Improving RSVP for Better Support of Internet Multimedia Communications

Laurent Mathy, David Hutchison, Stefan Schmid and Geoff Coulson Lancaster University Lancaster LA1 4YR UK {laurent, dh, sschmid, geoff}@comp.lancs.ac.uk

Abstract

We have observed the failure of RSVP to be widely accepted as the solution to resource reservation in the Internet. Because we are convinced of the need for resource reservation, in at least parts of the Internet, to support multimedia communications, we have set about trying to improve RSVP. By careful study of the protocol, we have identified areas for improvement, and propose REDO RSVP, a reduced overhead version that includes a fast establishment mechanism (FEM). In this paper we describe the rationale for REDO RSVP and present an analysis of its features and operations.

1. Introduction

The Internet was originally designed to offer a besteffort data transfer service. Such a type of service was simple to engineer while particularly well suited to applications whose utility is only loosely bound to the performance in the network. It is now widely recognised that to become a global telecommunication platform with integrated services – a must in the provision of information super-highways –the Internet must evolve to provide proper support to applications requiring stringent quality of service.

Several proposals have thus been made to support such an evolution of the network. These can be classified in three general categories: explicit QoS support [2](per flow or with aggregation), service differentiation [1] and IP/ATM integration [7]. Because they address different issues, several of these proposals will coexist in tomorrow's Internet. Without considering in further detail these solutions and numerous possibilities of integration, the following trend can be observed: the network will be based on a better control of its operations.

Although it was initially designed as a resource reservation protocol, RSVP ("Resource ReSerVation Protocol") is often considered as the control protocol in

the Internet (that is as a signalling protocol). Besides, several of the above mentioned proposals use RSVP in one way or another (e.g. [4][5]).

Consequently, RSVP's ability to carry control information efficiently will be vital to the effective operation of the Internet.

In this paper, we propose in section 2 a modified flow establishment mechanism for RSVP aimed at improving the resource reservation (or equivalently, distribution of control information) capabilities of the protocol. In section 3, we also seek ways to improve the scalability of RSVP in terms of the number of flows it can support. To that effect, we propose a REDuced Overhead (REDO RSVP) technique which is a form of aggregation for the control traffic to reduce the "steady state" overhead of RSVP at high loads of traffic.

2. Fast Establishment Mechanism (FEM)

RSVP uses periodic messages to manage its states. The lapse of time between consecutive messages defines the refresh period of the protocol (in a refresh period, there is one Path and one Resv message per flow on each link of the path). The default value for the refresh period R is 30 seconds [2]. Because RSVP messages are exchanged unreliably, such a lapse of time between messages seems prohibitively long, since it represents the average amount of time in which the loss of a control message can be corrected at reservation establishment. This could result in a very long establishment latency.

Simply reducing the value of the refresh period is not the right approach, however. Indeed, so doing would increase the control traffic associated with every flow, thus threatening to pose severe scaling problems. Consequently, reducing the refresh period *at establishment time only* is considered a better solution. Such shortened periods are called *establishment periods* in the rest of the paper. In modern high speed networks, message losses are mostly due to buffer overflow and consequently occur in bursts [3]. We therefore see that proper "inter-spacing" is required between consecutive control messages, to prevent them from encountering the same congestion conditions. This rules out the use of a fixed, short establishment period. Furthermore, to avoid unnecessary overhead, we must find a way to discover the end of the establishment phase, that is the moment after which the control messages related to a flow simply refresh the path states and reservations associated with that flow.

The first hurdle to overcome is the lack of an appropriate acknowledgement message in RSVP. However, when considering the different RSVP messages, it is clear that the role of an initial Path message is to "prepare" for a subsequent Resv message. The Resv message is then the obvious candidate to acknowledge the Path message: a Resv message indicates a successful reservation to the sender of the corresponding Path message. Therefore, any node that has forwarded a Path message, and has received a Resv message from every direct neighbour down the route followed by the corresponding flow, knows that the reservation has been successfully established downstream.

We still need to find a way for the receiver of a Path message to discover whether the establishment of a flow is in progress or has been completed. Because upstream nodes will use establishment periods shorter than the refresh period as long as they have not received a proper Resv message, a node can guess the status of a flow from the spacing of the Path messages it receives: if the lapse of time between consecutive Path messages is smaller than the shortest lapse of time allowed in "steady state" (that is $\mathbb{R}/2$ [2]), then the flow is more than likely being established and a Resv message should be forwarded as soon as possible to complete the establishment procedure (we thus see that the Resv message will be re-transmitted by the last RSVP node that correctly received the previous Resv message). On the other hand, if the time between consecutive Path messages is greater than or equal to the minimum allowed by the "classical" refresh periods, then we can suspect that the Path message is simply a refresh and a Resv message should only be sent when the current refresh period expires.

We have already ruled out the use of fixed periods at establishment. The other important point is that, if the establishment periods are too short, unnecessary RSVP messages will be sent, which increases the overhead of the protocol. Therefore, the initial establishment period (T_0) should not be smaller than the round-trip-time (RTT) for the RSVP messages, which may have to be estimated. After sending or forwarding the initial Path message, an RSVP node will wait for a lapse of time equal to the initial establishment period. If by that time a Resv message has not been received, the node suspects a loss of control messages and retransmits the Path message (this procedure is applied by all the nodes supporting the Fast Establishment Mechanism (FEM), so that the copy of the Path message is generated as close as possible to where the loss of the previous RSVP message occurred). The value of the establishment period should then be backed-off: we propose to multiply it by a factor $(1 + \Delta)$ at each retransmission of a Path message. As soon as a Resv message acknowledges the establishment of the reservation, the nodes start using the refresh period R for their Path messages. A refresh period equal to R is also used if no Resv message has been received, but the value of the establishment period has become greater than R/2. We see that, in any case, the nodes "fall back" to the behaviour prescribed by the "classical" **RSVP** specification.



Figure 1: Mean Establishment Time.

With T_0 set to 3 seconds and Δ set to 0.3, this timer scheme is equivalent to the staged refresh timers described in [6].

Simulations (see figure 1) show the superiority of FEM over "classical" RSVP. In these simulations, the

loss process on each direction of a link is represented, independently, by a two-state model. One of the states represents congestion (i.e. loss) periods while the other one represents no-loss periods. The loss process spends an exponentially distributed time in each state, with these exponential distributions set so that the mean congestion period is 200 ms and the loss process spends a long-term proportion of time of 90% and more in the no-loss state. Such a model was chosen because of its ability to mimic loss bursts in a simple way.

Configurations comprising respectively 3, 5, 10, 15, 20, 25 and 30 nodes (including the sender and the receiver) were considered. For every configuration, a flow was established 1000 times and each link had a delay of 10 ms (representing transmission, queuing and processing delays). Finally, the default of 30 seconds was used as the average value of the refresh periods in RSVP, with T_0 and Δ having the above mentioned values.

3. Reducing the Overhead

The concept of soft-state was originally introduced in RSVP to deal easily with a number of conditions [8]. These conditions all fall into one of the following categories:

- 1. changes in routes,
- 2. reclamation of obsolete resources,
- 3. dynamic membership of multicast groups,
- 4. loss of control messages,
- 5. temporary node failures.

However, it soon appeared that the soft-state mechanism used in RSVP was too slow to deal with conditions of type 1 or 3, and the mechanism of local repair [2] was then introduced to improve the protocol's responsiveness to such conditions. Furthermore, in the previous section, we have presented an improved method to deal with loss of control messages (at establishment time). This leaves the soft-state in charge of the reclamation of obsolete resources and of dealing with some temporary node failures. On the other hand, nodes relying on local repairs tend to reduce the amount of overhead by using a longer refresh period R. However, this has the consequence of increasing the lifetime of states and reservations [2], which considerably slows down the response to some error conditions. In this section, we seek ways to reduce the overhead of RSVP without impeding the protocol's responsiveness. We concentrate on RSVP nodes supporting both local repair and FEM.

3.1 "Steady State" Overhead in RSVP

In "classical" RSVP, periodic refresh messages have a keep-alive function which results in an overhead that is

linear in terms of the number of established flows. This overhead thus increases both the bandwidth requirement and CPU usage of the protocol, which results in scalability problems. This "steady state" overhead of RSVP is therefore a prime target when seeking to reduce the overall overhead.

When considering node or link failures, we see that refreshing each flow individually is a waste of resources. This is because of both the definition of a session in RSVP and the way IP routing works: all the data flows of a given session, visiting the same router at any given time, follow the same downstream path and are therefore collectively affected by any change of route or any network failure. A first improvement would be to refresh per session instead of per flow. This corresponds to an aggregation of control information, and is therefore independent of the number of flows. However, there is one condition for this technique to work properly: teardown messages must be delivered reliably. Indeed, if a teardown message on a flow went missing and was undetected, the associated states or resources would be kept partially alive by the session refreshes and would then waste resources until the whole session is released. Unfortunately, there is no easy way to introduce acknowledgement for the teardown message in RSVP without having to make substantial changes to the protocol. In particular, RSVP does not use any sequence number to identify its messages. Consequently, the only option left is to devise a totally new way of managing the reservations instead of trying to modify the messages of the classical protocol.

Doing so may at first glance seem drastic, but at least it has the potential virtue of leaving unchanged the core definition of the protocol: if the new mechanisms for reservation management can simply by added to the protocol specification, then backward compatibility with the classical mechanisms will be ensured.

3.2 New Focus for the Soft-State

Once flows are established, network failures can easily be detected by implementing the concept of soft-state *per neighbour*: neighbours periodically exchange *heartbeats* so that the absence of too many consecutive heartbeats is interpreted as a network failure. Note that such a mechanism allows the detection of every type of failure from the protocol point of view: link and router failure, as well as the failure of the RSVP process in a neighbouring node. In parts of the network using pointto-point links between nodes, there is only one neighbour per link, so the mechanism consists of a periodical check of each link. On broadcast links, the heartbeats can be sent to a well known multicast address so that only one heartbeat would be required from each node per refresh period. When implementing per-neighbour soft-state, a node only sends Resv messages in two cases: in response to Path messages; or after receiving a Resv message increasing the reservation on a flow. Similarly, after having received a Resv message, a node only forwards new Path messages or Path messages modifying the path state. Any other RSVP messages are treated in accordance with the RSVP specification.

The benefit of per-neighbour soft-state as opposed to per-flow soft-state is that it generates control messages at a fixed rate, independent of the number of established flows, as illustrated in figure 2. This makes it more scalable than its per-flow counterpart while potentially providing much faster reaction times.



Number of Flows Figure 2: Soft-State Overhead.

As mentioned in section 3.1, we now need to devise a way to exchange teardown messages reliably. However, need for complex there is no end-to-end acknowledgement semantic: after all, a signalling protocol carries information hop-by-hop, and the protocol can now rely on the heartbeats to detect the failure of a node. Again, introducing the concept of heartbeat in RSVP is probably enough of a "revolution", so we strive to avoid changing the existing messages, as well as defining any new message types, and in particular specific acknowledgement messages such as Teardown Acks. On the other hand, nothing prevents the heartbeats from carrying some form of identification (i.e. a sequence number field). Then, if each heartbeat sent on a link carries a copy of some or all the teardown messages that were previously sent on this link, reliable exchange of teardown messages between neighbours can guaranteed by having nodes piggybacking be acknowledgement of received heartbeats in their own heartbeats. A node will keep copying a teardown message in its heartbeats as long as a heartbeat containing it is not acknowledged by all the neighbours on the same link.

Of course, this means that each teardown appears at least twice on each link. However, because flows requiring reservations will usually be long-lived (e.g. flows belonging to multimedia sessions), such an extra overhead at teardown will usually be far smaller than the "steady state" overhead of "classical" RSVP. A protocol specification including local repair, FEM and per-neighbour soft-state (including reliable teardown messages) is called REDuced Overhead RSVP (REDO RSVP).

3.3 Compatibility with "Classical" RSVP

REDO mode should only be applied between REDO nodes. If a REDO node does not receive, or stops receiving, heartbeats from one of its neighbours, then "classical"/FEM RSVP must be used with that particular neighbour.

The following rules are followed by a REDO node applying "classical" mode with one of its neighbours:

- if the REDO node is *upstream* of its neighbour, **upstream classical mode** (UCM) is applied:
 - per-flow soft-state is applied to reservations;
 - periodical Path messages are sent downstream.
- if the REDO node is *downstream* of its neighbour,
 downstream classical mode (DCM) is applied:
 - per-flow soft-state is applied to the path states;
 - periodical Resv messages are sent upstream.

The rules about the sending of Path and Resv messages in REDO mode, described in section 3.2, ensure correct operation of the protocol.



Figure 3: Synchronisation of REDO nodes.

However, to avoid inconsistent states in the network, when a REDO node starts, or re-starts, sending heartbeats to one of its neighbours, *synchronisation* of these nodes must be completed before REDO mode is entered. In other words, during the synchronisation period, all the flows between the nodes being synchronised *must* be operated in "classical" mode. This synchronisation is illustrated in figure 3. The role of the stabilisation is to prevent re-incarnated control messages, which could have been queued (e.g. in device driver buffers) but not delivered before the start of the synchronisation, from wrongly triggering REDO mode in a node. Similarly, the contention ensures that the upstream node (relatively to a flow) enters REDO mode after the downstream one (if this was not the case, the absence of Path messages would cause the reservation to time-out in the downstream node). The length of the stabilisation and contention timers should therefore be greater than the maximum packet lifetime (MPL) in the network. A value of 30 seconds to 2 minutes is proposed.

5. Conclusions

We have presented FEM, a Fast resource Establishment Mechanism that is more robust to the conditions in the network than the establishment mechanism currently used in RSVP, and that establishes resources faster and without any unnecessary increase of control traffic. FEM relies totally on the Path and Resv messages defined in RSVP and thus does not require major amendments to the protocol specification.

The soft-state mechanism in RSVP is a simple and efficient way to deal with exceptional conditions in the network (see section 3). Unfortunately, it does not provide a good response to many error condition that can be encountered. In other words, the soft-state mechanism, as used in RSVP, is probably too simple to constitute the main building block of the protocol. Local repairs [2] have, for instance, been introduced because of the poor responsiveness of the soft-state in the event of route changes. Furthermore, the soft-state mechanism has the drawback of incurring an important "steadystate" overhead that jeopardises the scalability of RSVP. We have therefore proposed a way of overcoming these problems by "re-thinking" the use of the soft-state mechanism: the main idea of our REDO RSVP is that if the soft-state is applied per-neighbour instead of perflow, the steady-state overhead is reduced and is independent of the number of flows in the network.

As no change to the messages currently used in "classical" RSVP" is required by REDO RSVP (which instead relies upon a new message type), REDO RSVP can be seen as a super-set of the mechanisms defined in "classical" RSVP. This guarantees backward compatibility and allows for a progressive deployment of REDO RSVP in the Internet.

Finally, we believe that the added complexity in REDO RSVP is marginal compared to its benefits. However, FEM, which is an integral part of REDO RSVP, can also be deployed on its own as an amendment to "classical" RSVP.

References

[1] D. Black, S. Blake, M. Carlson, E. Davies, Z. Wang, W. Weiss: An Architecture for Differentiated Services. Internet Draft draft-ietf-diffserv-arch-00, IETF, May 1998. Work in Progress.

[2] R. Braden, L. Zhang, S. Berson, S. Herzog, S. Jamin: Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification. RFC 2205, IETF, September 1997.

[3] I. Cidon, A. Khamisy, M. Sidi: Analysis of Packet :Loss in High Speed Networks. *IEEE Trans. Info. Theory*, 39(1):98-108, January 1993.

[4] B. Davie, Y. Rekhter, E. Rosen, A. Viswanathan, V. Srinivasan, S. Blake: Use of Label Switching with RSVP. Internet Draft draft-ietf-mpls-rsvp-00, IETF, March 1998. Work in Progress.

[5] T. Li, Y. Rekhter: A Provider Architecture for Differentiated Services and Traffic Engineering (PASTE). RFC 2430, IETF, October 1998.

[6] P. Pan, H. Schulzrinne, R. Guérin: Staged Refresh Timers for RSVP. Internet Draft draft-pan-rsvp-timer-00, IEFT, November 1997. Work in Progress.

[7] E. Rosen, A. Viswanathan, R. Callon: Multiprotocol Label Switching Architecture, Internet Draft draft-ietf-mpls-arch-02, IETF, July 1998. Work in Progress.

[8] L. Zhang, S. Deering, D. Estrin, D. Zappala: RSVP: A New Resource ReSerVation Protocol. *IEEE Network*, 7(5):8-18, September 1993.