6.8 RSVP

As we learned in the previous section, in order for a network to provide QoS guarantees, there must be a signaling protocol that allows applications running in hosts to reserve resources in the Internet. RSVP [RFC 2205], is such a signaling protocol for the Internet.

When people talk about *resources* in the Internet context, they usually mean link bandwidth and router buffers. To keep the discussion concrete and focused, however, we shall assume that the word *resource* is synonymous with *bandwidth*. For our pedagogic purposes, RSVP stands for Bandwidth Reservation Protocol.

6.8.1 The Essence of RSVP

The RSVP protocol allows applications to reserve bandwidth for their data flows. It is used by a host, on the behalf of an application data flow, to request a specific amount of bandwidth from the network. RSVP is also used by the routers to forward bandwidth reservation requests. To implement RSVP, RSVP software must be present in the receivers, senders, and routers. The two principle characteristics of RSVP are:

1. It provides **reservations for bandwidth in multicast trees** (unicast is handled as a special case).
2. It is **receiver-oriented**, i.e., the receiver of a data flow initiates and maintains the resource reservation used for that flow.

These two characteristics are illustrated in Figure 6.8-1:.

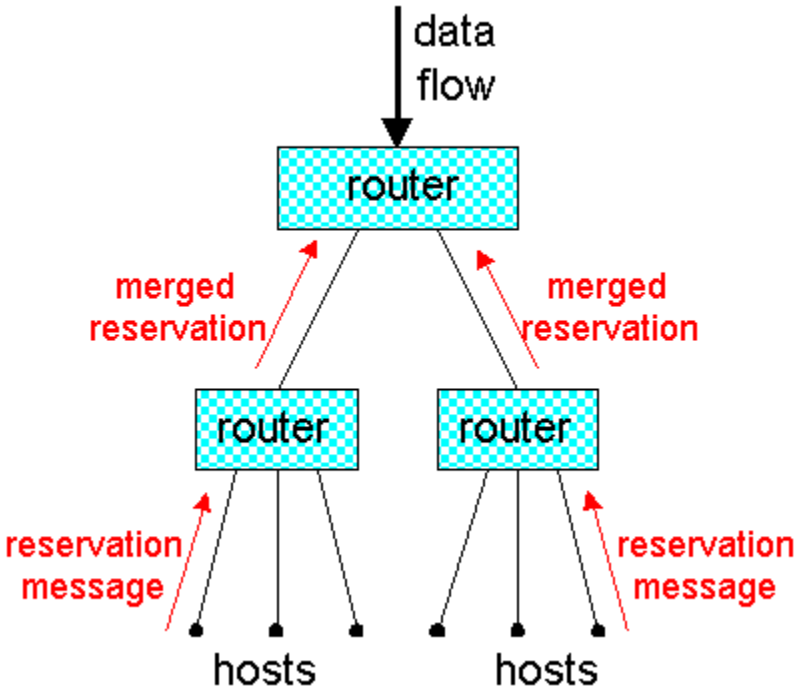

Figure 6.8-1: RSVP: multicast- and receiver-oriented.

The above diagram shows a multicast tree with data flowing from the top of the tree to six hosts. Although data originates from the sender, the reservation messages originate from the receivers. When a router forwards a reservation message upstream towards the sender, the router may merge the reservation message with other reservation messages arriving from downstream.

Before discussing RSVP in greater detail, we need to recall the notion of a **session**. As with RTP, a session can consist of multiple multicast data flows. Each sender in a session is the source of one or more data flows; for example, a sender might be the source of

a video data flow and an audio data flow. Each data flow in a session has the same multicast address. To keep the discussion concrete, we assume that routers and hosts identify the session to which a packet belongs by the packet's multicast address. This assumption is somewhat restrictive; the actual RSVP specification allows for more general methods to identify a session. Within a session, the data flow to which a packet belongs also needs to be identified. This could be done, for example, with the flow identifier field in IPv6.

What RSVP is Not

We emphasize that the RSVP standard [RFC 2205] does not specify how the network provides the reserved bandwidth to the data flows. It is merely a protocol that allows the applications to reserve the necessary link bandwidth. Once the reservations are in place, it is up to the routers in the Internet to actually provide the reserved bandwidth to the data flows. This provisioning is done with the scheduling mechanisms (prirority scheduling, weighted fair queuing, etc.) discussed in Section 6.6.

It is also important to understand that RSVP is not a routing protocol – it does not determine the links in which the reservations are to be made. Instead it depends on an underlying routing protocol (unicast or multicast) to determine the routes for the flows. Once the routes are in place, RSVP can reserve bandwidth in the links along these routes. (We shall see shortly that when a route changes, RSVP re-reserves resources.) And once the reservations are in place, the routers' packet schedulers can actually provide the reserved bandwidth to the data flows. Thus, RSVP is only one piece – albeit an important piece – in the QoS guaranteee puzzle.

RSVP is sometimes referred to as a *signaling protocol*. By this it is meant that RSVP is a protocol that allows hosts to establish and tear-down reservations for data flows. The term "signaling protocol" comes from the jargon of the circuit-switched telephony community.

Heterogeneous Receivers

Some receivers can receive a flow at 28.8 Kbps, others at 128 Kbps, and yet others at 10 Mbps or higher. This heterogeneity of the reservers poses an interesting question. If a sender is multicasting a video to a group of heterogeneous receivers, should the sender encode the video for low quality at 28.8 Kbps, for medium quality at 128 Kbps, or for high quality at 10 Mbps? If the video is encoded at 10 Mbps, then only the users with 10 Mbps access will be able to watch the video. On the other hand, if the video is encoded at 28.8 kbps, then the 10 Mbps users will have to see a low-quality image when they know they can something much better.

To resolve this dilemma it is often suggested that video and audio be encoded in layers. For example, a video might be encoded into two layers: a base layer and an enhancement layer. The base layer could have a rate of 20 Kbps whereas the enhancement layer could have a rate of 100 Kbps; in this manner receivers with 28.8 access could receive the low-quality base-layer image, and receivers with 128 Kbps could receive both layers to construct a high-quality image.

We note that the sender does not have to know the receiving rates of all the receivers. It only needs to know the maximum rate of the all its receivers. The sender encodes the video or audio into multiple layers and sends all the layers up to the maximum rate into multicast tree. The receivers pick out the layers that are appropriate for their receiving rates. In order to not excessively waste bandwidth in the network's links, the heterogeneous receivers must communicate to the network the rates they can handle. We shall see that RSVP gives foremost attention to the issue of reserving resources for heterogeneous receivers.

6.8.2 A Few Simple Examples

Let us first describe RSVP in the context of a concrete one-to-many multicast example. Suppose there is a source that is transmitting into the Internet the video of a major sporting event. This session has been assigned a multicast address, and the source stamps all of its outgoing packets with this multicast address. Also suppose that an underlying multicast routing protocol has established a multicast tree from the sender to four receivers as shown below; the numbers next to the receivers are the rates at which the receivers want to receive data. Let us also assume that the video is layered encoded to accommodate this heterogeneity of receiver rates.

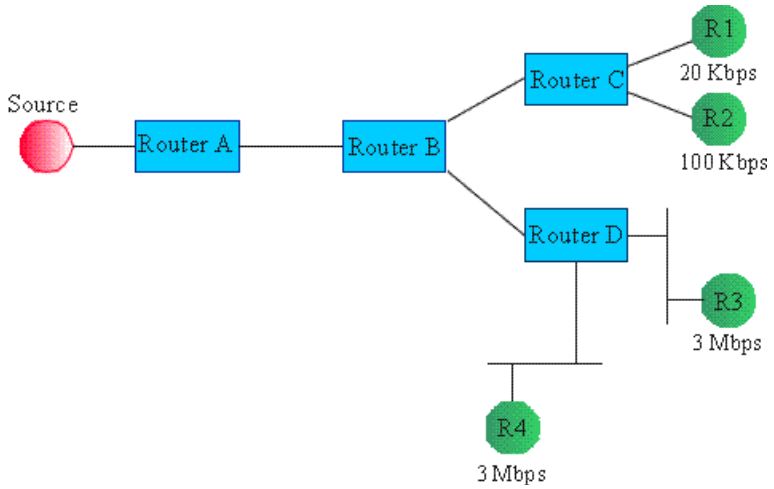

Figure 6.9-2: An RSVP example

Crudely speaking, RSVP operates as follows for this example. Each receiver sends a reservation message upstream into the multicast tree. This reservation message specifies the rate at which the receiver would like to receive the data from the source. When the reservation message reaches a router, the router adjusts its packet scheduler to accommodate the reservation. It then sends a reservation upstream. The amount of bandwidth reserved upstream from the router depends on the bandwidths reserved downstream. In the example in Figure 6.9-2, receivers R1, R2, R3 and R4 reserve 20 kbps, 120 kbps, 3 Mbps and 3 Mbps, respectively. Thus router D's downstream receivers request a maximum of 3 Mbps. For this one-to-many transmission, Router D sends a reservation message to Router B requesting that Router B reserve 3 Mbps on the link between the two routers. Note that only 3 Mbps is reserved and not 3+3=6 Mbps; this is because receivers R3 and R4 are watching the same sporting event, so there reservations may be merged. Similarly, Router C requests that Router B reserve 100 Kbps on the link between routers B and C; the layered encoding ensures that receiver R1's 20 Kbps stream is included in the 100 Mbps stream. Once Router B receives the reservation message from its downstream routers and passes the reservations to its schedulers, it sends a new reservation message to its upstream router, Router A. This message reserves 3 Mbps of bandwidth on the link from Router A to Router B, which is again the maximum of the downstream reservations.

We see from this first example that RSVP is **receiver-oriented**, i.e., the receiver of a data flow initiates and maintains the resource reservation used for that flow. Note that each router receives a reservation message from each of its downstream links in the multicast tree and sends only one reservation message into its upstream link.

As another example, suppose that four persons are participating in a video conference, as shown in Figure 6.8-3. Each person has three windows open on her computer to look at the other three persons. Suppose that the underlying routing protocol has established the multicast tree among the four hosts as shown in the diagram below. Finally, suppose each person wants to see each of the videos at 3 Mbps. Then on each of the links in this multicast tree, RSVP would reserve 9 Mbps in one direction and 3 Mbps in the other direction. Note that RSVP does not merge reservations in this example, as each person wants to receive three distinct streams.

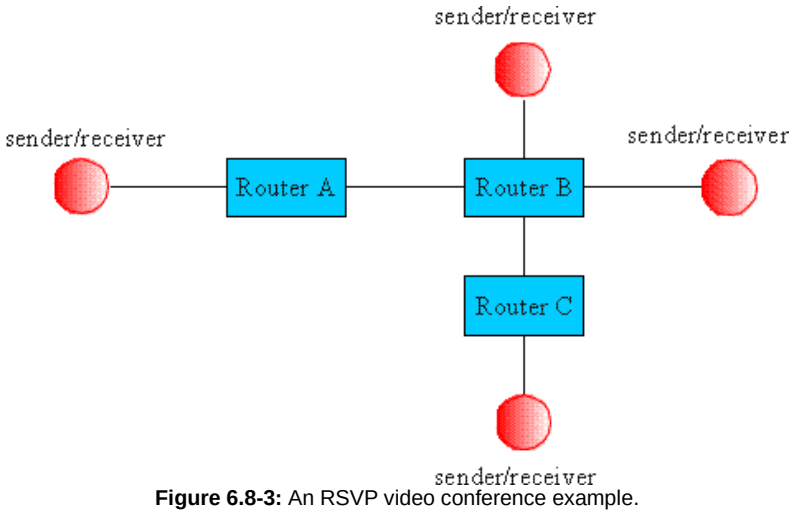

Figure 6.8-3: An RSVP video conference example.

Now consider an audio conference among the same four persons over the same multicast tree. Suppose *b* bps are needed for an isolated audio stream. Because in an audio conference it is rare that more than two persons speak at the same time, it is not necessary to reserve *3*b* bps into each receiver; *2*b* should suffice. Thus, in this last application we can conserve bandwidth by merging reservations.

Call Admission

Just as the manager of a restaurant should not accept reservations for more tables than the restaurant has, the amount of bandwidth on a link that a router reserves should not exceed the link's capacity. Thus whenever a router receives a new reservation message, it must first determine if its downstream links on the multicast tree can accommodate the reservation. This *admission test* is performed whenever a router receives a reservation message. If the admission test fails, the router rejects the reservation and returns an error message to the appropriate receiver(s).

RSVP does not define the admission test; but it assumes that the routers perform such a test and that RSVP can interact with the test.

6.8.3 Path Messages

So far we have only discussed the RSVP reservation messages, which originate at the receivers and flow upstream towards the senders. *Path messages* are another important RSVP message type; they originate at the senders and flow downstream towards the receivers.

The principle purpose of the path messages is to let the routers know on which links they should forward the reservation messages. Specifically, a path message sent within the multicast tree from a Router A to a Router B contains Router A's unicast IP address. Router B puts this address in a path-state table, and when it receives a reservation message from a downstream node it accesses the table and learns that it should send a reservation message up the multicast tree to Router A. In the future some routing protocols may supply reverse path forwarding information directly, replacing the reverse-routing function of the path state.

Along with some other information, the path messages also contain a *sender Tspec*, which defines the traffic characteristics of the data stream that the sender will generate (see Section 6.8). This Tspec can be used to prevent over reservation.

6.8.4 Reservation Styles

Through its *reservation style,* a reservation message specifies whether merging of reservations from the same session is permissible. A reservation style also specifies from which senders in a session the receiver desires to receive data. Recall that a router can identify the sender of a datagram from the datagram's source IP address.

There are currently three reservation styles defined: *wildcard-filter style; fixed-filter style; and shared-explicit style*.

Wildcard-Filter Style:

> When a receiver uses the wildcard-filter style in its reservation message, it is telling the network that it wants to receive all flows from all upstream senders in the session and that its bandwidth reservation is to be shared among the senders.

Fixed-Filter Style:

> When a receiver uses the fixed-filter style in its reservation message, it specifies a list of senders from which it wants to receive a data flow along with a bandwidth reservation for each of these senders. These reservations are distinct, i.e., they are not to be shared.

Shared-Explicit Style:

> When a receiver uses the shared-explicit style in its reservation message, it specifies a list of senders from which it wants to receive a data flow along with a single bandwidth reservation. This reservation is to be shared among all the senders in the list.

Shared reservations, created by the wildcard filter and the shared-explicit styles, are appropriate for a multicast session whose sources are unlikely to transmit simultaneously. Packetized audio is an example of an application suitable for shared reservations; because a limited number of people talk at once, each receiver might issue a wildcard-filter or a shared-explicit reservation request for twice the bandwidth required for one sender (to allow for over speaking). On the other hand, the fixed-filter reservation, which creates distinct reservations for the flows from different senders, is appropriate for video teleconferencing.

Examples of Reservation Styles

Following the Internet RFC, we now give examples for the three reservation styles. In Figure 6.8.4, a router has two incoming interfaces, labeled A and B, and two utgoing interfaces, labeled C and D. The many-to-many multicast session has three senders – S1, S2 and S3 – and three receivers – R1, R2 and R3. Figure 6.9-4 also shows that interface D is connected to a LAN.

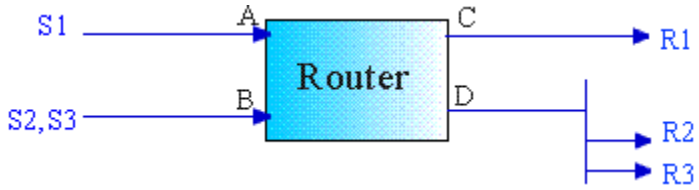

Figure 6.8-4: Sample scenario for RSVP reservation styles

Suppose first that all of the receivers use the *wildcard-filter reservation.* As shown in the Figure 689-5, receivers R1, R2, and R3 want to reserve 4b, 3b, and 2b, respectively, where b is a given bit rate. Then the router reserves 4b on interface C and 3b on interface D. Because of the wildcard-filter reservation, the two reservations from R2 and R3 are merged for interface D: the larger of the two reservations is used rather than the sum of reservations. The router then sends a reservation message upstream to interface A and another to interface B; each of these reservation messages requests is 4b, which is the larger of 3b and 4b.

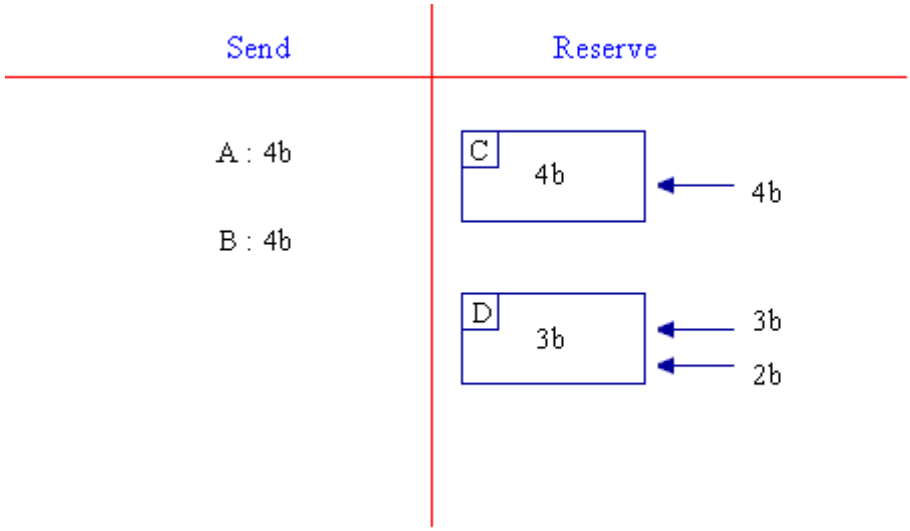

Figure 6.8-5: Wildcard filter reservations.

Now suppose that all of the receivers use the *fixed-filter reservation*. As shown in Figure 6.8-6, receiver R1 wants to reserve 4b for source S1 and 5b for source S2; also shown in the figure are the reservation requests from R2 and R3. Because of the fixed-filter style, the router reserves two disjoint chunks of bandwidth on interface C: one chunk of 4b for S1 and another chunk of 5b for S2. Similarly, the router reserves two disjoint chunks of bandwidth on interface D: one chunk of 3b for S1 (the maximum of b and 3b) and one chunk of b for S3. On interface A, the router sends a message with a reservation for S1 of 4b (the maximum of 3b and 4b). On interface B, the router sends a message with a reservation of 5b for S2 and b for S3.

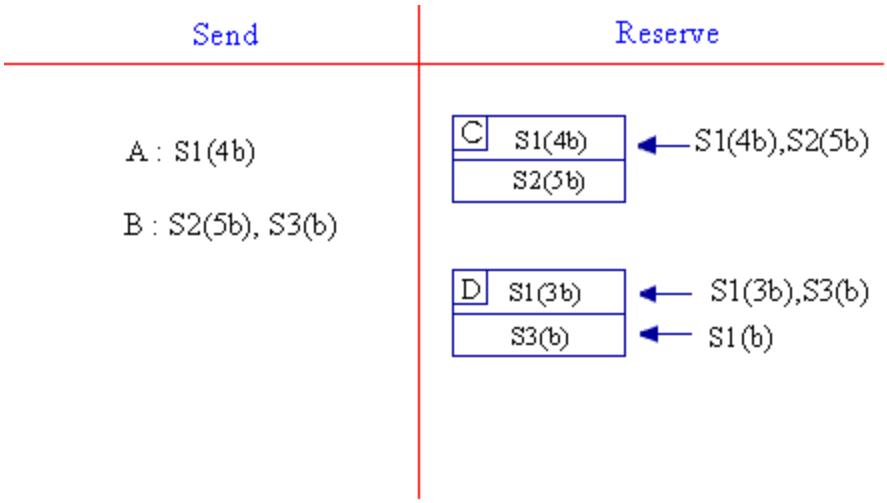

Figure 6.8-6: fixed filter reservations

Finally suppose that each of the receivers use the *shared-explict reservation*. As shown in tFigure 6.8-7, receiver R1 desires a pipe of 1b which is to be shared between sources S1 and S2, receiver R2 desires a pipe of 3b to be shared between sources S1 and S3, and receiver R3 wants a pipe of 2b for source S2. Because of the shared-explicit style, the reservations from R2 and R3 are merged for interface D: only one pipe is reserved on interface D, although it is reserved at the maximum of the reservation rates. RSVP will reserve on interface B a pipe of 3b to be shared by S2 and and S3; note that 3b is the maximum of the downstream reservations for S2 and S3.

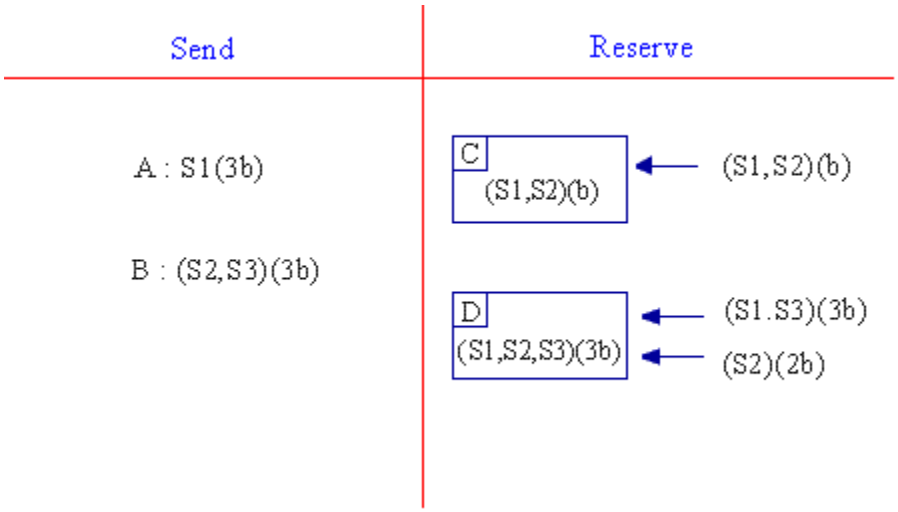

Figure 6.8-7: shared-explicit reservations

In each of the above examples the three receivers used the same reservation style. Because receivers make independent decisions, the receivers participating in a session could use different styles. RSVP does not permit, however, reservations of different styles to be merged.

6.8.5 Soft State

The reservations in the routers and hosts are maintained with *soft states*. By this it is meant that each reservation for bandwidth stored in a router has an associated timer. If a reservation's timer expires, then the reservation is removed. If a receiver desires to maintain a reservation, it must periodically refresh the reservation by sending reservation messages. A receiver can also change its reservation (e.g., the amount of bandwidth or the senders it wants to receive from) by adjusting its reservation in its stream of refresh messages.

The senders must also refresh the path state by periodically sending path messages. When a route changes, the next path message initializes the path state on the new route, and future reservation messages will establish reservation state in the route. The state on the old segments of the route will time out.

Soft state, whereby the state is maintained with refresh messages, is used by many other protocols in data networking. For example, as we learned in Chapter 5, in the routing tables in transparent bridges, the entries are refreshed by data packets that arrive to the bridge; entries that are not refreshed are timed-out. A protocol that takes explicit actions to modify or release state is called a *hard-state* protocol. An example of a hard-state protocol is TCP, whereby the connection does not timeout if it stops being used; instead one side of the connection must explicitly destroy the connection.

6.8.6 Transport of Reservation Messages

RSVP messages are sent hop-by-hop directly over IP. Thus the RSVP message is placed in the information field of the IP datagram; the protocol number in the IP datagram is set to 46. Because IP is unreliable, RSVP messages are not acknowledged upon arrival. If an RSVP path or reservation message is lost, a replacement refresh message should arrive soon.

An RSVP reservation message that originates in a host will have the host's IP address in the source address field of the encapsulating IP datagram. It will have the IP address of the first router along the reserve-path in the multicast tree in destination address in the encapsulating IP datagram . When the IP datagram arrives at the first router, the router strips off the IP fields and passes the reservation message to the router's RSVP module. The RSVP module examines the messages multicast address (i.e., session identifier) and style type, examines its current state, and then acts appropriately; for example, the RSVP module may merge the reservation with a reservation originating from another interface and then send a new reservation message to the next router upstream in the multicast tree.

Insufficient Resource

Because a reservation request that fails an admission test may embody a number of requests merged together, a reservation error must be reported to all the concerned receivers. These reservation errors are reported within **ResvError messages**. The receivers can then reduce the amount of resource that they request and try reserving again. The RSVP standard provides mechanisms to allow the backtracking of the reservations when insufficient resources are available; unfortunately, these mechanisms add significant complexity to the RSVP protocol. Furthermore, RSVP suffers from the so-called **killer-reservation problem**, whereby a receiver requests over and over again a large reservation, each time getting its reservation rejected due to lack of sufficient resources. Because this large reservation may have been merged with smaller reservations downstream, the large reservation may be excluding smaller reservations from being established. To solve this thorny problem, RSVP uses the ResvError messages to establish additional state in routers, called **blockade state**. Blockade state in a router modifies the merging procedure to omit the offending reservation from the merge, allowing a smaller request to be forwarded and established. The blockade state adds yet further complexity to the RSVP protocol and its implementation.