Journal of Computer Science 3 (8): 658-665, 2007
ISSN 1549-3636
© 2007 Science Publications

An Improved Resource Reservation Protocol

Desire Oulai, Steven Chamberland and Samuel Pierre
Department of Computer Engineering Ecole Polytechnique de Montreal
P.O. Box 6079, Station Centre-Ville, Montreal (Quebec), Canada H3C 3A7

Abstract: The classical resource reservation protocol (RSVP) is a flow-based signaling protocol used for reserving resources in the network for a given session. RSVP maintains state information for each reservation at every router along the path. Even though this protocol is very popular, he has some weaknesses. Indeed, RSVP does not include a bidirectional reservation process and it requires refresh messages to maintain the soft states in the routers for each session. In this paper, we propose a sender-oriented version of RSVP that can reserve the resources in both directions with only one message, thus reducing the delay for establishing the reservations. We also suggest a refreshment mechanism without any refresh message which could be applied to any soft states protocol. Simulation results show that the proposed protocol is approximately twice faster than RSVPv2 for establishing bidirectional reservations with almost no control overhead during the session.

Key words: Internet protocol (IP), quality of service (QoS), resource reservation protocol (RSVP), setup time, signaling load, soft state refreshment, failure scenarios.

INTRODUCTION

Nowadays, many applications such as voice and video applications have stringent quality of service (QoS) requirements. However, the Internet protocol (IP), which is the most used protocol at the network layer, lacks of QoS mechanisms. That is why many QoS architectures and protocols have been proposed, for instance, integrated services (IntServ), differentiated services (DiffServ) and multiprotocol label switching (MPLS) (see Ni and Xiao[9] for details concerning QoS in IP networks).

The classical resource reservation protocol (RSVP), such as defined in RFC 2205[5], is a flow-based resource reservation protocol. It has been originally created to work with the IntServ [4] architecture. Basically, IntServ implements three classes of services: guaranteed service (GS), controlled load (CL) and best effort (BE). GS guarantees maximal delay and bandwidth. Controlled load offers a QoS similar to what can be offered in a network without congestion and BE does not offer any guarantee.

RSVP allows signaling for multicast and unicast sessions. For instance, when Router A wants to establish a session with Router B, it sends a PATH message to Router B. The PATH message collects information about the resources on the routers on the path used from Router A to Router B and it also establishes a PATH state in each router. The PATH states also acts as a passive reservation with no physical reservation. When Router B receives the PATH message, it responds with a RESV message that is routed along the reverse path taken by the PATH message. The RESV message reserves the resources by creating a RESV state in each router. Such a protocol is called a receiver-oriented protocol. The RESV message physically reserves the resources that are available firstly by the flow for which the reservation has been requested. The PATH and RESV states are soft states, i.e., they will be deleted after a certain timeout if there is no refreshment. Refresh messages are scheduled each $I \in [0.5R, 1.5R]$ in order to avoid synchronization between routers (where R is a common predefined value for all the routers within the RSVP domain).

Refresh messages are also useful for path restoration in failure scenarios. When a failure occurs, the PATH refresh message will be sent through the network to pass around the failed network elements and reach the destination. For a faster restoration, RSVP implements the local repair process[6]. With this process, if a failure occurs on a link adjacent to a router, its RSVP module sends a PATH refresh message to restore the segment of the path passing through the failed link. As a result, the overall path may not be optimal.

RSVP has two main drawbacks. First, RSVP does not support bidirectional reservations.

Corresponding Author: Steven Chamberland, Department of Computer Engineering Ecole Polytechnique de Montreal P.O. Box 6079, Station Centre-Ville, Montreal (Quebec), Canada H3C 3A7

In fact, two unidirectional reservations can be used for bidirectional reservations. However, this implies more delays and if we have to recover from a failure or to signal new resources after a handover in a mobile network, these delays could be a problem. The second drawback is that refresh messages should be sent periodically.

In this paper, we propose RSVP+, an improved RSVP with bidirectional signaling and states refresh mechanisms without refresh messages.

The paper is organized as follows. Section 2 presents related works followed by the presentation of the protocol RSVP+ in Section 3. In Section 4, we present and analyze the simulation results and, finally, conclusions and further works are presented in Section 5.

RELATED WORKS

Over recent years, many improvements have been proposed for RSVP. RSVP-TE (Traffic Engineering), typically used with MPLS [3], distributes the labels to the routers and allows the sender to explicitly choose the path for a given session. With RSVP-TE, when Router A wants to initiate a bidirectional reservation to Router B, it sends a PATH message which distributes labels in downstream direction and other labels are distributed by the RESV in the reverse direction. RSVPv2[10] used almost the same principle but does not work with MPLS. Abondo and Pierre[1] introduced a sender-oriented RSVP for fast signaling. The reservation is done by the PATH messages. RFC 2961 [2] proposes to bundle the refresh messages between two adjacent routers. The bundles many refresh messages and sends them in one packet, thus reducing the amount of overhead needed for the refresh messages. There is a gain on the IP overhead but the RSVP messages are unchanged. Moreover, RFC 2961[2] introduces a Summary Refresh message (SRefresh). In the SRefresh message, there is an object MESSAGE ID containing an identifier, called MESSAGE IDENTIFIER, used to identify the PATH messages. As a result, the routers do not need all the content of the refresh messages because they are able to identify the session and update the timers. These identifiers can be aggregated in one packet. These proposals reduce the size of the refresh messages, but one problem remains if one or more refresh messages are lost on their ways. The RFC 2961[2] suggests a MESSAGE ID ACK to acknowledge the messages, but with an increase in signaling load.

Presently, the next steps in signaling (NSIS) working group of the IETF works on protocols for signaling information about a data flow along its path in the network. A framework for NSIS signaling has been proposed in the RFC 4080[7]. The NSIS framework is composed of two layers. The upper layer, called NSLP (NSIS Signaling Layer Protocol), includes the signaling applications and the lower layer, called NTLP (NSIS Transport Layer Protocol), provides a generic transport service for those applications. Due to its generic framework, NSIS runs over many transport protocols.

NSIS introduces a session identifier which is different from the flow identifier. One session can carry information related to several signaling applications. The session identifier allows NSIS to support mobility easily. If a flow identifier changes along the path, the flow can still be associated with the signaling session. It is assumed that applications controlled by a single session could perform bandwidth sharing[8]. IntServ and DiffServ could be used with NSIS.

In the case of signaling for QoS, NSIS has many features. First, NSIS QoS allows sender-initiated and receiver-initiated reservations. For bidirectional reservations, a common message can reserve resources on both directions if the physical path is the same. Otherwise, two unidirectional processes have to be initiated. For scalability, NSIS QoS uses a soft state mechanism and the refresh is done with a reduced message containing a session ID as in RFC 2961[2]. However, the lost of refresh messages is still a problem.

THE NEW VERSION OF RSVP: RSVP+

RSVP+ is an improved RSVP with new features. First of all, because most of the Internet communications are unicast, RSVP+ is defined for unicast sessions although some ideas might work with multicast sessions. By this way, RSVP+ is less complex than RSVP. Furthermore, RSVP+ is a sender-oriented reservation protocol, i.e., the reservation is done by the PATH message. It is supposed that the sender is aware of the requested QoS.

Resources Reservation: When Router A wants to initiate a QoS session with Router B, it sends a PATH message as in RSVP. The PATH message creates the PATH and the RESV states on the router along the path with the information contained in the session object. The PATH message can reserve the resources on the direction from Router A to Router B, Router B to Router A, or both. The direction can be marked in the first two bits of the Flags field in the RSVP Common Header which is currently unused. For instance,

- 00: for the normal direction only (from Router A to Router B);
- 01: for the reverse direction only (from Router B to Router A);
- 10: for both directions.

When Router B receives the PATH message, the

QoS is already configured on the path and can be used. Next, Router B sends a RESV message to Router A as a confirmation.

Bidirectional Reservation: The bidirectional reservation can be done in two ways: both directions take the same physical path or not.

Same Path: The sender sends one PATH message that reserves the resources on both directions. If we want to have a receiver-oriented protocol, the PATH message works as in RSVP but collects information for both directions and the RESV message is used to reserve the resources. With this approach, the number of signaling messages for establishing the bidirectional reservation is divided by two when using the same physical path.

Different Paths: The sender sends simultaneously two PATH messages, one for each direction. When the receiver receives both PATH messages, it sends a RESV message to confirm that the reservations have been successful. If one or the two reservations have failed, the RESV message can initiate new reservations. The main difference between our process and NSIS QoS is that we allow the sender to reserve resources for the receiver without receiving a query request. As a result, the process is faster. We can also have a receiver-oriented protocol by utilizing the RESV to reserve the resources.

In the sender-oriented case, the QoS is configured along the paths as soon as the PATH messages are received. Even if there are similarities with RSVP-TE and NSIS QoS, the main difference is that RSVP+ allows a PATH message to reserve resources in every direction. There is no need to wait for a query message before launching the reservation.

States Refresh: We propose a new and simple way to refresh the states in a router. Our proposal is based on the fact that the refresh messages are most of the time copies of the original messages. They are useful when a new path has to be signaled due to a network failure or when the specifications of the reservation have changed. In the former case, the local repair process in RSVP is enough to signal the new path[5]. For the latter case, we can notice that the modification of a reservation does not happen frequently.

Based on these observations, we can say that the best way to know if a session is still working is to receive packets associated to this session. Data traffic is most of the time enough to refresh the sessions. Our proposal is as follows. When a router receives a data packet of a given flow, the router can update the expiration time associated to the states of the flow. The router can update the expiration time for each packet or after a certain amount of time since the last update. In this case, the states will not expire while packets are flowing. If there is no traffic but the sender wants to keep the reservation, it sends a classical PATH refresh message along the path. This message is forwarded with high priority along the path to maintain the states. If a state expires in a router, it sends a PathTear message to delete the reservations along the path.

With this solution, the need for explicitly refresh messages may be reduced to nothing, improving the bandwidth usage. Furthermore, a refresh message could be lost while the session is still active. With refreshment done by the data packets, each packet acts as a refresh message so the probability to lose refresh information is consequently reduced. Our refresh process is more robust and the idea can be used to any soft state protocol.

Session Blocking Reduction: In sender-oriented reservation protocols, a potential problem is that an uncompleted session blocks another one. For instance, suppose that Sender #1 sends a PATH message to Receiver #1 and, after a few seconds, Sender #2 wants to reserve resources to Receiver #2 but the request is refused due to the first reservation (see Fig. 1).

We propose a solution to reduce this kind of blocking. First, the following notation is used. Let C be the capacity of the interface (port), C_c, the reserved capacity that has been confirmed and cannot be used by a new request, C_u, the unconfirmed but reserved capacity and cannot be used by a new request, C_r, the residual capacity that is not reserved and, finally, A, the capacity requested on the interface. First note that $C = C_c + C_u + C_r$.

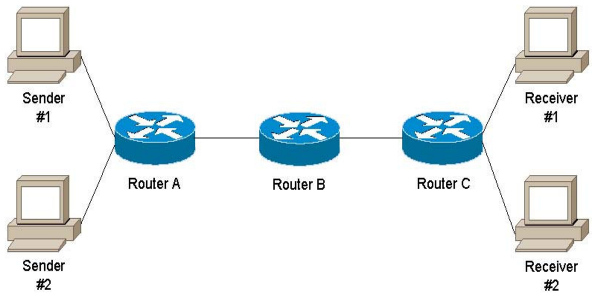

Fig. 1: Session blocking example

When a request reaches an interface, there are three cases.

Case 1: $A < C_r$

In that case, the request is accepted. The router reserves the capacity and forwards the request to the next router.

Case 2: $A > C - C_c$

The request is rejected due to insufficient capacity.

Case 3: $C_r < A < C_r + C_u$

In that case, our proposition follows. First, the router puts the request on hold for a timeout. The router can keep a waiting list ordered by priority class or arrival time. If during this timeout, enough unconfirmed resources are freed, the ongoing reservation will be accepted and forwarded to the next hop. Otherwise, the request is rejected. With this process, the setup time may increase but the blocking due to uncompleted reservation is reduced.

PERFORMANCE ANALYSIS

In this section, we assess the performance of RSVP+. The metrics chosen are the time to establish a reservation, the restoration time when a failure occurs and the amount of signaling messages. We compare RSVP+ to the classical RSVP for unidirectional reservations because RSVP and RSVP-TE have the same procedure for such reservations. For bidirectional reservations, we compare RSVP to RSVPv2 because RSVPv2 has better performance than classical RSVP for such reservations [10]. It is important to mention that we choose RSVPv2 instead of RSVP-TE because the latter works with MPLS and involves label distribution. However, RSVPv2 and RSVP-TE have the same procedure for bidirectional reservations except

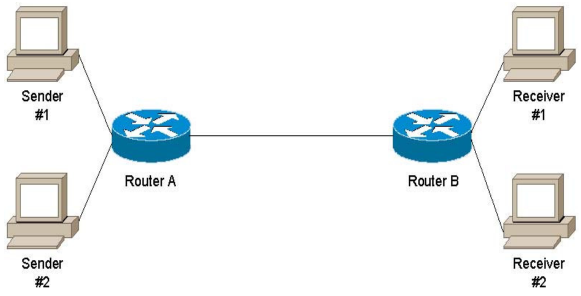

Fig. 2: Test network used to analyze the setup time

that in RSVPv2, the downstream router could not transmit after receiving the PATH messages even if resources are reserved. As we are concerned with the time for establishing the reservation, this detail is not important for us. We did not choose NSIS because some processes are similar to RSVPv2 and is still in development.

Setup Time Reduction: Let T_e be the establishment time of a given reservation, T_t, the transfer time from the sender to the receiver and T_c, the configuration time of every router along the path. The following equations can be written down for unidirectional and bidirectional reservations.

Unidirectional Reservation

RSVP : $T_e = T_c + 2\ T_t$

RSVP+: $T_e = T_c + T_t$.

Bidirectional Reservation

RSVPv2: $T_e = 2\ T_c + 2\ T_t$

RSVP+ : $T_e = 2\ T_c + T_t$ (for one PATH message),

$T_e = T_c + T_t$ (for two PATH messages).

The simulator OPNET 11.0 is used to run our simulations. The first test is to compare the setup time between RSVP and RSVP+. Fig. 2 shows the test network that is composed of two senders and two receivers who are communicating through a bidirectional service of 96 kbps. Sender #1 is communicating with Receiver #1 and Sender #2 with Receiver #2. We used three types of links: 256, 512 and 1024 kbps.

Fig. 3 shows that RSVP+ has a better setup time than RSVP for unidirectional reservations. For bidirectional reservations, we considered the case where both sessions use the same physical path. If both sessions are not required to use the same path, the results for RSVP+ for bidirectional and unidirectional reservations should be the same since both reservations are launched simultaneously. Fig. 4 shows that RSVP+ has a better setup time than RSVP for bidirectional reservations. It can also be observed that RSVP+ is approximately twice faster than RSVPv2 for establishing bidirectional reservations. This can be explained by the fact that the router configuration time, T_c, is very small. As a result, $T_e \approx 2\ T_t$ *for RSVPv2,* $T_e \approx T_t$ for RSVP+ and T_e (RSVP+) $\approx 0.5\ T_e$ (RSVPv2).

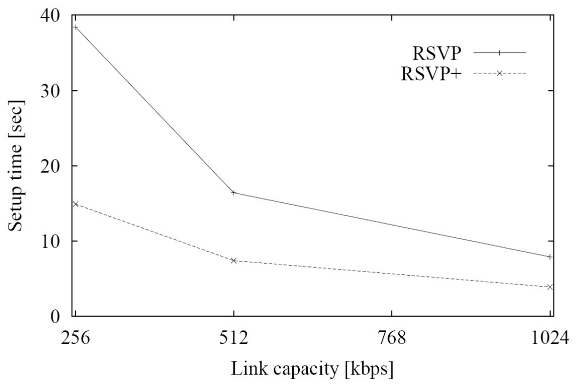

Fig. 3: Setup time as a function of the link capacity for unidirectional reservation

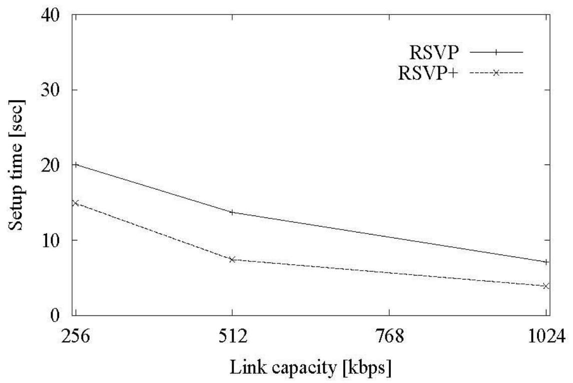

Fig. 4: Setup time as a function of the link capacity for bidirectional reservation

Restoration Time in Failure Scenarios: Failure recovery is an important issue in networks. The recovery time has to be short in order to quickly restore the QoS for the affected flows. In our experiment, we use two senders and two receivers with the same 96 kbps application. The test network is presented in Fig. 5.

Sender #1 is communicating with Receiver #1 using RSVP and Sender #2 is communicating with Receiver #2 without RSVP. In the no failure scenario, the flows use the path A-C-B. After 100 seconds, we simulate the failure of Router C. Thus, the flows are rerouted through the path A-D-B. We use 128 kbps links to put emphasis on congestion. Our objective is to determine if our algorithm is able to quickly restore the QoS on the alternate path. We use RSVP, RSVPv2 and RSVP+ for the simulation. The results are illustrated in Fig. 6.

Fig. 6 shows that the QoS restoration time is smaller for RSVP+. This can be explained by the fact that when a failure occurs, the traffics are forwarded through the alternate path in best effort until the QoS is restored. Since RSVP+ configures the QoS with the PATH message, the QoS is restored faster. So when the path message joins the receiver, the entire alternate path can provide the QoS end-to-end.

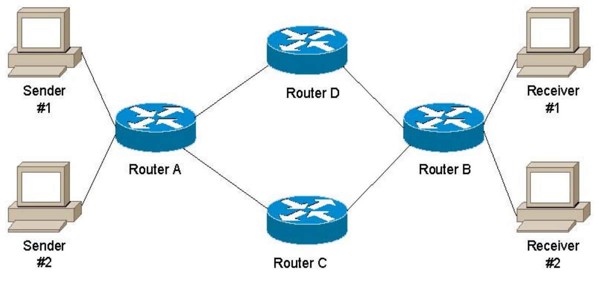

Fig. 5: Test network used to analyze the restoration time in failure scenarios

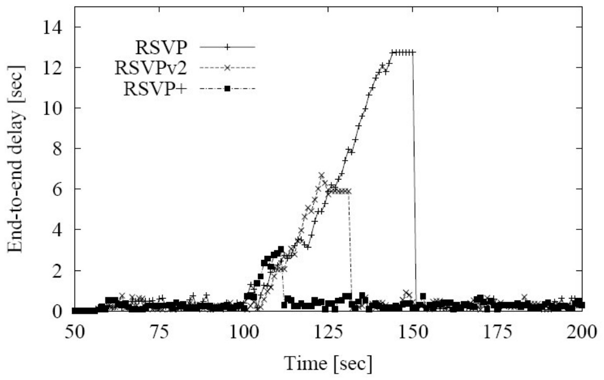

Fig. 6: QoS (end-to-end delay) restoration time in a failure scenario for RSVP, RSVPv2 and RSVP+

Refresh Reduction: In this section, we want to assess the proposed refresh procedure. The test network, presented in Fig. 7, has seven senders and seven receivers who are communicating through a bidirectional service of 96 kbps. The link capacity used is 1500 kbps and the simulation time is 600 seconds.

Fig. 8 shows that the states are being refreshed when using RSVP+. For each RSVP packet sent or received, we checked the number of active RESV states in the router. (Note that the results for PATH states are similar.)

As mentioned before, the protocol RSVP+ is implemented such that the states are refreshed for every packet associated to a flow. This approach does not have a measurable impact on the CPU utilization of the routers (see Fig. 9).

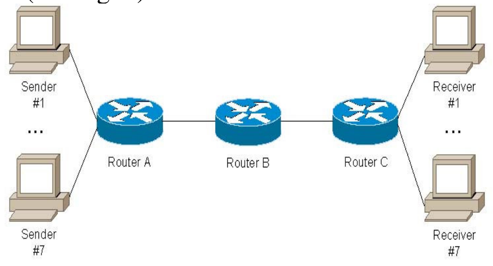

Fig. 7: Test network used to analyze the refresh load reduction

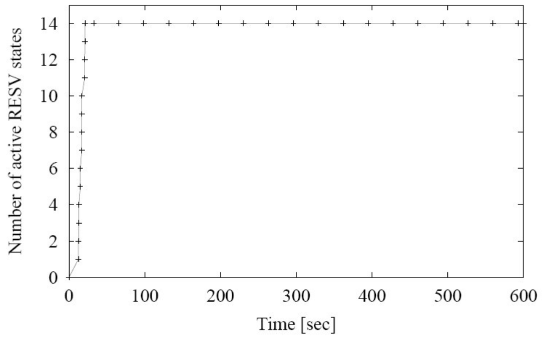

Fig. 8: Number of active RESV in Router B for RSVP+

Fig. 10 illustrates the number of refresh messages (PATH and RESV) sent by Router B for RSVP, RSVPv2 and RSVP+. The refresh interval used I is 32.92 seconds and the simulation time is 600 seconds (i.e., 18 refresh cycles).

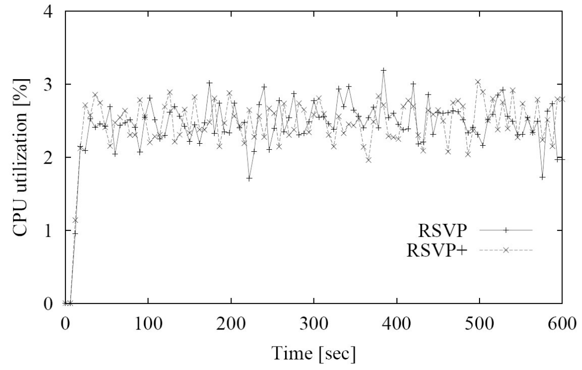

Fig. 9: CPU utilization of Router B for RSVP and RSVP+

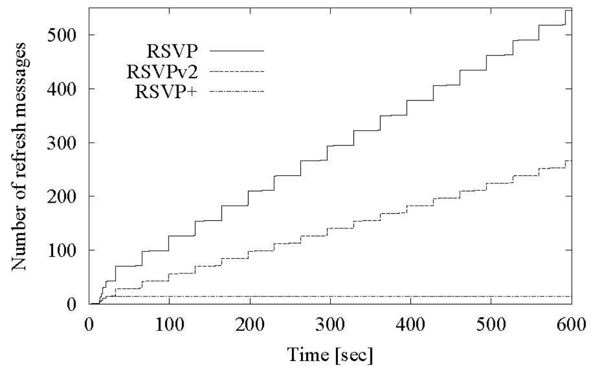

Fig. 10: Number of refresh messages sent by Router B for RSVP, RSVPv2 and RSVP+

It can be observed that for RSVP, the number of refresh messages at each cycle is 28 (i.e., 14 PATH messages and 14 RESV messages). The total number of messages for the simulation is then 546 (i.e., 18 cycles of refresh messages, 28 setup messages and 14 ResvConf messages). For RSVPv2, the number of refresh messages at each cycle is 14 (i.e., 7 PATH messages and 7 RESV messages). The total number of messages for the simulation is then 266 (i.e., 18 cycles of refresh and 14 setup messages). Finally, RSVP+ performs a total of 14 messages for the setup of the 7 bidirectional sessions. The teardown messages are not considered in the simulations.

Note that RSVP and RSVPv2 signaling messages grow linearly with the simulation time where RSVP+ signaling messages stay constant at 14 messages during the simulation time. This is an advantage for scalability assuming a large number of simultaneous sessions flow through the network.

The following notation is used to evaluate the average signaling load generate by a single router:

- N, the number of routers in the network;
- B, the bundle header length in RFC 2961 (in bits);
- C, the average number of bidirectional connections flowing through a router;
- D, the average session duration (in sec);
- F, the average number of connected interfaces on a router;
- H_{IP} , the length of an IP packet header (in bits);
- I_r, the refresh interval for a given router (in sec);
- M_{ID}, the length of message ID object in RFC 2961 (in bits);
- P, the length of the PATH message (in bits);
- R, the length of the RESV message (in bits);
- S, the header length of the message that contains the messages ID (in bits);
- L, the average refresh signaling load generated by a single router when using classical refresh procedure (in bps);
- L_B, the average refresh signaling load generated by a single router when using bundle described in RFC 2961 (in bps);
- L_S, the average refresh signaling load generated by a single router when using SRefresh described in RFC 2961 (in bps).

First note that for RSVP+, the signaling load is zero since there are no refresh messages. Moreover, for RFC 2961 solutions, there is only one IP header per bundle of RSVP messages and we consider the acknowledgement messages. The equation of the signaling load for each protocol follows.

RSVP

$$\begin{aligned} L &= 2C\left[\frac{1}{I_r}(P+H_{IP})+\frac{1}{I_r}(R+H_{IP})\right] \\ &= \frac{2C}{I_r}(P+R+H_{IP}) \qquad (1) \end{aligned}$$

$$\begin{aligned} L_B &= \frac{1}{I_r}(2CP+2CR+FB+FH_{IP}+2CM_{ID} \\ &\quad +2CM_{ID}+FS) \\ &= \frac{1}{I_r}\left[2C(P+R+M_{ID})+F(B+S+H_{IP})\right] \qquad (2) \end{aligned}$$

$$\begin{aligned} L_S &= \frac{1}{I_r}(2CM_{ID}+2CM_{ID}+2CM_{ID}+2CM_{ID} \\ &\quad +FS+FH_{IP}) \\ &= \frac{1}{I_r}\left[8CM_{ID}+F(S+H_{IP})\right] \qquad (3) \end{aligned}$$

RSVPv2

$$L = C\left[\frac{1}{I_r}(P+H_{IP})+\frac{1}{I_r}(R+H_{IP})\right]$$

$$= \frac{C}{I_r}(P+R+2H_{IP}) \quad (4)$$

$$L_B = \frac{1}{I_r}(CP+CR+FB+FH_{IP}+CM_{ID} + CM_{ID}+FS)$$

$$= \frac{1}{I_r}[C(P+R+2M_{ID})+F(B+S+H_{IP})] \quad (5)$$

$$L_S = \frac{1}{I_r}(CM_{ID}+CM_{ID}+CM_{ID}+CM_{ID} + FS+FH_{IP})$$

$$= \frac{1}{I_r}[4CM_{ID}+F(S+H_{IP})] \quad (6)$$

Based on RFC 2205, I_r is supposed to be between 15 and 45 seconds. Let us calculate the bandwidth (in bps) for the extreme values when B is set to 64 bits, F to 3, H_{IP} to 160 bits, M_{ID} to 64 bits, P to 896 bits, R to 736 bits, and S to 64 bits. The results are presented in tables 1 and 2. The results are described for classical refresh, bundle extensions and SRefresh extensions. For all scenarios, RSVPv2 gives better performances than RSVP.

Table 1: Refresh signaling load (in bps) for I_r = 15 seconds

	RSVP	RSVPv2
Classical	260.27 C	130.14 C
Bundle	234.67 C + 57.6	117.33 C + 57.6
SRefresh	34.13 C + 44.8	17.07 C + 44.8

Table 2: Refresh signaling load (in bps) for I_r = 45 seconds

	RSVP	RSVPv2
Classical	86.75 C	43.38 C
Bundle	78.22 C + 19.2	39.11 C + 19.2
SRefresh	11.37 C + 14.93	5.67 C + 14.93

The results in those tables also present the gap between RSVP+ and RSVP and the gap between RSVP+ and RSVPv2 since the signaling load of RSVP+ due to refresh message is zero.

These results are only the average signaling load for one router. Now consider a network with N = 30 nodes and C = 1000. Let us calculate the overall network signaling load NL_S, for some cases.

RSVP and classical Refresh, I_r = 15 seconds:
NL_S = 30[260.27(1000)] = 7.808 Mbps
RSVP and classical Refresh, I_r = 45 seconds:
NL_S = 30[86.75(1000)] = 2.602 Mbps
RSVPv2 and SRefresh, I_r = 15 seconds:
NL_S = 30[17.07(1000)+44.8] = 513.444 kbps

RSVPv2 and SRefresh, I_r = 45 seconds:
NL_S = 30[5.67(1000)+14.93] = 170.548 kbps.

As a result, RSVP with classical refresh is the worst case and RSVPv2 with SRefresh is the best case.

Best Field of Applications: RSVP+ is a protocol that can be used instead of RSVP. However, RSVP+ works better in unreliable networks because its restoration time is better.

Another important area is mobile networks where the network needs to quickly restore QoS in a new path when it switches from an access router to another. RSVP+ reduces the delay to restore the QoS in the new segment.

CONCLUSIONS AND FURTHER WORKS

In this paper, we proposed RSVP+, an improved resource reservation protocol. Our protocol is sender-oriented. With one PATH message, we can signal bidirectional reservations or unidirectional reservations in every direction without the need of a query message like in NSIS QoS. RSVP+ sets up the reservations more quickly than the other protocols and the restoration time is faster. Moreover, we have proposed a new refresh procedure with a minimum number of refresh messages. RSVP+ is also more robust because every data packet acts like a refresh packet. Many features of RSVP+ could be included in other resource reservation protocols like NSIS QoS.

There are several avenues of research that are open at this point. Indeed, it would be interesting to implement the refresh mechanism in a way that each router will update the state only after a certain interval of time and to see the performance of RSVP+ in a mobility context and to implement the solution for reducing the blocking due to uncompleted reservation.

REFERENCES

1. Abondo, C., and S. Pierre, 2004. Hierarchical Proxy Mobile Resource Reservation Protocol, *IET*F *Interne*t *Draft*, draft-abondo-hmprsvp-00.txt.
2. Berger, L., D. Gan, G. Swallow, P. Pan, F. Tommasi and S. Molendini, 2001. RSVP Refresh Overhead Reduction Extensions, *IET*F *RF*C *2961*.

3. Berger, L., 2003. Generalized Multi-Protocol Label Switching (GMPLS) Signaling Resource ReserVation Protocol-Traffic Engineering (RSVP-TE) Extensions, IETF RFC 3473.
4. Braden, R. and D. Clark, 1994. Integrated Services in the Internet Architecture: an Overview, *IET*F *RF*C *1633*.
5. Braden, L., S. Zhang, S. Berson, S. Herzog and S. Jamin, 1997. Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification, *IET*F *RF*C *2205*.
6. Braden, R. and L. Zhang, 1997. Resource ReSerVation Protocol (RSVP) – Version 1 Message Processing Rules, *IET*F *RF*C *2209*.
7. Hancock, R., G. Karagiannis, J. Loughney and S. Van den Bosch, 2005. Next Steps in Signaling (NSIS): Framework, *IET*F *RF*C *4080*.
8. Manner, J., G. Karagiannis, A. McDonald and S. Van den Bosch, 2005. NSLP for Quality-of-Service signaling, *IET*F *Interne*t *Draft*, draft-ietf-nsis-qos-nslp-08.txt.
9. Ni, N. and X. Xiao, 1999. Internet QoS: A Big Picture, *IEE*E *Networks*, 13:2, 8-18.
10. Westberg, L., A. Bader, D. Partain and V. Rexhepi, 2003. A Proposal for RSVPv2-NSLP, *IET*F *Interne*t *Draft*, draft-westberg-proposal-for-rsvpv2-nslp-00.txt.