A State Management Protocol for IntServ, DiffServ and Label Switching

Hari Adiseshu
hari@arl.wustl.edu
Applied Research Laboratory
Department of Computer Science
Washington University in St. Louis
St. Louis, MO 63130

Guru Parulkar
guru@arl.wustl.edu
Applied Research Laboratory
Department of Computer Science
Washington University in St. Louis
St. Louis, MO 63130

Raj Yavatkar
raj@ibeam.jf.intel.com
Communication Architecture Lab
Intel Corp,JF3-206
2111 NE 25th Avenue, Hillsboro, OR 97124

Abstract

*Providing Quality of Service (*QOS*) in an efficient and scalable manner in the Internet is a topic of active research. The technologies that have drawn the most attention are Integrated Services* (IntServ)*, Differential Services* (DiffServ) *and Label Switching(*MPLS*). While these technologies are orthogonal in many respects and can coexist, they are all similar with respect to the fact that some or all the routers in such networks need to be programmed with state information associating data flows with* QOS *classes or priorities or labels. This paper describes a protocol called* SSP *(State Setup Protocol) which is designed to disseminate and manage this state information. In addition to a simple design which makes fast implementations feasible,* SSP *provides many features like state aggregation and third party signaling which makes* SSP *a suitable tool for network configuration, management and provisioning as well. A detailed discussion of* SSP *is presented along with numerous examples and performance measurements.*

1. Introduction

Currently there is a great deal of interest in the Internet community in extending the single best effort class of service to include multiple classes of service. The main motivation is the commercialization of the Internet, prompting network operators to distinguish and prioritize packets belonging to different classes of customers, and the development of new multimedia applications which are not loss and delay tolerant and which require guaranteed network performance. IntServ, DiffServ and Label Switching are the major technologies being pursued to realize QOS in the Internet. We begin by describing each scheme briefly and showing that a common factor in each is the requirement for a protocol to install the requisite flow state in network routers.

1.1. IntServ

An IntServ reference model defines the following elements: multiple classes of service, a means for applications to signal to networks and routers their desired class of service, a means for admission control, flow classification at each router and packet scheduling to ensure each service class gets its required service.

The Internet community has standardized the classes of service in the IntServ WG, and has standardized a signaling protocol called RSVP in the RSVP WG of the IETF. For many reasons, including complexity, scalability and billing, the IntServ scheme and RSVP have not met widespread success or even acceptance in the marketplace. This has lead to a search for a simpler model for IntServ, the direct result being DiffServ or Differential Services.

1.2. DiffServ

DiffServ eliminates the need for applications to signal their individual requirements at run time by letting customers negotiate service *profiles* with their service

providers, which permit customized tailoring of BW usage over a given time period. Providers use profiles to stamp customers' packets with the required QOS at the network entry points. The QOS information is carried in band within the packet in the ToS (Type of Service or Priority) field of the IP header. This eliminates the need to maintain per flow state within the interior of the network. While these are the broad details, the specifics, such as the number of service classes, the nature of the profiles and how out of profile traffic is treated are being resolved in the DiffServ WG of the IETF.

1.3. Label Switching

Related to the idea of providing a scalable QOS solution is the notion of Tag Switching/Label Switching. At its simplest, label switching adds labels to each packet, and lets routers (called Label Switched Routers or LSRs) use the labels to determine forwarding and scheduling decisions. The association between data flows and the labels is set up explicitly through a label switching protocol. The details of the label switching protocol are being worked out in the MPLS WG of the IETF. Label switching can be driven directly by the presence of flows, like IFMP [6], or can be coupled to routing protocols (Tag Switching [7] and ARIS [1]), or can be driven by intelligent network probes which can detect and/or predict network loading patterns.

1.4. State Setup

A common thread running through all these schemes is the need to install state in network routers. In IntServ, RSVP [3] is responsible for installing the $<$ flow, QOS $>$ state in the routers along the path of the flow. In MPLS, it is the responsibility of the label switching protocol to install the mapping between the packets and the associated label along a specified path. Even DiffServ, which explicitly seeks to avoid the complexity of a signaling protocol like RSVP, needs a mechanism to set up the $<$ profile, IP ToS $>$ state in border routers. We ask:

- Is it possible to have a single protocol for installing state in routers for IntServ and DiffServ and Label Switching ?
- And if so, is it possible to do so in a simple and efficient manner ?

We believe our state setup protocol, SSP, meets the above two goals. The following sections describe SSP in depth. In addition to a simple design which makes fast implementations feasible, SSP provides many features like state aggregation and third party signaling which makes SSP a suitable tool for network configuration, management and provisioning as well.

1.5. Outline of Paper

Section 2 describes the design goals and basic functionality provided by SSP. This section motivates SSP design by describing RSVP features and its deficiencies. SSP specific notions like *filters* and *routing addresses* are introduced. Section 3 describes the use of SSP in IntServ and Label switching networks. Different examples are used to show how SSP establishes both unicast and multicast reservations. In particular, the multicast reservations can be tailored to a variety of scenarios, including reservations for a selected subset of senders. Section 4 describes the use of SSP in DiffServ networks to manage state at border routers and as a tool for network management and provisioning. Section 5 provides details of the actual implementation on the NetBSD operating system and performance measurements. The implementation is lightweight enought to support many thousand signaling operations per second. Section 6 describes related work and Section 7 presents our conclusions and plans for future work.

2. Design Goals and Features

To understand the design of SSP, it is necessary to first take a look at RSVP, since SSP is a superset of a simplified subset of RSVP.

2.1. RSVP

RSVP is a receiver initiated soft state simplex two pass resource reservation protocol. In this paper, we assume the readers are familar with the basic design of RSVP. The state that RSVP disseminates is the association or binding between a flow and its QOS. While RSVP introduced several innovative ideas, including receiver initiated reservations and soft state, its design and implementation are unduly complex, Some of the perceived problems with RSVP include:

Two Pass Approach Although the reservation is made only at the receiving end, it is necessary for the sending end also to periodically send messages.

Heterogeneous QOS in Reservations ability of different receivers to request differing QOS for the same multicast flow. This can lead to denial of service attacks, necessitating complex counter measures, such as the newly introduced *blockade state*. Furthermore it increases both size and complexity of state maintained at routers.

Heterogeneous Styles of Reservations This can again lead to ambiguities in merging differing resource requests from different senders.

No Support for Label Switching RSVP provides no intrinsic support for label switching.

No Wildcarded Reservations All the fields in the RSVP filter are fully specified, i.e., no fields are wildcarded in traditional RSVP reservations. While RSVP has recently introduced the notion of a CIDR style wildcarded source and destination address in filters [4], this is still short of full generalization of all fields.

We have briefly listed some of the drawbacks of the current RSVP design. These deficiencies of RSVP, coupled with the need to disseminate state in different types of networks provided the direct motivation for SSP. SSP is a lightweight flow setup and state setup protocol. Based on a subset of RSVP with many additional features, the design is driven by the following goals:

- To eliminate bottlenecks in RSVP described previously.
- To provide a single protocol for programming state information in routers in IntServ, DiffServ and Label Switched networks.
- To do so with minimal number of protocol messages.

2.2. Features

We now discuss the features of SSP. While each feature has an associated cost/benefit tradeoff, the decision to include each has been taken keeping the above goals in mind.

Soft State Like RSVP, SSP reservations install soft state in routers which needs to be refreshed periodically, else the state times out. This greatly helps in keeping SSP simple since a number of error prevention and recovery features which are needed in a hard state signaling protocol can be eliminated. For example, SSP messages can be sent using UDP datagrams, rather than the more heavyweight TCP.

The need for continuous refreshes means that a soft state system must handle a large number of refresh messages in addition to the original reservation message. We describe a simple technique based on label switching to handle refresh messages efficiently.

Single Pass Operation Unlike RSVP which has messages flowing from both the sender and the receiver(s) before a reservation is set up, SSP is based on a single pass. State programming of routers is driven by a single message originating from the receiver side. This single pass operation cuts down on latency while speeding up processing.

In the current implementation of SSP, reservations are initiated by the receiving end. An issue with receiver initiated single pass reservations is that in case of asymmetric routing, the path set up from the sender to the receiver will not be along the default path, since the path setup is done from the receiver end. The assumption here is that flow specific routing overrides default routing, so if a router has forwarding state for a flow which is different from the default forwarding state for that IP address, then the flow specific state will be chosen. We expect that with the evolution of QoS aware routing protocols, this will become a non issue. It is important to note however that nothing in the SSP design precludes a sender initiated version of the protocol, and in fact, a sender initiated version is one of the goals of the next phase of implementation.

Third party reservations Unlike other signaling protocols, SSP allows a reservation to be setup not only from the receiver but also from a third party like a Network Operations Center(NOC). Furthermore, this reservation can be setup not only along the entire path from a receiver to a sender, but also along any chosen subset of the path. This is essential for DiffServ networks, as well as for general network management and configuration of Virtual Private Networks(VPNs).

Single flexible style of reservations SSP provides a single style of reservation which is general enough to account for reserving resources for different senders to a multicast group. The simplification is possible by eliminating heterogeneous requests from different senders. Until such a time that routers can intelligently discard packets based on application content, we feel this is the best choice to reduce complexity in signaling.

Generalized Filters SSP filters which describe the set of packets requiring a particular QOS can be wildcarded to an arbitrary degree. A single filter can describe many different application level flows. This type of state aggregation prevents the state space explosion caused by keeping state on a per flow basis only. Generalized filters can lead to *filter conflicts*, in which a single packet can match multiple filters, leading to ambiguities in mapping packets to filters. Elsewhere [8], we describe a simple geometrical technique to resolve such conflicts.

Support for Label Switching SSP provides native support for label switching, thereby eliminating the need to run separate resource reservation and label switching protocols. This is of special significance for label switched routers which typically have a high speed hardware path for label switched traffic and a slower software path for non label switched traffic. When a resource reservation protocol is invoked before a label switching protocol in such networks, the traffic is forced into

the slow path initially. SSP eliminates such race conditions.

Dynamic Reservations Since SSP is based on soft state, it is necessary to periodically refresh the state which has been set up in the network. This provides an opportunity to *modify* the state based on current requirements, thereby providing optimum utilization of resources. An application or end user can continuously vary its resource requirements based on current requirements by sending its current requirements in each refresh message.

We now turn to the operation of SSP. In order to understand how SSP works it is necessary to look at some of the basic components of SSP messages, and how they are handled at each hop.

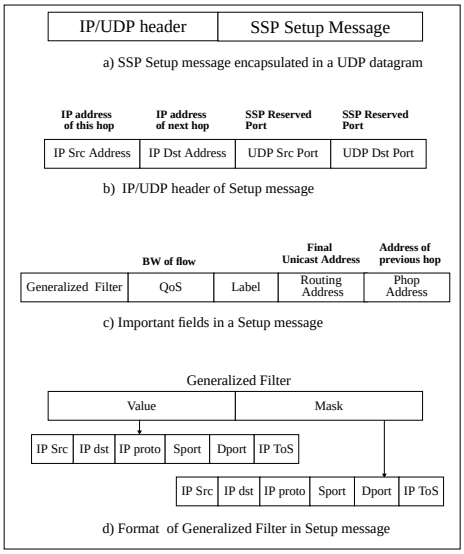

Figure 1. SSP **Setup message**

2.3. Overview

The basic SSP message is the *Setup* message. The Setup message is propagated from the receiver to the sender, or any subset of the path, and as it propagates it installs state in the intermediate routers. The Setup message is transmitted hop by hop in UDP datagrams as shown in Figure 1.a. The IP/UDP header shown in Figure 1.b shows that the Setup messages are sent by adjacent SSP peers on the SSP reserved port. Figure 1.c shows the important components of the Setup message. The *Generalized Filter* describes the packets for which the state is to be setup in the form of the associated QOS and label information. Though SSP itself is independent of the nature of QOS information, for simplicity, we work with a simple model of QOS in which the QOS is represented by a single bandwidth(BW) parameter. In addition, the Setup message also contains a *routing address* which is a unicast address to which the SSP messages are forwarded, as well as the *Phop Address* which is the IP address of the previous hop. The Phop Address is the same as the IP source address of the IP/UDP datagram carrying the Setup message. It is duplicated because user level implementations of SSP may not have access to the IP header of the incoming SSP datagram. The filter carried in SSP is a generalized filter which can be masked to an arbitrary extent as shown in Figure 1.d.

It is the hop by hop forwarding of Setup messages towards the routing address which installs the necessary state in routers. Depending on who initiates the Setup messages, the degree of wildcarding of the filters, and the routing address, many different scenarios are possible, which are described in detail in the subsequent sections.

It is our claim that this approach to programming state is necessary as well as sufficient when the following conditions are met:

- **Information needed to set up state is obtained out of band**. To initiate a reservation request, the requestor needs to know the desired QOS. We assume that this information is available out of band, for example, through a service advertisement protocol, or through well known values associated with individual applications. Thus no signaling messages are needed to obtain this information.
- **State is atomic**. When any node receives a request, the request is either accepted or denied. It is not modified and there are no negotiations. Therefore, no signaling messages are traded back and forth to negotiate a particular QOS.
- **Implicit acknowledgements**. The initiator of the reservation gets an error message in case the reservation is unsuccessful, but no message otherwise. This eliminates the need to have a separate signaling message for positive acknowledgement.

We observe that installing state in a router associating a set of packets with a particular QOS requires that the router receive at least one message. Furthermore, once the message is received by all the relevant nodes, all the programming necessary in the network and end systems is done. In the case of an IntServ network, for example, an end to end path with the desired QOS would have been established. Thus, no additional messages are needed.

3. SSP in IntServ/Label Switching Networks

Our examples in this section are based on networks which support both Label switching and IntServ, however, it is

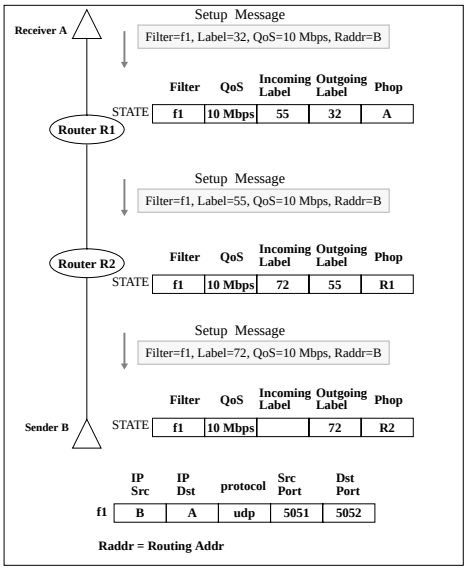

Figure 2. SSP Setup Message used to setup a unicast reservation from node B to node A

fairly easy to understand the behavior of SSP in pure Label switched networks and IntServ networks. Our examples assume that the end user initiates the signaling request, and are intended to illustrate the use of the filter and the routing address in Setup messages to set up flexible reserved label switched paths. Scenarios where the network, as opposed to users, initiates reservations are described in the next section.

3.1. Unicast Reservations

Figure 2 shows an IntServ Label Switched network containing end systems A and B, connected by routers R1 and R2. Node A wishes to reserve some specified BW for a flow from B to A. It sends a Setup message with the relevant fields as shown in Figure 2. As can be seen, the requested BW is 10 Mbps for a UDP flow originating at B from port 5051 and terminating at port 5052 on node A. Node A selects the first hop label to be 32. SSP works with a model of receiver allocation of labels, in which the downstream end of a link allocates the labels to be used by the sending side. The routing address field of the Setup message has been set to B. The Setup propagates hop by hop based on reverse path forwarding to the routing address, namely B, setting up an end to end label switched path with the requested BW provided the Setup passed admission control at each network node. As can be seen, each node along the path selects a new label before forwarding the Setup request, and internally updates its state to maintain the mapping between the incoming and outgoing labels for the flow. Once the Setup reaches B, B can start sending using the label switched path which has been setup from B to A. The state that is set up at each router is also shown in figure, with the incoming and outgoing labels of the data flow.

There are several points about SSP operation that we can make at this point.

- First, node A gets implicit notification of the success of the reservation by the fact that it receives data on the switched path. In case of failure of the Setup, an *Error* message would have been sent back to the originator of the Setup. The Error message would be routed along the same path as the Setup message, thereby canceling the installed state at the intermediate nodes. In the case of a conventional IntServ network (without LSRs), each router would have programmed its scheduler to let this flow's packets receive 10 Mbps BW. In this case, the label field would be left blank. In general, it is not necessary to specify both the QOS field as well as the label field at the same time for networks which do not support both label switching and IntServ.. It is therefore possible, for example, to create an end to end label switched path without any associated BW reservations for a pure label switched network, or to create a reservation without specifying a label for a pure IntServ network. However, as pointed out in Section 7, specifying non zero labels in Setup messages is useful even in non label switched networks for fast processing of Setup refreshes.

- Second, in this example, node A requested a reservation for a fully specified flow from B. By appropriate wildcarding of the filter in the Setup message, it is possible to ask for reservations for a broader set of packets, for example, all TCP packets, or even all packets from B to A. This example also illustrates the use of the routing address to propagate the Setup message from the receiver of data to the sender based on reverse path forwarding. In the case of asymmetric routing, it might happen that the path set up from the sender to the receiver does not match the default path. One important assumption made in SSP is that flow specific state overrides default state. So if a router has information regarding both the default path for a flow and the path based on the newly installed state for the flow, and these two are different, the assumption is that the flow specific state will be used.

- Third, the state installed in the network is *soft state*, meaning that it times out unless it is refreshed periodically. The receivers periodically refresh the state by sending Setup messages. If these stop for some time, each node along the path times out the state and releases any reserved resources. It is also possible to explicitly tear down the state using a *Tear* message.

3.2. Multicast

Having looked at a simple example of SSP being used as a unicast signaling protocol, we now look at some multicast examples which illustrate the flexibility of reservation supported by SSP. Unlike RSVP, in SSP there is no explicit notion of different styles of multicast reservations. The multicast flow setup is similar to the unicast setup. Each receiver of a multicast group which requires non-default delivery of data from senders transmits a Setup message to the first hop router. In this case, the receiver must select a specific sender as the routing address, so the Setup message propagates to a specific sender. The receiver is free to send multiple Setup messages to different senders. We work with an example scenario of a single receiver and two senders. In the first example, we show how the receiver can reserve BW for a single sender by sending a Setup message with the routing address set to the desired sender. In the second example, we show how the receiver can reserve BW jointly for both senders, by sending two setup messages with differing routing addresses, but the same filter specification. And finally in the third scenario, we show how the receiver can reserve BW separately for the two senders by sending two separate Setups with separate filters. These examples correspond to the *Fixed Filter* and the *Wildcarded Filter* style of reservations in RSVP. Similarly it is possible to construct examples corresponding to the *Shared Explicit* style of RSVP, all based on the single type of Setup message of SSP.

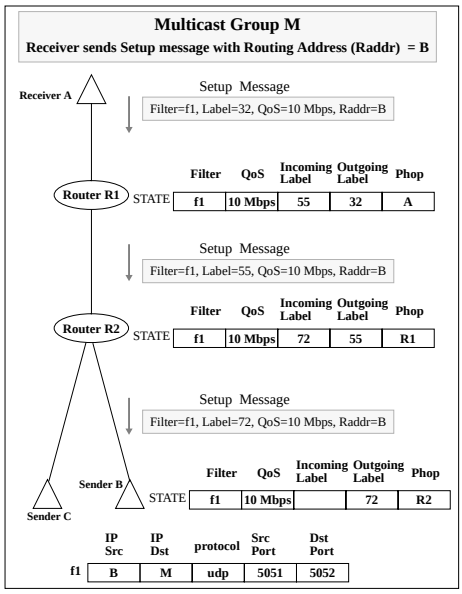

Figure 3. SSP **Setup Message used to setup a reservation for a multicast flow from node B to node A**

The first example is shown in Figure 3. This shows a multicast group M with member A, and two senders B and C. In this example the receiver is interested in associating B's packets with a particular QOS, in this case 10 Mbps. So A sends a Setup message, with the routing address set to B, and with the IP Filter set to match packets sent by B to the multicast group M. The propagation of these Setup message towards B results in the establishment of a label switched path from B to A with a reserved capacity of 10 Mbps, shown in the figure. Packets sent from C to the group M traverse the default path without any reservation.

As Setup messages from different receivers propagate towards the sender, they are *merged* at intervening nodes. Thus there is no implosion problem at the sender. SSP works with a model of homogeneous reservations in which reservations from different receivers to the same sender are expected to request the same QOS, else an error message is returned. Receivers are expected to know the correct BW out of band through service advertising protocols, similar to the way they learn about the existence of senders. It can be shown that heterogeneous reservations, a key feature of RSVP, do not deliver any value without the network knowing the data content of packets, and can lead to denial of service attacks. These problems are avoided with SSP.

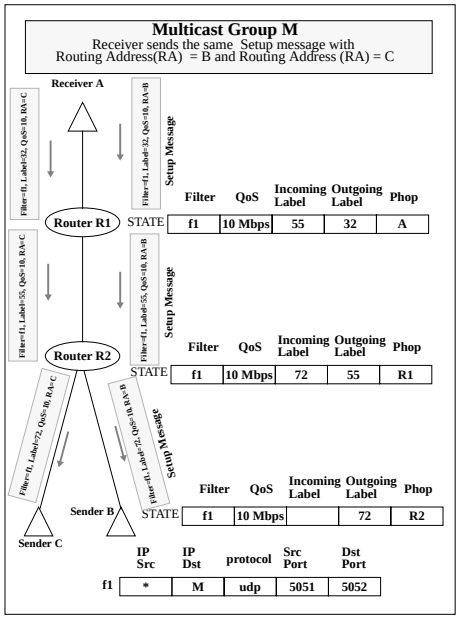

Figure 4. SSP **Setup Messages used to setup a reservation for all multicast packets from B and C to A**

Now, let us consider the same example with multicast senders B and C, and receiver A, and assume that A wants to reserve BW for *all* multicast packets from B and C to A, i.e.,

a shared pipe for data from B and C. In that case we have the scenario shown in Figure 4, in which A sends two separate Setup requests with identical IP Filter fields, but distinct routing addresses as shown. As a result, a switched reserved path is set up from the senders to the receiver, in which the switched path is the same from router R2 to B. As can be seen, the source address in the IP Filter *f1* has been wildcarded to let packets from both sources share the reservation.

In a label switched network based on cell switching there is a problem of cell interleaving if multiple senders are allowed to send on a single label (VC). In our model, we assume that the LSRs support multiple senders on a single VC without cell interleaving, for example, by using frame reassembly. If not, the LSR should send an error message back to the originator of a Setup message, if the message involves multiple senders. Of course, this issue does not arise with conventional routers, or with LSRs based on non-ATM technology. In our examples, we assume LSRs which are able to multiplex multiple senders on a single label without interleaving.

4. SSP for Network Management and DiffServ

A major challenge facing service providers is the provision of Virtual Private Networks (VPNs). At its simplest, a VPN defines a reserved pipe through a service provider, linking different customer sites through an overlay network with some QOS guarantees. We refer to this as a QOS *tunnel*.

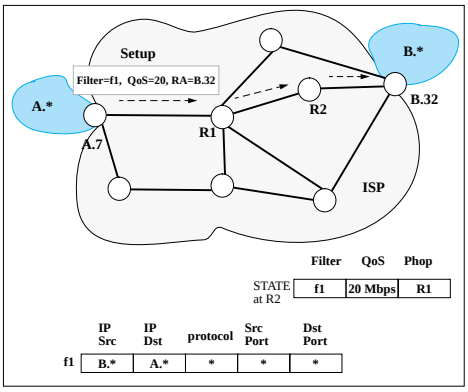

Figure 5. SSP Setup Message used to setup a QOS tunnel between two networks

Consider Figure 5, which shows two distinct networks A and B. Assume that A and B are remote from each other, and that connectivity is provided by an ISP (Internet Service Provider). The ISP needs to provide a QOS tunnel from B to A. We wish to program the routers of the ISP so that packets sent from network B to network A get an assured BW of 20 Mbps. Assume that the border routers are A.7 and B.32 as shown. Setting up a QOS tunnel from net B to net A is easily done by sending a Setup message from A.7 to B.32 as shown in Figure 5. The Filter is used to specify all packets originating in network A and destined for network B. This is done by using the mask field of the IP Filter to mask out all the host related parts of the IP source and destination address, and also to mask out the IP protocol and upper level ports. Also, the routing address is specified as B.32 to terminate the tunnel at B.32. This sets up a simplex QOS tunnel from net B to net A. The state at router R2 is shown in the figure. For setting up a full duplex QOS tunnel, it is necessary for B.32 to send a similar Setup message to A.7. In case the VPN is encrypted, port level information is not available directly from the packet headers without reference to the SPI field. If the QOS tunnel is set up for all packets traversing the VPN then the port level information is not needed. If more differentiation is needed, the filter specification of SSP will have to be enhanced to incorporate the SPI.

4.1. DiffServ

We now turn our attention to DiffServ networks. Several ISPs have already announced support for DiffServ based on the IP ToS field. In this case, border routers (between the customer and the network) stamp selected packets which fit a profile with a higher value of priority in the IP ToS field, and internal routers schedule packets based on this field. The profile is negotiated out of band by the customer and the service provider. This scheme does away with the need for user level signaling and admission control and packet classification. Even though there is no user level signaling in this scheme, we still need a tool which can download the profiles from the ISP's NOC to the border routers. This is a task ideally suited for SSP.

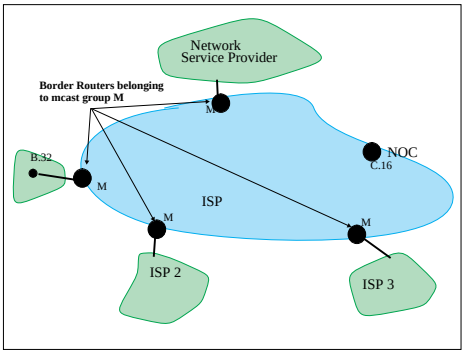

Figure 6. DiffServ network

One interesting and powerful aspect of SSP is how SSP can be used to setup state not only along a specified path as described in previous examples, but also throughout an *entire* network or an arbitrary subset. For example, assume a priority based DiffServ network and assume the following:

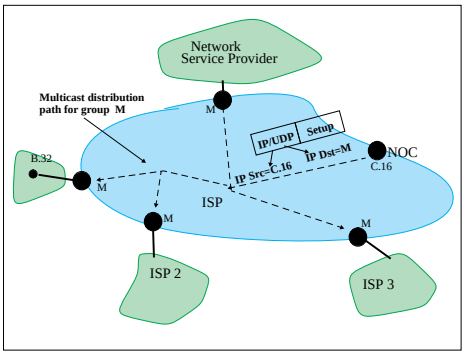

Figure 7. SSP Setup for programming border routers

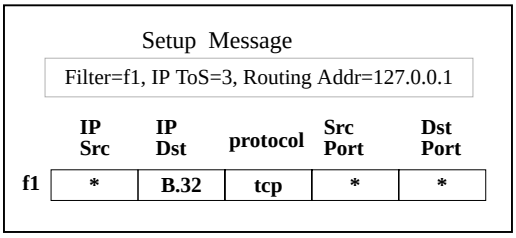

Figure 8. Filter for incoming TCP traffic to a customer

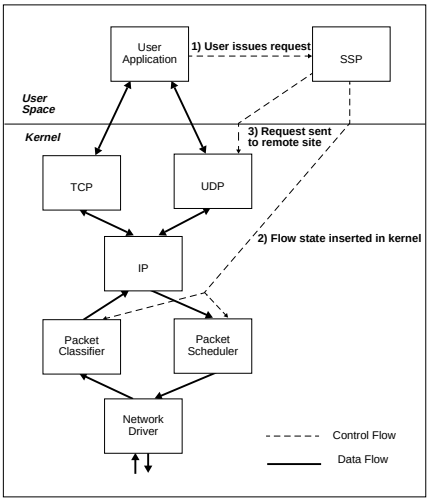

Figure 9. SSP Host Implementation

- A customer has negotiated a profile which lets all the customer's TCP traffic come in as high priority.
- High priority traffic means that the IP ToS byte is set to 3 in IP packets.
- The customer's IP address is B.32.
- All the border routers of the ISP are configured to be part of a multicast group M.
- The ISP's NOC, which keeps track of customers profiles, has address C.16.

Such a network is shown in Figure 6. We wish to program border routers in that network such that all TCP packets to B.32 are stamped with the IP ToS byte set to 3. This can be done using the single Setup message issued from the NOC as shown in Figure 7. The Setup message is sent to all the border routers by addressing the IP packet which encapsulates the Setup message to the special multicast group M which includes all the border routers in the net. The relevant fields of the Setup message are shown in Figure 8. Once the Setup message reaches each border router, it does not propagate further, since the routing address is set to 127.0.0.1 which corresponds to the local loopback address. Each router will treat this address as the terminal condition for the propagation of the Setup message, since it matches its own address. The desired profile is therefore installed in each border router and all traffic which meets this profile, in this case all incoming TCP traffic to B.32, will be stamped with the IP ToS set to 3. As can be seen, the filter field includes the IP ToS.

5. Implementation Details

SSP has been implemented as part of a testbed on IntServ label switching networks [5]. The testbed comprises multiple workstations linked together by label switched routers based on cell switching. The workstations, as well as the control processors of the LSRs are 200 MHz Pentium Pro PCs running the NetBSD operating system. SSP is used as a signaling protocol to reserve resources for application level flows in this environment.

5.1. Host Implementation

The SSP implementation architecture on hosts is as shown in Figure 9. SSP is responsible for setting up the control path so that the data path has the requisite QOS. Applications are linked with a library which provides the SSP API. The SSP API is both simple and powerful, supporting calls both for issuing as well as receiving Setup requests. There are also special calls for setting up the third party reservations described previously.

5.2. Router Implementation

The router implementation is substantially the same as the host implementation, except that in this case several additional modules are exercised.

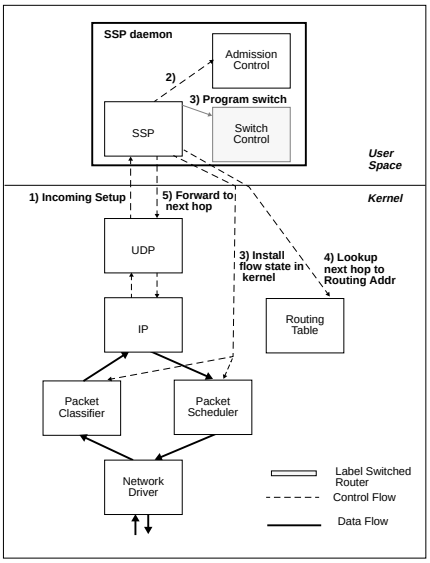

Figure 10. SSP **Router Implementation**

The control and data paths in the router are as shown in Figure 10. When a router receives a Setup message, it first goes through admission control. The admission control module is based on a simple peak BW allocator model, and ensures that the requisite BW in the desired service class is available at the interface the Setup message arrives. For a conventional router, this is followed by programming the packet classifier and scheduler. For an LSR, the switching fabric is instead programmed with the appropriate incoming and outgoing labels. The routing table is then looked up to determine the next hop towards the routing address, and the Setup is relayed to the next hop. These steps set up the data path in the router so that the incoming flow gets the required QOS.

5.3. Implementation Status

The SSP implementation for both end systems and routers resides in the user space. The router implementation also incorporates the admission control and for SSP implementations on ATM subnets, the switch control library within SSP. The switch control library can control ATM switches from Fore Systems and Bay Networks. There is also a VC allocator which is used to allocate VCs to new reservations. Several multimedia applications like **nv**, **vic**, **ttcp** together with a proxy front end called **stap** have been modified to incorporate the SSP API. Using a menu driven interface, it is possible to move from the default path to the QOS path and see the perceptible difference in image quality with a mouse click. The performance of these multimedia applications matches that of native mode ATM applications, for example, both NV and VIC produce full motion 20 Mbps media streams, with the bottleneck being the media processing and image rendering, rather than the network.

5.4. Experimental Results

The signaling throughput and latency were measured for the SSP implementation on NetBSD running on a 200 MHz Pentium Pro based platform. Our measurements indicate that the major overhead in setting up state in a router do not lie in the signaling protocol itself, but rather in the software and hardware components which need to be programmed.

The signaling throughput of SSP was measured by creating a special application called the **signal generator**. The **signal generator** would create a sequence of alternating Setup and Tear requests with a specified QOS and filter, and issue them to SSP via the SSP API at a specified rate. The throughput was measured by counting the number of signaling messages received at the next hop router. With the packet scheduler and classifier module and the switch control module bypassed, SSP could achieve a sustained throughput of 5600 signaling operations per second. However, throughput slumped to a few hundred signaling operations per second with these modules in place. This motivated us to measure the processing times of individual modules in detail. To isolate the running time of individual modules, the code from each module was extracted and exercised in a stand alone manner. Each code fragment was run 1000 times and the average running time calculated as shown in Table 1. As can be seen, the lion's share of the time is taken in setting up the state in the classifer and scheduler and switch.

Module	**Time (in ms)**
Packet Scheduler and Classifier Module	2
Transmitting a signaling request to the Switch Control Library	1.5
Routing Table lookup	0.2
User to daemon latency (round trip)	0.1
Transmitting a UDP *packet*	0.07
Host ATM VC *initialization*	0.04

Table 1. Processing Overhead for Individual Modules in SSP

The current measurements have been taken with no optimizations done to the Setup processing code. We expect these readings to come down substantially once processing on refreshes is to lookup the state entry in the SSP database

based on the label carried by the Setup message. This provides a single fixed length O(1) lookup into the database, avoiding complex filter based lookups. This important benefit hold even for non label switched networks, since there is no reason non label switched networks cannot carry non zero labels in their signaling messages.

6. Related Work

There have been several signaling protocols designed earlier, but none with the same scope and state management capabilities as SSP. In the Internet, ST and ST-2 [9] were examples of a hard state approach to signaling in an IntServ environment, and proved unsuccessful because they depended on a virtual circuit connection oriented service, as opposed to the traditional IP datagram service.

The MCHIP effort [2] specified a *congram* approach towards building of an Internet supporting resource reservation. A congram is a service primitive which combines the low overhead of datagram service with the guarantees of a connection oriented service. This scheme, while conceptually simple, suffers from the drawback of having to use a new congram Internet Protocol for data packets as well as for control packets. With the current datagram based Internet infrastructure firmly in place, we do not expect any other data protocol to succeed.

The ATM Forum has specified a signaling protocol - UNI 3.1 for setting up a flow between two ATM endpoints. The specification itself runs into more than 400 pages of closely packed text, and involves a hard state approach to flow setup which requires extreme reliability in network and link hardware in order to function efficiently.

RSVP was the first soft-state reservation protocol, and the first to allow a default best effort datagram path, but it suffers from the complexity and scalability issues listed earlier. SSP extends RSVP functionality in many ways by adding support for label switching and DiffServ, among other capabilities. In addition, SSP simplifies RSVP by eliminating one of the two passes and the heterogeneous styles of reservations. Recently, the YESSIR reservation protocol [10] has been proposed as a simple substitute for RSVP. YESSIR is a sender oriented reservation protocol which reserves resources for RTP streams. Like SSP, it is single phase. Unlike SSP, however, YESSIR is limited to RTP based traffic, and provides no support for label switching or DiffServ nets.

7. Conclusions and Future Work

We have described the features of a simple and versatile signaling and state management protocol called SSP. Its operation has been illustrated through many examples. SSP contains a number of features not previously available in any single protocol which make it suitable for both signaling and management of IntServ, DiffServ and Label Switched networks. The features range from a single flexible style of reservation to aggregate reservations to generalized filters to single pass operation to third party reservations.

SSP is currently being used to setup reservations over an IntServ label switched prototype network testbed. We also propose to use SSP to manage a DiffServ testbed that we are developing, in the manner described in Section 4. Future work includes extensions to SSP to permit sender oriented reservations and duplex reservations. The switch control library will be written to work with a new Gigabit ATM switch currently under development. In addition, we plan to extend SSP to propagate Ethernet addresses in addition to labels, so that SSP can be used to manage an Ether switched network.

References

[1] A. Viswanathan, N. Feldman, R. Boivie and R. Woundy. ARIS: Aggregate Route-Based IP Switching—Work in Progress. In *draft- viswanathan-aris-overview-00.txt*, March 1997.

[2] Anderson, Jim, Z. Dittia and G.M. Parulkar. Persistent connections in high speed internets. In *Proceedings of the IEEE Globecomm'91*, December 1991.

[3] B. Braden, L. Zhang, S. Berson, S Herxog and S Jamin. Resource Reservation Protocol (RSVP) – Version 1 Functional Specification. In *RFC2205*, September 1997.

[4] J. Boyle. RSVP Extensions for CIDR Aggregated Data Flows — Work in Progress. In *draft-ietf-rsvp-cidr-ext-01.txt*, December 1997.

[5] Decasper D., Waldvogel M., Dittia Z., Adiseshu H., Parulkar G. and Plattner B. Crossbow - A Toolkit for Integrated Services over Cell Switched IPv6. In *Proceedings of the IEEE ATM'97 workshop*, 1995.

[6] P. N. et. al. Ipsilon Flow Management Protocol Specification for IPv4. In *rfc1953*, May 1996.

[7] Y. R. et. al. Tag Switching Architecture Overview—Work in Progress. In *draft-rfced-info-rekhter-00.txt*, September 1996.

[8] Hari Adiseshu, Subhash Suri and Guru Parulkar. Packet Filter Management for Layer 4 Switching. *Submitted to Infocom'99*.

[9] L. Delgrossi and L. Berger. Internet Stream Protocol Version 2(ST2) Protocol Specification - version ST2+. In *RFC1819*, 1995.

[10] Ping Pan and Henning Schulzrinne. YESSIR: A Simple Reservation Mechanism for the Internet. In *NOSSDAV*, 1998.