Report Number: WUCS-98-07

1998-01-01

A Simplified Reservation and State Setup Protocol

Hari Adiseshu, Guru Parulkar, and Subhash Suri

The last few years have seen the development of a model for Integrated Services Internet, which extends the traditional Internet by adding multiple service classes in addition to the traditional best effort service class, and a signaling protocol called RSVP for applications to reserve resources. While this framework has been standardized in the IETF WGs and the RSVP protocol has been defined, there has been no movement towards a commercial implementation of this framework, principally due to its perceived complexity and lack of scalability. This paper analyzes RSVP, discusses some of the its bottlenecks and shows how they can... **Read complete abstract on page 2.**

A Simplified Reservation and State Setup Protocol

Hari Adiseshu, Guru Parulkar, and Subhash Suri

Complete Abstract:

The last few years have seen the development of a model for Integrated Services Internet, which extends the traditional Internet by adding multiple service classes in addition to the traditional best effort service class, and a signaling protocol called RSVP for applications to reserve resources. While this framework has been standardized in the IETF WGs and the RSVP protocol has been defined, there has been no movement towards a commercial implementation of this framework, principally due to its perceived complexity and lack of scalability. This paper analyzes RSVP, discusses some of the its bottlenecks and shows how they can be eliminated to create a trimmer signaling protocol with enhanced functionality and scalability. We have created such a trimmed down version called SSP (State Setup Protocol). Some of the key improvements that we focus on are - single pass operation, elimination of receiver heterogeneity, single unified style of reservation, generalized filter specification, integrated label switching and third party signaling setup.

A Simplified Reservation and State Setup Protocol

Hari Adiseshu, Guru Parulkar and Subhash Suri

WUCS-98-07

February 1998

Department of Computer Science
Washington University
Campus Box 1045
One Brookings Drive
St. Louis MO 63130

A Simplified Reservation and State Setup Protocol

Hari Adiseshu
Guru Parulkar
Subhash Suri
Department of Computer Science
Washington University in St. Louis
St. Louis, MO 63130

Abstract

The last few years have seen the development of a model for Integrated Services Internet, which extends the traditional Internet by adding multiple service classes in addition to the traditional best effort service class, and a signaling protocol called RSVP for applications to reserve resources. While this framework has been standardized in the IETF WGs and the RSVP protocol has been defined, there has been no movement towards a commercial implementation of this framework, principally due to its perceived complexity and lack of scalability. This paper analyzes RSVP, discusses some of its bottlenecks and shows how they can be eliminated to create a trimmer signaling protocol with enhanced functionality and scalability. We have created such a trimmed down version called SSP (State Setup Protocol). Some of the key improvements that we focus on are - single pass operation, elimination of receiver heterogeneity, single unified style of reservation, generalized filter specification, integrated label switching and third party signaling setup.

1 Introduction

The dramatic growth and popularity of the Internet over the last few years has underscored the need to enhance its service capabilities. The development of multimedia applications and the need to provide paying customers with assured service qualities have motivated the requirement to develop classes of services with service guarantees and to provide a way for applications to signal their desired service level. These needs gave rise to the Integrated Services Internet framework [BS94]. According to this framework the basic best effort service class would be augmented by two additional classes of service, the Guaranteed service [SSG97] and the Controlled Load service class [Wro97].

In order for end users to specify their service requirements a signaling protocol called RSVP was developed [HJ97]. While this paper assumes no background on RSVP some of the points made in this paper are better understood by reading an RSVP overview paper [SZ93] or tutorial [RSV97]. Essentially a reservation protocol like RSVP serves to install a mapping between a *filter* and a *qspec* in a network. A filter describes a particular set of packets while the qspec describes the service to be accorded to that set of packets. While this task is seemingly simple, the current specification of RSVP has rendered it complex and difficult to implement. The aim of this paper is to point out various aspects in which RSVP can be improved. These improvements serve to both reduce the latency and processing requirements of RSVP as well as to enhance its functionality. Each of the following sections describes an optimization to RSVP. In Section 2 we describe the two passes of RSVP and show how to eliminate either of the two passes thereby cutting down on latency and increasing the signaling processing capacity. The rest of the paper assumes a single pass receiver oriented version of RSVP. In Section 3 we describe ambiguities present in RSVP protocol processing which can be removed by eliminating heterogeneous reservations for the same session. Section 4 introduces the concept of *terminating addresses* which can be used to set up reservations across a subset of a path. We introduce *generalized filters* in Section 5 and show how they can be used for state aggregation and for creating QoS pipes. We also describe an efficient and optimal algorithm for detecting conflicts across generalized filters. Section 7 shows how to integrate RSVP and label switching and the advantages of integration even in the absence of a label switched network. Finally we describe our implementation of a reservation protocol called SSP (State Setup Protocol) which implements these optimizations and compare it with current work in reservation protocols.

2 Eliminate One Pass

RSVP is a two pass protocol. The first pass consists of PATH messages sent from the sender to the receiver (multiple receivers in case of multicast sessions) followed by RESV reservation messages sent back from the receiver to the sender. PATH messages are addressed by the sender to the receivers and are promiscously picked up by intermediate RSVP-aware routers. PATH messages serve three purposes:

i) Set up state for reverse path forwarding of RESV messages.

ii) Set up state for forwarding RESV message through non-RSVP clouds. Note that RSVP is designed for operation in both RSVP-aware and non-RSVP-aware environments.

iii) Propagate hop by hop network parameters called *adspec* to the receiver.

RESV messages flow back from the receivers to the senders. Unlike PATH messages they flow hop by hop towards the sender. Each hop forwards the RSVP message to the next hop towards the sender based on the state established by the PATH messages. RESV messages serve the following purposes:

a) Convey the necessary reservation to the sender and to each intermediate RSVP-aware router.

b) Allow merging to prevent implosion at sender.

c) Allow different reservation styles.

These two phases are illustrated for multicast reservations in Figure 1.

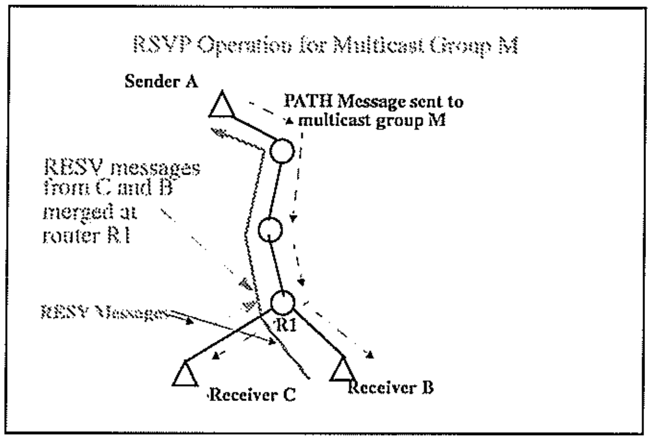
RSVP Operation for Multicast Group M
Sender A
PATH Message sent to multicast group M
RESV messages from C and B merged at router R1
RESV Messages
R1
Receiver C
Receiver B

Figure 1: RSVP Operation

We first ask if both phases are necessary, i.e., if we can work only with PATH messages or only with RESV messages. The answer with some caveats which we explain, appears to be yes.

First consider the case when there are only PATH messages and no RESV messages. We now need to provide the functionality enumerated above in RESV messages. Clearly with only PATH messages we have a sender initiated reservation scheme, so a) is reversed- the reservation messages are now used to send reservations from the sender to the receiver rather than the other way around.

Clearly b) is not an issue since the PATH messages are fanning out from the sender to the many receivers rather than the other way around.

As for c), RSVP allows three different styles of reservations - fixed filter (FF), shared explicit (SE) and wildcard filter (WF) styles [HJ97] which we discuss in Section 6. Again, it is possible for senders to mark their PATH messages with one of the three styles and for intervening routers to set up the appropriate kind of reservation. So for example, if all the senders were to mark their reservations as being WF then each receiver would see only a composite reservation.

As can be seen, it is fairly straightforward to merge the functionality of RESV messages into PATH messages provided we move from a receiver initiated model of reservation to a sender initiated model.

Let us consider the other alternative to creating a single pass reservation protocol, namely eliminating PATH messages.

Regarding i) there are two possibilities for forwarding RESV messages back to the sender. One is reverse path forwarding in which the RESV message takes the path taken by packets from the receiver to the sender. In general this might not correspond to the default path from the sender to the receiver. However this is not an issue if the intermediate hops do flow based forwarding, i.e., forward packets based on the per flow state installed by RSVP rather than the default forwarding state. So in case the default path is via one interface and the flow state points to another interface there is no problem as long as the flow state is given priority over the default state.

In case it is desired that the RESV messages always traverse the default path from the sender to the receiver it is necessary for the sender to send data initially before the receiver sends a RESV message. The initial data will serve as a pseudo PATH message by installing state in intermediate hops regarding the path to the sender. For example, this approach is taken by IFMP [ea96b].

With the establishment of a Quality of Service (QoS) Routing WG in the IETF [OoS97] and a framework for QoS routing [RS97] the Internet is likely to see QoS based flow routing shortly. The two pass scheme does not fit well with QoS routing since in the two pass scheme the route is determined by PATH messages while the actual reservation which should really determine the path is carried in the RESV message. No such problems exist for a single pass reservation protocol.

As for ii) PATH messages contain a field (RSVP_HOP) which is filled by each RSVP-aware hop with its outgoing IP address. In case the PATH message traverses a non-RSVP-aware cloud the RSVP_HOP field contains the IP address of the last RSVP-aware hop. When an RSVP-aware router receives a RESV message setting up a reservation for a flow, it looks up the corresponding PATH message for that flow and forwards the RESV message to the IP address given in the RSVP_HOP field of the PATH message. This ensures that the RESV message can traverse non-RSVP-aware clouds.

Without PATH messages how can a router know the next hop RSVP-aware router ? We assume that in the single pass version of the protocol RSVP-aware hosts and routers run an adjacency protocol which detects if their immediate neighbors are RSVP-aware or not. In case a router detects that the next hop router is non-RSVP-aware and it needs to forward a RESV message towards this router then it simply addresses the RESV message directly to the sender. The RESV message tunnels through the non-RSVP-aware cloud and is picked up by the first RSVP-aware router on the other side of the cloud which resumes the hop by hop forwarding. Note that this is exactly the same scheme that PATH messages use to propagate through non-RSVP-aware clouds. This illustrated in Figure 2.

We see that iii) is not part of a signaling protocol. It belongs to a separate resource discovery protocol, not resource reservation protocol. It leads to unnecessary computation and propagation of values which are needed only once. In case it is really necessary the values can always be propagated to the sender by being tagged to RESV messages rather than by being tagged to PATH messages.

We now see that both passes of RSVP are not really needed for the functioning of a reservation protocol. The choice as to which pass to retain depends on which direction reservations are more likely to be made— from the sender or from the receiver. We feel that with the current Internet there are considerable advantages to sticking with a receiver initiated reservation protocol and hence propose a single pass receiver oriented version of RSVP by eliminating PATH messages from RSVP. The remaining sections assume such a single pass protocol.

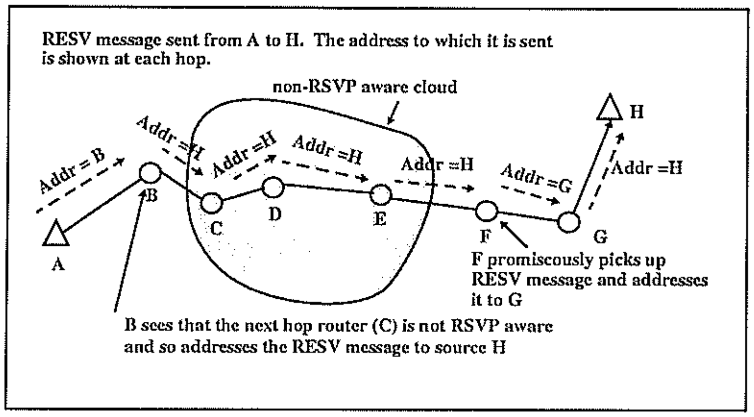
RESV message sent from A to H. The address to which it is sent is shown at each hop.
non-RSVP aware cloud
Addr =B
Addr =H
Addr =H
Addr =H
Addr =H
Addr =G
Addr =H
A
B
C
D
E
F
G
H
F promiscously picks up RESV message and addresses it to G
B sees that the next hop router (C) is not RSVP aware and so addresses the RESV message to source H

Figure 2: Operation of single pass receiver oriented version of RSVP across non-RSVP-aware clouds. Hop by hop forwarding of RESV messages in such an environment is shown.

2.1 Single Pass Receiver Initiated Reservations and Multicast Flows

Note that in the current Mbone multicast trees are set up using DVMRP which uses reverse path forwarding. Hence the tree setup by the single pass version of RSVP will coincide with the default tree setup by DVMRP. The use of RSVP with other multicast routing protocols has not yet been studied. Here again with QoS aware routing protocols a single phase setup protocol is at an advantage compared to RSVP. In the current RSVP, RESV messages are forced to be routed along the same path as the PATH message. However the actual reservation is carried in the RESV message while the QoS routing protocols are used to route only the PATH messages as described earlier. This raises serious doubts about the suitability of the current RSVP design for future QoS routing protocols. On the other hand these protocols fit in neatly with a single phase receiver oriented setup protocol.

3 Eliminating Heterogeneous Reservations for the Same Session

For a unicast session the QoS is chosen by a single receiver. For multicast sessions each receiver is free to choose a different QoS. Reservations originating from different receivers are *merged* as the RESV messages propagate towards the sender. The RSVP protocol has as an explicit goal the aim of providing resource reservations for heterogeneous receivers. While this is a laudable goal the current implementation of this goal in the form of allowing receivers to specify arbitrary reservations in RESV messages can severely impact the scalability of the protocol and the performance of best effort receivers. There are two serious objections to heterogeneous reservations.

- Heterogeneous reservations cause extra RESV messages destroying scalability of the RESV merge protocol.

The scalability of the RESV merge procedure is based on the fact that new reservations are merged into existing reservations thereby preventing an all new reservation from propagating all the way to the sender from each receiver.

However, consider the simple case of a sender and multiple receivers. If each succeeding receiver initiates a RESV message with an incremental increase in the BW then each new RESV message would have to be propagated all the way to the sender. If the receivers number in the millions this is clearly an impractical proposition. So heterogeneous reservations requirements can cause new RESV messages to propagate all the way to the sender instead of flowing to the nearest point on the current multicast tree.

- Heterogeneous reservations cannot deliver layered encoding.

  Why should it be necessary for receivers to ask for differing reservations or differing rates? This makes sense only if the sender is using a form of layered or hierarchical encoding which delivers different layers of information for differing bandwidth (BW) capacities of different receivers. However, in the current scheme the trimming of BW for heterogeneous rate specifications occurs at the merge points and is done within the packet scheduling software of the router which does not have any knowledge of the information content of the packets. In such a case it makes more sense for the sender to initiate distinct flows each carrying a different layer of encoding and for the receiver to request a reservation only for those layers which satisfy its BW requirements. This is sometimes referred to as MMGs (multiple multicast groups). However, with MMGs each MMG has a distinct BW requirement and a receiver should ask for precisely that BW which is necessary for the set of MMGs which it intends to receive. There is no heterogeneity in resource reservations with MMGs. So clearly heterogeneous rate specifications do not deliver any benefits to receivers.

  The simplest way to ensure BW and service class homogeneity within a given session would be for the sender to reject a non-conformant reservation request with an appropriate error code. Once this percolates down to to offending receiver, the routers along the path know the right value of the reservation so that they can deny future non-conformant reservations.

4 Separation of Terminating Address from Filter Specification

Define a *terminating address* as the address of an endpoint which terminates RSVP messages. In the current RSVP only the IP source and destination addresses specified in the filter carried in RSVP messages act as terminating addresses. Consider the topology shown in Figure 3. Networks A and B are separated by a cloud. Network A contains host A.1 and border gateway A.2. Net B contains host B.1 and border gateway B.2. With RSVP it is not possible to setup a reserved pipe between the two border routers for a flow from A.1 to B.1 - RSVP can only setup an end to end pipe from A.1 to B.1.

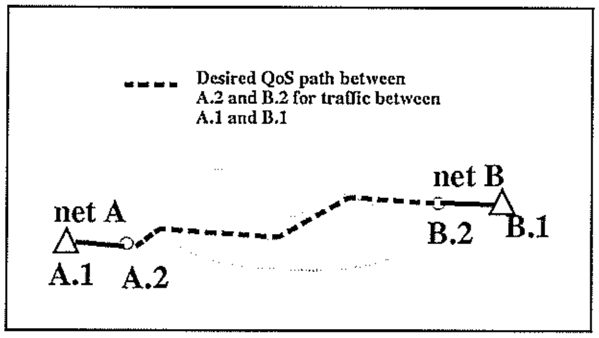
Desired QoS path between A.2 and B.2 for traffic between A.1 and B.1
net A
A.1
A.2
net B
B.2
B.1

Figure 3: RSVP not able to set up a reservation along a subset of the path

If we decouple terminating addresses from filter specification then we have a powerful tool for allowing third party reservations and for setting up reserved QoS pipes between sites in conjunction with generalized filters discussed in Section 5. This can be easily done by adding the terminating address explicitly in reservation messages. In the single pass receiver oriented setup protocol a reservation initiating entity would send a reservation message containing a <filter, QoS> mapping together with a terminating address. The reservation message would be propagated hop by hop to the terminating address. Thus the reservation would be setup between the initiating entity and the terminating address. In the example that we discussed earlier, if we wish to setup a reservation between A.2 and B.2 then B.2 would transmit a reservation request with the terminating address set to A.2 thereby ensuring that the reservation propagates from B.2 to A.2. This is shown in Figure 4. Of course, for duplex reservations A.2 would have to issue a similar reservation with terminating address B.2.

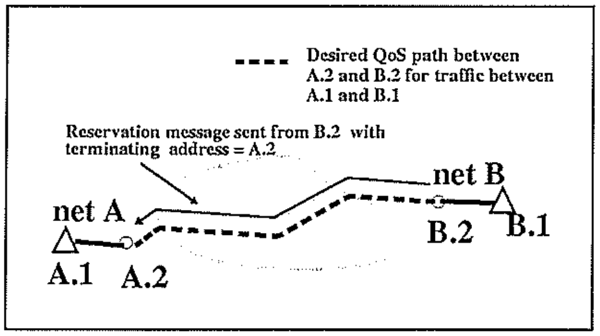
Desired QoS path between A.2 and B.2 for traffic between A.1 and B.1
Reservation message sent from B.2 with terminating address = A.2
net A
A.1
A.2
net B
B.2
B.1

Figure 4: Setting up reservations along a subset of the path using terminating addresses

5 Generalized Filters

In RSVP, filters are expected to be well specified, i.e. the fields are supposed to be well defined. The current standard defines a 5 tuple consisting of IP source and destination addresses, IP protocol type and upper level protocol ports as shown in Figure 5. Currently the upper level protocols are restricted to TCP and UDP. Recently its been proposed [Boy97] to enhance filter specification to allow CIDR style IP source and destination addresses. However, this proposal does not allow for wildcarding of all the fields in the filter specification, specifically, the protocol type and the upper level source and destination ports are not wildcarded. We instead propose the generalized filter shown in Figure 5. As can be seen the generalized filter consists of a value part and a mask part. The mask part acts as a bitmask specifying which bits of the value part are significant. There are a number of advantages of such generalized filters. First, we can aggregate state since a single generalized filter can replace many individual per flow reservations. Second, we can use generalized filters for network management and capacity allocation by setting up QoS pipes and Virtual Private Networks (VPNs) between different sites. Third, it allows reservations to be used without end system participation. Forth, it allows for a single flexible style of reservation to replace the three different styles of reservation in RSVP as discussed in Section 6.

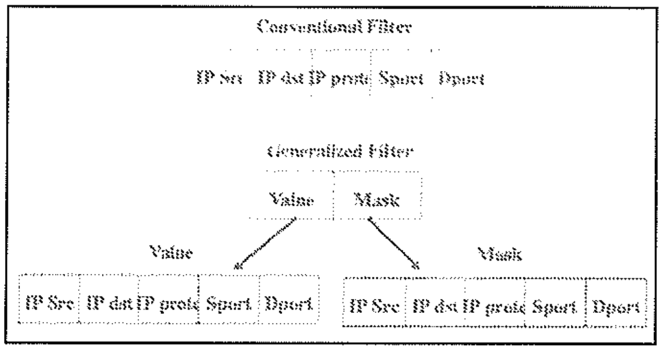

Figure 5: Generalized Filters

5.1 Generalized Filters and Conflict Resolution - a Geometric Perspective

While generalized filters provide a scaling mechanism for aggregating state information regarding a large number of flows they can also lead to ambiguities in packet classification. This occurs because with generalized filters it is is a possible that a packet might get mapped to multiple filters each with a different QoS leading to a conflict in packet classification.

To illustrate this problem consider the case of simple two tuple filters consisting of source and destination IPv4 address prefixes. The two fields can either be fully specified or wildcarded, or can be partially wildcarded in standard prefix format. Let $x.*$ denote the prefix corresponding to network x and let $y.*$ denote the prefix of network y. Consider the two filters $< (x.*, *), QoS1 >$, and $< (*, y.*), QoS2 >$. The first filter assigns all packets from source network x a QoS equal to QoS1 while the second filter assigns all packets destined to network y a QoS equal to QoS2. In case the router receives a packet sourced from net x and destined to net y we have a conflict since the packet matches both filters. In case of conflict which filter should be selected ?

Some of the possibilities are:

a) Assign the first match in the filter database. So for example, if $< (x.*, *), QoS1 >$ is the first match for the incoming packet then the packet gets QoS1. On the other hand, if $< (*, y.*), QoS2 >$ is matched first then the packet gets QoS2. This approach is commonly taken to resolve conflicts in firewalls where incoming packets are matched against filters specified in access control lists and the first access control list the packet matches determines if the packet crosses the firewall or is dropped.

b) Assign priorities to different filters so that in case of multiple matches the highest priority filter is returned. However, this scheme turns out to be identical to scheme a) if we sort the filters in the order of priority.

c) Rather than assigning priorities to filters, assign priorities to fields so that in case of multiple matches the filter with the most specific matching field with the highest priority is selected. For example, if the source address is given higher priority on matches than the destination address, then for packets going from network x to network y the filter $< (x.*, *), QoS1 >$ is a better match than $< (*, y.*), QoS2 >$ since matching the source field is more important that matching the destination field and the source field is a better match with the first filter $(x.*)$ than with the second filter $(*)$.

However, these types of implicit conflict resolution schemes while simple to implement, suffer from some serious drawbacks. For example, in case a), depending on whether $< (x.*, *), QoS1 >$ is listed first or $< (*, y.*), QoS2 >$ is listed first packets from net x to net y will be classified as having QoS1 or QoS2. Thus this scheme imposes an arbitrariness on the conflict resolution.

Similarly with c) there is no way we can assign QoS2 to packets from net x to net y with the filters described above. This scheme substitutes arbitrariness with inflexibility in filter matching..

At this point we make a crucial observation. We note that if we introduce the filter $< (x.*, y.*), QoSz >$ where QoSz is the QoS level we wish to assign to packets from net x to net y and if we resolve filters based on most specific match on *all* fields, then we eliminate the conflict with respect to packets from net x to net y. This is because now these packets will match $< (x.*, y.*), QoSz >$ rather than $< (x.*, *), QoS1 >$ or $< (*, y.*), QoS2 >$. In short, we have resolved the filter conflict by the addition of a new filter which explicitly tells us the QoS to assign in the region of conflict.

The question arises whether given a set of filters it is possible to resolve conflicts by the addition of new filters in a simple, computationally efficient way. We believe the answer is yes. Our solution is motivated and better understood by a geometric transformation of the filter matching problem.

To obtain this geometric transform we note that we can graphically view the two tuple filter described earlier as a region of a discrete 2 dimensional space bounded by 0 and 2^{32} on either axis.

Incoming packets correspond to points in this space. The problem of matching packets to filters is transformed into the geometric problem of matching points to regions within this discrete 2D space.

To illustrate this mapping between filters and geometric spaces let us consider some simple 2-tuple filters as shown in Figure 6. A fully specified filter is a point, shown in figure

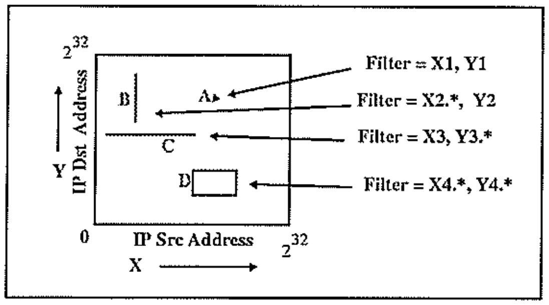

Figure 6: Geometric representation of two tuple Filters. The two fields are the IPv4 source and destination addresses

as filter A with IP source address X1, and IP destination address Y1. A filter well specified in one field and wildcarded in another field is a line. As shown in the example, filters B and C fall into this category. For B the IP source address is partially wildcarded (X2.*) while the IP destination address is fully specified (Y2). A filter wildcarded in both fields is a rectangle like filter D in which both the source and destination IP addresses are partially wildcarded.

Two filters can overlap with each other in two different ways. One is a *full overlap* in which every point of one filters is contained in the other. As shown in Figure 7 filter C is completely contained in filter D. We refer to C as the inner filter and D as the outer filter.

The other type of overlap is a *partial overlap* also shown in Figure 7 with filters A and B. Note that in case of partial overlap neither filter is fully contained within the other.

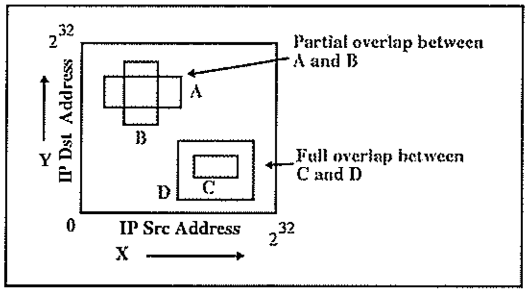

Figure 7: Overlapping Filters - Partial and Full Overlap

We note that *two filters conflict if and only if they have a partial overlap.* In case of full overlap a point which falls within the inner filter is matched to the inner filter rather than the outer filter since the inner filter is more tightly specified in all dimensions. In case of partial overlap we have a conflicting region which is the area which is common to both filters. A point within this area cannot be unambiguously matched to either filter since neither is better specified than the other in both axes.

Consider Figure 8 which shows two filters A and B partially overlapping. Note that if an incoming packet maps to region a1 or b1 there is no conflict as to which filter it corresponds to, but if the packet maps to region a2 which is the region of overlap then we cannot resolve if the packet belongs to filter A or B.

We make two important observations at this time.

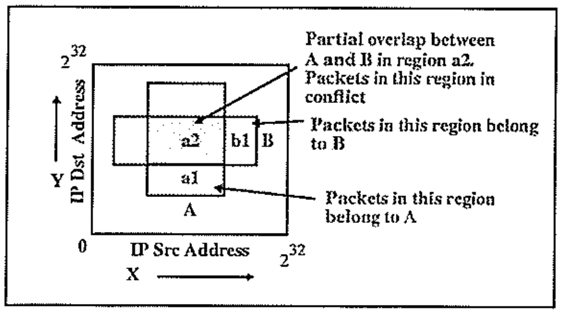
Partial overlap between A and B in region a2. Packets in this region in conflict
Packets in this region belong to B
Packets in this region belong to A
2^{32}
Y IP Dst Address
a2
b1
B
a1
A
0
IP Src Address
2^{32}
X

Figure 8: Conflicts caused by partial overlap

- If each field of the filter is a prefix, conflicts can only be of the categories shown in Figure 9A. In particular it is not possible to have conflicts of the type shown in Figure 9B. These types of conflicts are only possible in case the fields are arbitrary ranges rather than prefixes.
- By the addition of a new filter which covers the conflicting region we can eliminate conflict between two conflicting filters. For example, in Figure 8, if we add a new filter covering the region a1 then we resolve the conflict between filters A and B. figure 6.

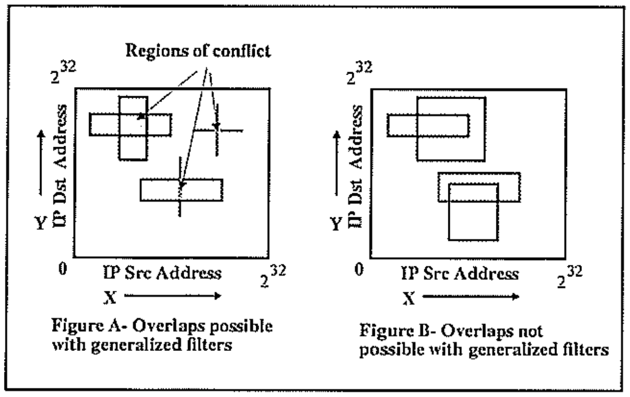

Figure 9: Types of conflicts possible and not possible with generalized prefix wildcarded filters

Based on this geometric view of filter conflict detection, in Appendix A we describe a simple algorithm which when given a new filter checks to see if it conflicts with any of the existing filters and if so, returns the region of conflict. We first present the algorithm for the simple 2 tuple filter conflict detection problem and show how it can be extended to n tuple filters. We believe this conflict detection scheme is optimal in the sense that it is not possible to detect and resolve all the possible conflicts generated by the addition of a new filter in fewer steps. With this algorithm each time a router receives a reservation with a generalized filter it can quickly check to see if the new filter conflicts with any existing filter or not. The simplest way of handling conflicts would be to send back an error message containing the region of conflict. Note that if the router now receives the original filter and a new filter corresponding to the region of conflict the conflict is effectively removed. Of course we assume that only authorized users like network managers or administrators will be allowed to initiate reservations with wildcarded generalized filters while end users will be restricted to fully specified filters.

6 Eliminating Different Styles of Reservations

Currently when a receiver uses RSVP to make a multicast reservation the receiver needs to specify not only the class of service and its associated parameters (QoS), but also a style of reservation which determines which set of senders are associated with the specified reservation. As mentioned earlier RSVP supports three styles of reservations:

- *Wildcard Filter*(WF) in which the QoS specified is used for all senders.
- *Shared Explicit*(SE) in which the QoS specified is used for the explicitly enumerated set of senders and
- *Fixed Filter*(FF) in which each sender is associated with a separate reservation. This is the only style for unicast reservations.

These different styles of reservation lead to a lot of implementation complexity in RSVP and also pose problems in merging heterogeneous resource requests. How does one merge WF filters with SE or FF ? How does one merge two different SE filters ? These questions are not answered by the current RSVP design - they simply cause an error message to flow back towards the receivers.

We propose instead a *single flexible style of reservation* to replace the above three. This is done using the generalized filters described earlier. Basically by masking the source address in the generalized filter to the desired extent we can recreate RSVP's styles of reservation. If the source address field in the filter is fully wildcarded, for example, we would get a reservation corresponding to the wildcard style of reservation. If the source is fully specified we would get a reservation corresponding to the fixed filter reservation. And if the source is partially wildcarded we would get an SE style reservation. For example, if the source address in the generalized filter is wildcarded to correspond to a network then a reservation is made for all sources from that network.

7 Label Switching

Tag switching [ea96d] [ea96a] or ARIS [Wou97] or Label switching [Vis97] schemes have recently generated a great deal of interest. This concept was first proposed as *threaded indices* [CV95]. Label switching allows use to replace variable length prefix lookups with fixed size label lookups. This is also used in Cell switching or IP switching to map IP flows to ATM VCs [ea96c]. However, Ipsilon's solution does not currently permit QoS specification for a flow. Thus in an IP switched network we would have to run IFMP to map IP flows to ATM