RSVP: General-Purpose Signaling for IP

Chris Metz • *Cisco Systems* • *chmetz@cisco.com*

Quality of Service (QoS) is one of the holy grails of data networking applications (see last issue's On the Wire, "IP QoS," Mar.-Apr. 1999, pp. 84-88). For well over a hundred years, the telephone networks have allowed users to dynamically request and receive a fixed amount of network resource for their communications. Today's Asynchronous Transfer Mode (ATM), with its sophisticated signaling and traffic management mechanisms, lets end users exchange data, voice, and video over a fixed-bandwidth, bounded-delay virtual connection (VC). TCP/IP, on the other hand, traditionally works on a different paradigm—all applications are equal—and the network provides a best-effort service on a first-come, first-served basis. Naturally, this leads to situations where, for example, a real-time video application's usefulness is greatly diminished because its packets must contend for network resources with those of a bulk file transfer.

A Different Approach

Noting this inherent limitation in traditional IP-based networks, the Internet community developed an architecture capable of supporting both best-effort and real-time application traffic flows.[1] A fundamental component of this architecture is the setup or signaling protocol, which conveys an application's resource requirements into the network. The most prominent and well-known of IP signaling protocols is the Resource Reservation Protocol (RSVP).

But prominence and the potential to deliver on a feature as significant (and alien to the IP environment) as QoS generates overzealous market interest (read hype) as well as close technical scrutiny. Indeed, this noise reached a crescendo in the mid-1990s when RSVP was touted as the IP counterpoint to ATM's clear-cut QoS superiority. It was not ATM, however, that dimmed the spotlight on RSVP, but rather, a lack of both RSVP-enabled applications and robust, scalable network machinery. Without these accoutrements, RSVP appeared to be an "emperor with no clothes," and attention soon turned toward developing and implementing more manageable, scalable class-of-service approaches that did not require explicit signaling.[2]

Subsequent events have led to a comeback of sorts for RSVP. For example, application developers now have a standard API (Winsock2) for invoking RSVP reservation setup requests. There are also techniques emerging to make RSVP more scalable, such as reducing its message overhead and state in the network, and a framework is in the works for more efficiently administering RSVP policies to a set of applications or end users. Some router implementations are even using RSVP to build and manage explicitly routed paths through a routed backbone.

Quietly, and perhaps unintentionally, RSVP has re-emerged as a general-purpose IP signaling protocol for a suite of different network applications, including QoS signaling and traffic engineering.[3]

Integrated Services

RSVP's origins can be traced back to the effort to develop an Integrated Services Architecture (IntServ) for IP networks. The objective of IntServ was to create a reference framework of components to facilitate the transmitting of best-effort and real-time traffic flows over an IP network. In the context of the IntServ model, a flow defines a stream of packets with the same source and destination addresses and port numbers. Supporting a flow with real-time properties thus involves a fundamental departure from the traditional connectionless mode of IP networking. Instead, it requires a protocol that would first check for available resources (admission control) and then reserve some amount of flow-specific state in the network.

The notion of admission control leads to two more functional requirements: a means of characterizing or quantifying the amount of network resources requested by the application, and the ability to communicate that information into the network. RSVP handles the latter function.

Figure 1 illustrates the IntServ architecture's basic components:

- *Admission control.* Performs a local accept/reject decision for the reservation request based on a comparison of the resources requested and the network resources available in the node

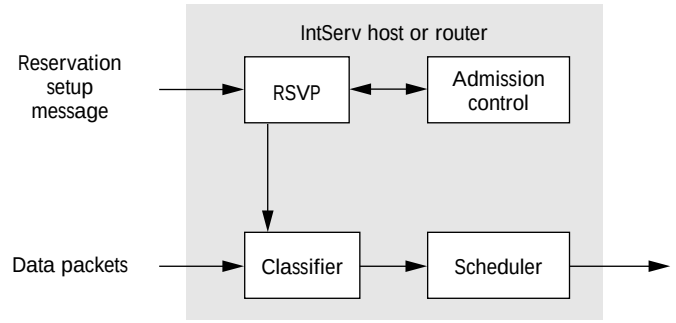

Figure 1. Basic components of the IntServ architecture. IntServ was created to help facilitate transmitting best-effort and real-time traffic flows over an IP network.

(router), along with any additional configured policies relevant to resource allocation and administrative reservation permission.
- *Packet classifier.* Inspects multiple fields in each incoming packet to determine the packet's class and, therefore, the service level to accord it.
- *Packet scheduler.* Applies one or more traffic management mechanisms, such as advanced queue scheduling (for example, Weighted Fair Queuing or WFQ), to ensure that the packet is transmitted into the network in time to satisfy the bandwidth and delay constraints of the flow.
- *RSVP.* Conveys an application's QoS requirements into the network.

Two additional elements round out the IntServ architecture.

- *Flowspec*. The flow specification is carried by RSVP messages into the network and defines an application's QoS requirements as a series of objects (such as token bucket parameters).[4]
- *Traffic classes.* The IntServ architecture currently supports two traffic classes in addition to best-effort: g*uaranteed service*, or GS, supports applications that require absolute, quantifiable bounds on network delay,[5] for example, real-time multimedia applications; c*ontrolled load*, or CL, is a more qualitative service that emulates a best-effort service operating in a congestion-free network.[6]

Classical RSVP

RSVP has often been incorrectly characterized as a routing protocol or a transport protocol, but it is neither. The term "classical RSVP" applies to version 1 of the protocol, as defined in RFC 2205, and is used by host applications to signal their QoS requirements into the network.[7] As a signaling protocol, RSVP v1 was designed with the following functional attributes[8]:

- Supports both unicast and multicast data flows.
- Establishes receiver-initiated reservations. Unlike the traditional telephone or ATM model where the caller (sender) signals the connection and QoS request to the callee (receiver), in RSVP the receiver of the data flow initiates the reservation request.
- Establishes reservation state in one direction. In other words, the exchange of RSVP messages only reserves resources for data flowing between the sender and the receiver.
- Maintains independence from the existence and function of current unicast and multicast routing protocols.
- Transports traffic, QoS, and policy objects opaquely. RSVP has no idea what is inside the objects that it transports and deposits in RSVP-capable hosts and routers in the network. The definition of the objects transported by RSVP is handled by various working groups depending on the service and state installed in the network.
- Maintains soft-state operation. RSVP reservations must be periodically refreshed, or they will time out. This lends a degree of flexibility for adapting to changes in network capacity or topology.

Figure 2 shows the basic exchange of RSVP messages to reserve resources in the network. An application on a sending host initiates an RSVP flow, and the RSVP component uses information conveyed from the application over an API to construct a series of objects. The important objects to note here are the Sender Template that identifies the IP address and port number of the sending host, and the Sender Tspec that describes the characteristics of the data traffic generated by the sending application. The host places the objects inside a Path message it transmits to the unicast host address or the multicast group address of the receiver(s).

Path to the receiver. The Path message follows the path computed by the dynamic IP routing protocol (such as, Open Shortest Path First or OSPF) that runs on each router. At each hop, the router processes the RSVP message by installing state about the pending reservation request and records the IP address of the upstream router. The succession of these IP addresses, or PHOP (for previous hop) objects, serves as a "trail of bread crumbs" that the RSVP reservation request will use to find its way back upstream to the sender over the same path.

In addition, the router may update an ADSPEC object contained in the Path message. RSVP uses the ADSPEC object to summarize the path's characteristics and to deliver this information to the receiver who can then make a more informed decision about how much bandwidth and delay to ask for. This technique of "advertising" the QoS viability of a path as the Path message travels to the destination is known as One Path With Advertising (OPWA).[9]

Reservation. The Path message provides the receiver of the data flow with the identity and traffic characteristics of the sending application, the

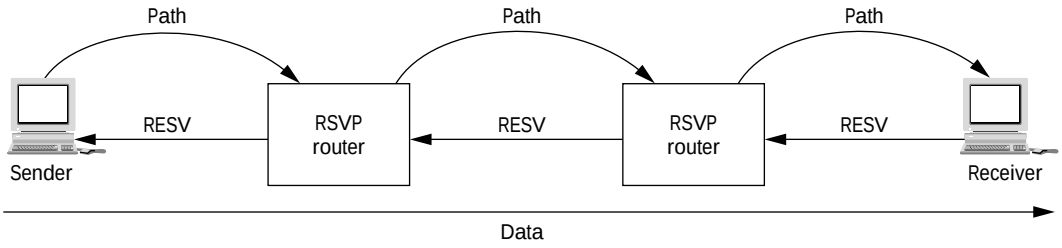

Figure 2. The basic exchange of RSVP messages to reserve resources in the network. The host transmits the Path message to the unicast or multicast group address(es) of the receiver(s), and the RESV message follows the same path in the upstream (reverse) direction.

characteristics of a routed path from the sender to the receiver, and a trail to follow back to the sender. With this information in hand, the application signals the RSVP component to generate an RSVP RESV message containing a flow descriptor that can be broken down into a filterspec and a flowspec. The filterspec identifies the source address and port number of the sending application, and the flowspec contains the traffic and QoS information that the receiver application is requesting. The classifier is updated with information contained in the filterspec, and the scheduler is updated with information contained in the flowspec.

The RESV message flows upstream in a hop-by-hop fashion all the way to the host and deposits its contents in the traffic control component of each router along the path. If traffic control determines that there are sufficient resources to satisfy the request, the appropriate resources are allocated on the router to support the data traffic flowing from the sender to the receiver. (It should be noted that RSVP does not in any way handle or support data traffic. The packet classifier and scheduler provide the QoS support for the data flow.)

Soft-state exchange. The soft-state nature of RSVP is illustrated by the fact that the Path/RESV message exchange is performed under a number of static and dynamic conditions. If a previous reservation request fails, the receiver application may reissue a RESV message with a different flow descriptor. If dynamic IP routing computes a new path due to a change in topology, the RSVP message exchange must occur over that new path.

In addition, RSVP must generate periodic Path and RESV messages to refresh the reservation state in the network, or it will time out. And finally, note that data traffic can always flow between the sender and the receiver regardless of whether reservations exist in the network. If no reservations exist, then the packets are serviced in a best-effort manner.

Reservation Styles

To make best use of reserved state in the network and to provide applications a degree of control over it, RSVP supports different reservation styles. One style specifies whether a reservation should be dedicated for a particular sender or shared among multiple senders in the same session (an RSVP session is defined as packets sharing the same destination IP address and port number). Another identifies one or more senders permitted to use the reservation.

For example, a fixed-filter (FF) reservation style indicates that a single reservation will be allocated for a single sender, which might be useful for a videoconference application requiring minimum bandwidth and tight bounds on delay. The shared-explicit (SE) style enables multiple senders in the same session to share a reservation. For example, 10 people may participate in a packet-based audio conference call, but only one can talk at a time. Instead of reserving 10 separate reservations of 64 Kbytes of bandwidth each, a single 64-Kbyte reservation is allocated and shared by all senders (participants) and only used by the person currently speaking.

To conserve the amount of state signaled and maintained by the routers in the network, RSVP allows reservations to merge at branch points in a multicast tree. This means that the router will install the largest of all received reservation requests for a sender or set of senders and then pass that merged reservation value (in the form of a new flowspec) to the next upstream router.

Merging requires the router to perform admission control on the result of the merged flowspec, and this can lead to some interesting scenarios with equally interesting names. *Killer reservations* occur when a new reservation (call it R2) prevents either a currently installed reservation or another reservation request (call it R1) where R1 < R2 to be denied service. The solution to the first problem (termed KR-I) is to ignore R2 if it impacts a currently installed reservation. The solution to the second problem (termed KR-II) is for the routers to enter into a "blockade state" where R2 is ignored while allowing R1 to proceed.

Improving RSVP Scalability

One criticism of RSVP v1 was that it imposed per-flow state on every router in the network. This could certainly introduce scaling problems for routers, particularly for those in the core of a large network where flow density tends to be quite high. For example, a core router providing transport service for thousands or tens of thousands of individual RSVP flows would have to maintain per-flow state as well as handle the periodic refresh messages for each one of those flows. There are certainly scenarios where an application would

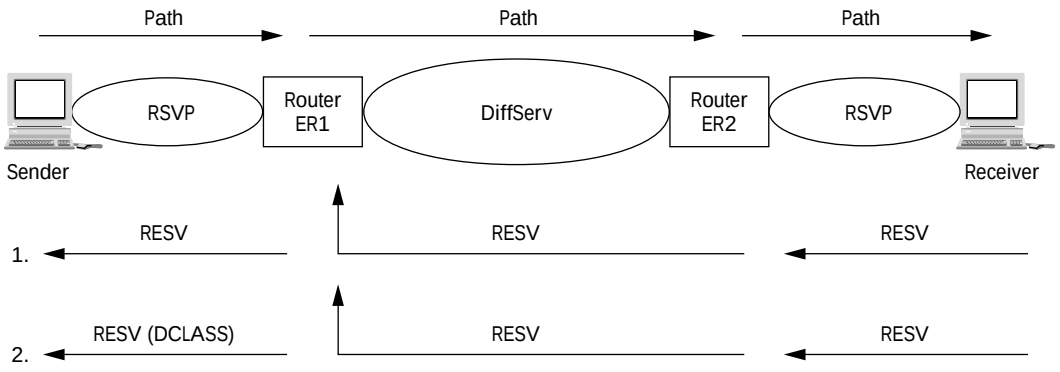

Figure 3. Two methods for implementing RSVP signaling to place application flows in the appropriate DiffServ aggregated flow. Two separate regions of RSVP-capable hosts and routers are connected to a DiffServ transit domain of routers.

want to explicitly convey its quantitative QoS requirements into the network, but it must do so without burdening the network with excess state and message overhead.

Diffserv model. One solution to this problem would be to allow RSVP signaling to interoperate with transit routers supporting the Differentiated Services (DiffServ) model.[10] Diffserv enables the individual packets of an application traffic flow to be classified (using a multifield (MF) classifier) and then marked with one or more DiffServ codepoints (DSCP) at the edge of the network. The DSCP bits indicate that the packet belongs to one of a few aggregated traffic flows. The interior of the network is then provisioned to support these aggregated traffic flows with services (termed per-hop behaviors) that describe how to treat the packet based on the DSCPs in the packet header.

The benefit of the DiffServ approach is that it does not require explicit signaling or per-flow state to be maintained in the network.

RSVP cooperation. Figure 3 illustrates two different ways RSVP signaling could be implemented to essentially place application flows in the appropriate DiffServ aggregated flow to ensure that applications receive their requested QoS. Two separate regions of RSVP-capable hosts and routers are shown connected to a DiffServ transit domain of routers. In both cases, the sender and receiver exchange RSVP Path and RESV messages, and normal processing occurs for all routers that support RSVP. The DiffServ routers ignore the messages since they do not understand RSVP.

In the second case, the ingress router (ER1) may use information contained in the RESV message to dynamically install an MF classifier for the RSVP flow. When data packets arrive at the ingress Diffserv router from the sender, the appropriate DSCP bits will be set, and the packets of the flow will receive the appropriate service across the DiffServ transit domain.

In the second case, the ER1 uses the RESV message to communicate the appropriate DSCP bit back (contained in a DCLASS object) to the sending host. The sender marks all subsequent data packets with the DSCP bit indicating the service that the packet should be accorded in the network. It may even be possible for the ER1 to redirect the Path message to a nearby policy server that will, in turn, generate the RESV message back to the sender containing a new DSCP bit. The clear advantage of these approaches is that per-flow state is offloaded to the edge of the network, while RSVP-capable hosts can still request and receive QoS from the network.

New APIs

Applications must declare their QoS characteristics and requirements over an API to RSVP, but until recently there has not been a standard API that allowed applications to invoke RSVP functions. This has contributed to the reluctance of application developers to write RSVP-based applications. Winsock2 will change all this.

Winsock2 is a new Windows-based networking API that supports several new features including multicast, enhanced multiprotocol standards, and QoS. Winsock2 defines data structures and calls that enable applications to signal their requirements through a variety of techniques including RSVP and ATM.

More currently, Microsoft is completing work on a Generic QoS (GQoS) API for Winsock2 that should make it easier for applications to support RSVP. Furthermore, Windows 98, NT 5.0, and Windows 2000 will all support the GQoS API, which should certainly encourage application developers to write more RSVP-based applications.

Policy

Networks that support RSVP's QoS signaling mechanisms will be allocating network resources from a finite set to service a particular application traffic flow. Before admitting the flow and allocating the resources contained in the RSVP request, the network may want to verify that the request conforms to the resource allocation policies of the network at large. Failure to do so could result in theft-of-service or denial-of-service scenarios.

The IETF's RSVP Policy working group has developed a framework to exert policy control during the admission control phase of the RSVP setup process.[11] In this scenario, RSVP messages carry an additional policy object that is passed to a policy decision

point (PDP). The role of the PDP is to decide if the RSVP process should continue and then to communicate that information down to a policy enforcement point (PEP), which is responsible for executing the policy. The protocol that enables a PDP and PEP to communicate with each other is called the Common Open Policy Server (COPS).

Traffic Engineering

Until recently, RSVP was presented and defined as a host-based QoS signaling protocol. RSVP must also run on routers, however, and its object transparency allows a significant degree of extensibility. Thus it presented an attractive solution for enabling routers to signal the establishment of a dedicated path through a set of explicitly defined nodes in a network.[12] This ability to provision nondefault paths that override dynamic IP routing is called Traffic Engineering. TE is a useful mechanism for large ISPs that need to direct traffic away from congestion or that wish to balance traffic across multiple paths in the network.

The components of TE solutions include extensions to existing link-state routing protocols such as OSPF, Multiprotocol Label Switching (MPLS) and RSVP operating on routers with only slight modifications. In this scenario, the routing protocols propagate information about the available capacity in the network. The ingress router kicks off an RSVP Path message containing an Explicit Route Object (ERO) that defines the set of nodes that the Path should traverse. The RESV message generated by the egress router flows back toward the ingress depositing labels in each hop along the way. The result is an ingress-to-egress MPLS Label-Switched Path (LSP) capable of carrying aggregate traffic flows between edge routers over a backbone network. Routers from Cisco, Juniper, and Torrent (now Ericcson) currently or will soon support the RSVP and MPLS approach for traffic engineering.

Interest in RSVP continues to grow and can be best expressed by the growing number of RSVP-capable hardware and software components that vendors are shipping.[13] To evolve in both efficiency and effectiveness, IP-based networks are being engineered to perform special tasks. RSVP, as it turns out, is proving to be an effective general-purpose signaling protocol for requesting and allocating resources for these special tasks. ■

Chris Metz is a Consulting Systems Engineer for Cisco Systems. His current areas of interest include IP service differentiation, multicast, high-performance routing, and IP/ATM integration. He is the author of *IP Switching: Protocols and Architectures*, McGraw-Hill, 1999.

REFERENCES

1. R. Braden, D. Clark, and S. Shenker, "Integrated Services in the Internet Architecture: an Overview," RFC 1633, Network Working Group, June 1994; available at http://www.rfc-editor.org/rfc/rfc1633.txt.
2. Differentiated Services (DiffServ) Charter http://www.ietf.org/html.charters/diffserv-charter.html.
3. M. Ni and X. Xiao, "Internet QoS: A Big Picture," *IEEE Network*, Vol. 13, No. 2, Mar.-Apr. 1999, pp. 8-18.
4. C. Partridge, "A Proposed Flow Specification," RFC 1363, Network Working Group, Sept. 1992; available at http://www.rfc-editor.org/rfc/rfc1363.txt.
5. S. Shenker and C. Partridge, "Specification of Guaranteed Quality of Service," RFC 2212, Network Working Group, Sept. 1997; available at ftp://ftp.isi.edu/in-notes/rfc2212.txt.
6. J. Wroclawski, "Specification of the Controlled-Load Network Element Service," RFC 2211, Network Working Group, Sept 1997; available at ftp://ftp.isi.edu/in-notes/rfc2211.txt.
7. P. White, "RSVP and Integrated Services in the Internet: A Tutorial," *IEEE Comm.*, Vol. 35, No. 5, May 1997, pp. 100-106.
8. L. Zhang et al., "RSVP: A New Reservation Setup Protocol," *IEEE Network*, Sept. 1993; available at ftp://parcftp.xerox.com/pub/net-research/rsvp.ps.Z.
9. S. Shenker and L. Breslau, "Two Issues in Reservation Establishment," *Proc. ACM SIGComm 95*, ACM Press, New York, 1995.
10. S. Blake et al. "An Architecture for Differentiated Services," RFC 2475, Network Working Group, Dec. 1998; available at ftp://ftp.isi.edu/in-notes/rfc2475.txt.
11. RSVP Admission Policy (RAP) Charter, http://www.ietf.org/html.charters/rapcharter.html.
12. T. Li and Y. Rekhter, "A Provider Architecture for Differentiated Services and Traffic Engineering (PASTE)," RFC 2430, Network Working Group, Oct 1998; available at ftp://ftp.isi.edu/in-notes/rfc2430.txt.
13. G. Gaines and M. Festa, "A Survey of RSVP/QoS Implementations, Update 2," RSVP Working Group, July 1998; available at http://www.iit.nrc.ca/IETF/RSVP_survey/ietf_rsvp-qos_survey_02.txt.

RSVP-Related Resources

RSVP ReSerVation Protocol Project Home Page • http://www.isi.edu/rsvp/
Intel's PC-RSVP Page • http://intel.com/ial/rsvp/
WinSock 2 Information • http://www.sockets.com/winsock2.htm
Winsock2 Generic QoS Mapping Spec (Draft) • ftp://ftp.microsoft.com/bussys/winsock/winsock2/gqos_spec.doc
Juniper Traffic Engineering White Paper • http://www.juniper.net/leadingedge/whitepapers/TE_NPM.html
Tel-Aviv University RSVP Tutorial • http://www.rad.com/networks/1997/rsvp/rsvp_welcome.html
For authoritative information on Internet traffic engineering, see **Daniel Awduche's page** at http://www.awduche.com/