carried within the packet, and the fair share at the router. If packet dropping occurs, the rate is updated to reflect this. Thus, no per-flow state is needed in the core routers.

Advantages

The primary advantage of SCORE is the stateless properties which make it scalable. Consequently, this statelessness also results in better robustness in the case of physical link failures, since there is no replicated or inconsistent state to deal with. SCORE is also able to provide both Guaranteed Services from IntServ and service differentiation from DiffServ.

DPS packets can be used to find misbehaving elements in the network, by using a “verify and protect” approach, in which routers statistically verify packet state.

RSVP and other stateful solutions allow route pinning, which is typically done for traffic engineering, and is often required to provide Guaranteed Services. To provide this in a stateless network, SCORE labels a path using router identification numbers by simply XOR’ing them together. The ingress router keeps these labels, and attaches them through DPS as packets leave. At each intermediate node, the label is updated by XOR’ing it with its own identification number. The new label is then used to forward the packet to the appropriate router.

Disadvantages

While SCORE can provide IntServ Guaranteed Services, limitations presented by a stateless solution do not allow predictive load and controlled delay services.

SCORE does not handle partial reservation failures well, and in a worse case scenario, it can affect an entire domain. The verify-and-protect approach has been suggested, but comes at the cost of added complexity. Even without this, packet processing in a SCORE domain is very complex (especially in the case of Guaranteed Services) and results in a great deal of processing overhead being required.

By pushing complexity to the network edge where there are typically less flows and lower speeds, the architecture is scalable. Since routers must be aware of the architecture, it can not be deployed incrementally, only on a domain by domain basis. If domain edges touch core routers during deployment, scalability is lost for flows traversing through that edge.

Since state has effectively been moved from being stored in the routers to being stored within packets themselves, there would naturally be additional processing overhead in the routers. It has not been addressed whether scalability concerns could potentially arise under high router load.

Little work has been accomplished in providing multicasting support.

5.7 Aggregate RSVP

More currently in 2001, RSVP extensions [40] have been developed to allow a hierarchical reservation scheme to combat scalability concerns, which are due to the lack of ability to aggregate small flows. This extension would allow many smaller flows to be aggregated in a few larger flows similar to a virtual pipe, reducing per-flow state in the routers and signaling overhead. Classification of these larger flows would be done through DiffServ.

An aggregation region is defined, with end-to-end flows that cross into this region being aggregated and deaggregated as they enter and leave respectively. This region must have a contiguous set of RSVP-aware routers that can perform aggregation and deaggregation along all possible routes between. End-to-end reservation messages are hidden from the aggregation region, to prevent wasted resources. This is done by having routers at the border of the domain change the IP protocol number of certain RSVP messages to a special ignore case, and restoring it upon leaving the domain.

The flow descriptors of many individual flows are summed at the aggregator, and used to generate an aggregate reservation request to the corresponding deaggregator on the other side of the domain. With this approach, the number of RSVP reservations within a network will significantly decrease, but it all depends on how many aggregation regions are defined, and their sizes.

Advantages
The ability to make guaranteed reservations for large aggregates in the core network results in the reduction of reservation state. Routers in the aggregation region only need to keep reservation state for the aggregates. By offering few QoS classes, packet scheduling is kept simple.

The virtual pipes in an aggregated region are dynamic. They can grow and shrink depending on the particular demand, and cease to exist when there are no flows. This leads to greater scalability.

Disadvantages
When classifying packets into large aggregates, there is still a need for fine granularity. Aggregators must dig into the headers to extract specific source and destination values, which are then compared to a list of reservations. Additionally, since the virtual pipes are not end-to-end, heterogeneity cannot be supported in the aggregating region.

5.8 Simplified Guaranteed Service

Introduced in 2003 [41] and using ideas from FIRST [35], this design provides a simple QoS architecture that can dynamically provide IntServ classes in a scalable fashion, with one exception; the Guaranteed Services class cannot provide any delay bounds, only rate guarantees. Thus, it would act similar to the Controlled Load service class.

For each link on the router, there are four local variables stored in the routers for each service type; the *capacity*, *requested rate*, *confirmed rate*, and *refreshed* capacity. These variables are dynamic and are modified through the use of a signaling protocol. The flow descriptor contained within reservation requests is by far simpler than conventional descriptors, requiring only a peak rate.

Although work is being done to provide multicasting, this architecture is currently for unicast sessions only.

Signaling Protocol
While it may be possible that other more common signaling protocols could be used, a new sender initiated, simplex, lightweight protocol has been proposed, called Sender Oriented Signaling (SOS). There are four types of messages required for this architecture; *reservation*, *confirmation*, *refresh*, and *teardown* messages. While the protocol has been established, current work is being done on how and where to store the flow descriptor within the IP header.

The architecture specifies that all signaling messages are considered part of the Guaranteed Service flow, and are thus immune to congestion and purposeful packet dropping.

Operation
To make a resource reservation, a uniquely identified reservation message is sent to the receiver, with a timer started. If the timer timeouts, it is assumed the reservation was lost, and a new reservation request can be issued. If a reservation fails, the request is dropped. If the reservation is accepted, the rate contained in the flow descriptor is then added to the requested rate field on the appropriate link in the router.

When a request reaches its destination, an acknowledgement message is then returned. This is also done in the form of a reservation message, with

Table 3. Architectural Comparisons

	SRP	FIRST	IOD	EQOS	EAC	SCORE	A-RSVP	SGS
Year	1998	1999	2000	2000	2000	2000	2001	2003
Reservation Initiation	Sender	Receiver	Receiver	Receiver	-	Either	Receiver	Sender
Reservation State	Soft	Hard	Soft	Soft	-	Hard	Soft	Soft
Data Path	Fixed	Fixed	Dynamic	Fixed	Fixed	Fixed	Dynamic	Dynamic
Reservation Types	Homo	Hetero	Either	-	-	-	Either	-
Signaling Band	-	O-Band	O-Band	O-Band	-	O-Band	O-Band	O-Band
Multicasting	Limited	Yes	Limited	No	No	No	Limited	No
Services	CL/BE	CL/BE	CL/BE	CL/BE	CL/BE	GS/BE	GS/CL/BE	CL/BE

the same identifier as the request, and the rate set to zero. Resource reservation requests are typically sent repeatedly until either an acknowledgment is returned, or the sender chooses to stop. If by some chance a reservation request is delayed in the network, and a second request is made, the oldest acknowledgement will be ignored by the sender, with core routers cleaning it up during *garbage collection* (discussed in the next section).

Upon receipt of the acknowledgement, a confirmation message containing the same requested rate is sent back towards the receiver. Once the confirmation is sent, data can start to be sent at the newly reserved rate. As routers encounter the confirmation message, the rate is then added to the confirmed rate field. Finally, a second acknowledgement is sent to acknowledge the confirmation message.

Periodic refreshes are required to be sent once per cycle during the session. Within the network, they are not taken as per-flow refreshes, but rather per-class refreshes. The refreshed rate field is updated appropriately as refreshes are passed through the nodes.

A teardown message, once again containing the reserved rate, is sent to end the reservation. The rate is subtracted from both the requested and confirmed rate fields in the routers.

Garbage Collection

There are two types of garbage collection; short and long term. In short term garbage collection, if a reservation request is rejected while traversing nodes, any reservations upstream will not receive notice. These upstream nodes will still be expecting confirmations, even though the reservation has been rejected. It is necessary to perform garbage collection to restore proper rate levels in the routers.

To conserve resources, garbage collection happens only when the requested resources reach the particular service limit. Any unused reservation space during a timed period is then freed up.

In long term garbage collection, the results of abnormal failures such as link or bit errors are fixed. At the end of each cycle, the refreshed field is compared with the reserved field, with any difference being subtracted from the reserved field.

Advantages

The largest advantages to the Simplified Guaranteed Service architecture are that no per-flow state is required. It also allows for simplified flow descriptors to be used in a simplified signaling protocol, and it is still able to provide some form of Guaranteed Service.

Disadvantages

Currently, there has been little work done to support multicasting for this architecture, and little consideration has been put towards security. Additionally, there has been little experimentation in real networks, relying only on network simulators.

While the additional resources and complexity needed by routers to maintain per-flow state have been removed, there has been the need to introduce other complex features. Two separate garbage collection schemes are needed, as well as the requirement for a router to re-mark packets in the case of a routing change for a particular time period.

Since no delay bounds are possible, a true Guaranteed Services class is not achievable. It would provide a service similar to the Controlled Load class.

6. Future Work

Though many solutions have been proposed to provide end-to-end QoS, one has yet to be embraced by the Internet community. There are several reasons why this is the case. First and foremost, there have been serious performance problems with most of the proposed solutions. Due to these problems, there has yet to be a commonly agreed upon architecture or signaling protocol which can address resource reservation across differing network environments. Even the only standardized signaling protocol, RSVP, has serious scalability concerns. Technical issues aside, business models are lacking which would be required to generate the revenue required for deployment.

6.1 Next Steps in Signaling

Originally started in 2001, the Next Steps in Signaling Working Group (NSIS) is currently working on standardizing a next generation multipurpose signaling protocol [42]. While primarily geared towards providing QoS to data flows, it is also meant for non-QoS applications as well. Design issues such as performance, flexibility, mobility, interoperability, and security will be addressed. The intent of the new signaling protocol, in which some are coining RSVP v2, is to reuse the valuable parts of RSVP, and form a much simpler and modular signaling model.

Currently work is still in the requirements phase, and no solution has been proposed yet.

Two Layered Signaling Protocol

To modularize, the signaling protocol will be broken into two layers; one for common lower layer transport functions, and the other for upper layer application specific signaling functions. These are referred to as the NSIS Transport Layer Protocol (NTLP) and NSIS Signaling Layer Protocol (NSLP) respectively. NSLPs provide custom services for applications, and take advantage of the common NTLP service. The two layered approach allows a simplified design for new signaling applications, and independent development of signaling applications and transport methods. It is also aimed at providing non-QoS signaling capabilities, such as network property discovery and management, and firewall/NAT configuration. Multiple components can be combined in the NSLP layer, such as providing both QoS and Firewall parameters in one set of messages.

Design Goals

The signaling protocol designed by NSIS is intended to be general, and not focused on a single application; it should be useful for all QoS applications and technologies. There should be a distinct separation of the signaling protocol and any control information for extensibility. Similarly, there should also be a separation between signaling and actual QoS provisioning. Generally speaking, it should be comparable to the modular design of RSVP in most aspects, allowing greater flexibility with other protocols. It should be scalable, offer quick setup times, and provide low bandwidth solutions when signaling. Flow aggregations are a must, along with the provisions to allow both unidirectional, and bidirectional flows. There is no specific requirement for multicast, since it will increase complexity and the resources needed.

While there many other design goals listed by NSIS, meeting each one is not necessarily attractive due to the complexity needed. There is currently discussion regarding the importance of many of the design features.

Signaling and Control

The signaling protocol will allow greater flexibility by allowing multiple end-points, and usable in different areas of the Internet without the need to establish complete end-to-end solutions. It should be able to travel end-to-end, end-to-edge, or edge-to-edge. The ability to allow tunneling and hierarchical reservations similar to aggregated RSVP should be permitted as well.

Control messages should use as little resources as possible. Both the reservation identifier and flow identifier should remain independent to support mobility, and a set of flows should also be able to group their control signaling messages to save on messaging overhead.

6.2 Conclusion

There have been many lessons learned over the past decade with regards to QoS. With the need for QoS ever growing, the next few years will present newly standardized protocols which will address the issues that have hindered current architectures.

References

[1] “Introduction to Quality of Service,” White Paper, Nortel Networks.
[2] R. Braden, D. Clark, S. Shenker, “Integrated Services in the Internet Architecture: an Overview,” RFC 1633, June 1994.

[3] J. Forgie, “ST - A Proposed Internet Stream Protocol," Internet Experimental Notes IEN-119, September 1979.
[4] C. Topolcic, “Experimental Internet Stream Protocol: Version 2 (ST-II)," Internet RFC 1190, October 1990.
[5] D. J. Mitzel, D. Estrin, S. Shenker, L. Zhang, “An architectural comparison of ST-II and RSVP,” Proc. of IEEE Infocom '94, June 1994, Toronto, Canada.
[6] L. Delgrossi, L. Berger, “Internet Stream Protocol Version 2 (ST2) Protocol Specification – Version ST2+,” Internet RFC 1819, August 1995.
[7] D. Luca, R. G. Herrtwich, C. Vogt, L. C. Wolf, “Reservation Protocols for Internetworks: A Comparison of ST-II and RSVP,” In 4th Int. Workshop on Networks and Operating System Support for Digital Audio and Video, October 1993.
[8] D. Estrin, S. Shenker, L. Zhang, S. Deering, D. Zappala, “RSVP: A New Resource ReSerVation Protocol,” In IEEE Network, pages 8-18. IEEE, September 1993.
[9] P. Pan, H. Schulzrinne, “Yessir: A Simple Reservation Mechanism for the Internet,” Proc 8th International Workshop on Network and Operating Systems Support for Digital Audio and Video, Cambridge, United Kingdom, July 1998.
[10] P. Pan, H. Schulzrinne, “Lightweight Resource Reservation Signaling: Design, Performance and Implementation,” Bell Labs Technical Memorandum 10009669-03, July 2000.
[11] A. Eriksson, C. Gehrmann, “Robust and secure lightweight resource reservation for unicast IP traffic,” Proc International WS on QoS'98, May 1998.
[12] P. White, J. Crowcroft, “A Dynamic Sender-Initiated Reservation Protocol for the Internet,” 8th IFIP Conference on High Performance Networking, Vienna, September 1998.
[13] G. Fehér, K. Németh et al., “Boomerang: A Simple Protocol for Resource Reservation in IP Networks,” IEEE Workshop on QoS Support for Real-Time Internet Applications, Vancouver, Canada, June 1999.
[14] S. Littlejohns, “Performance Comparison of the RSVP and Boomerang IP Resource Reservation Protocols”, University of Wales, Swansea.
[15] C. Vogt, “Admission Control and Resource Reservation on the Internet,” ACM SIGSOFT, 2002
[16] N. Alborz, B. Chen, L. Trajkovic, “Modeling Packet Scheduling Algorithms in IP Routers,” Simon Frasier University, 2001.
[17] S. Shenker, C. Partridge, R. Guerin, “Specification of Guaranteed Quality of Service,” RFC 2212, September 1997.
[18] S. Shenker, C. Partridge, “Specification of Predictive Quality of Service,” Internet Draft, March 1995.
[19] S. Shenker, C. Partridge, J. Wroclawski, “Specification of Controlled Delay Quality of Service,” IETF Draft, 1995.
[20] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, “An Architecture for Differentiated Services,” RFC 2475, December 1998.
[21] V. Jacobson, K. Nichols, K. Poduri, “An Expedited Forwarding PHB,” RFC 2598, June 1999.
[22] J. Heinanen, F. Baker, W. Weiss, J. Wroclawski, “Assured Forwarding PHB Group,” RFC 2597, June 1999.
[23] D. Clark, W. Fang, “Explicit Allocation of Best Effort Packet Delivery Service,” Internet Draft, 1998.
[24] K. Nichols, V. Jacobson, L. Zhang, “A Two-bit Differentiated Services Architecture for the Internet,” Internet Draft, 1997.
[25] Z. Wang, “User-Share Differentiation: Scalable Bandwidth Allocation for the Internet,” In Proc. HPN’98, September 1998.
[26] A. Odlyzko, “Paris Metro Pricing: The Minimalist Differentiated Services Solution,” Proc. of the IEEE/IFIP International Workshop on Quality of Service - IWQoS'99, June 1999.
[27] C. Dovrolis, P. Ramanathan, “A Case for Relative Differentiated Services and the Proportional Differentiation Model,” IEEE Network, vol. 13, no. 5, September/October 1999.
[28] M. A. Bauer, H. A. Akhand, “Managing Quality of Service in Internet Applications Using Differentiated Services,” JNSM: Vol. 10, No. 1, 2002
[29] E. Rosen, A. Viswanathan, R. Callon, “Multiprotocol Label Switching Architecture,” RFC 3031, January 2001.
[30] V. Fineberg, “QoS Support in MPLS Networks,” May 2003.
[31] J. Chung, M. A Benito, G. Y. Cho, P. Rasiah, H. Chhabra, H. M. Soo, “Extensions to MPLS Networking for Enhanced Multiplatform Multicasting Services,” International Journal of Electronics and Communications, vol. 58, pp. 41-50, Jan. 2004.
[32] P. Pan, “Fast Reroute Extensions to RSVP-TE for LSP Tunnels,” Internet Draft, 2002.
[33] R. Bonica, D. Tappan, D. Gan, “ICMP Extensions for Multiprotocol Label Switching,” Internet Draft, 1999.

[34] W. Almesberger, T. Ferrari, J. Y. Le Boudec, “SRP: a Scalable Resource Reservation Protocol for the Internet,” March 1998.
[35] T. W. K. Chung, H.C.B. Chan and V.C.M. Leung, “Flow Initiation and ReServation Tree (FIRST): A New Internet Resource Reservation Protocol," in Proc. IEEE 1999 Pacific Rim Conf. on Communications, Computers and Signal Processing, Victoria, B.C, pp. 361-364.
[36] Y. Bernet, P. Ford, R. Yavatkar, F. Baker, L. Zhang, M. Speer, R. Braden, B. Davie, J. Wroclawski, E. Felstaine, “A Framework for Integrated Services Operation over DiffServ Networks,” RFC 2998, November 2000.
[37] S. Bhatnagar, B. J. Vickers, “Providing Quality of Service Guarantees Using Only Edge Routers,” In Proceedings of IEEE Globecom, San Antonio, November 2001.
[38] L. Breslau, E. Knightly, S. Shenker, I. Stoica, H. Zhang, “Endpoint Admission Control: Architectural Issues and Performance,” In Proceedings of ACM Sigcomm 2000, Stockholm, Sweden, September 2000.
[39] I. Stoica, “Stateless Core: A Scalable Approach for Quality of Service in the Internet,” 2000
[40] F. Baker, C. Iturralde, F. Le Faucheur, B. Davie, “Aggregation of RSVP for IPv4 and IPv6 Reservations,” RFC 3175, September 2001.
[41] E. Ossipov, G. Karlsson, “SOS: Sender Oriented Signaling for a Simplified Guaranteed Service,” In Proc. of Third International Workshop on Quality of Future Internet Services, pp. 100 – 114, 2002.
[42] R. Hancock, I. Freytsis, G. Karagiannis, J. Loughney, S. Van den Bosch, “Next Steps in Signaling: Framework,” Internet Draft, October 2003.