REDO RSVP: Efficient Signalling for Multimedia in the Internet.

Laurent Mathy, David Hutchison, Stefan Schmid, and Steven Simpson

Lancaster University, UK
{laurent, dh, sschmid, ss}@comp.lancs.ac.uk

Abstract. Alarming reports of performance and scalability problems associated with per-flow reservations have led many to doubt the ultimate utility of RSVP and the Integrated Services Architecture that relies on it. Because we are convinced of the need for some form of resource reservation, to support multimedia communications in the Internet, 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 a detailed analysis of its features and operations. We also analyse REDO RSVP by means of simulations, and show that it offers improvements to the performance of RSVP.

1 Introduction

The Internet was originally designed to offer a best-effort 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 recognized that to become a global telecommunication platform with integrated services—a must in the provision of information superhighways—the Internet must evolve to provide proper support to applications requiring stringent qualities 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 (per flow or with aggregation), service differentiation and IP/ATM integration. Because they address different issues, these proposal 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 resource usage.

Among resource management mechanisms, those offering the finest grain of traffic control operate on a per-flow basis. These mechanisms, however, suffer

[0] In Proc. of 6th Intl. Workshop on Interactive Distributed Multimedia Systems and Telecommunication Services (IDMS'99). M. Diaz, P. Owezarski, P. Sénac (Eds.). Toulouse, France, October 1999. LNCS 1718. pp. 17–30. Springer.

from scalability problems as the number of flows with reservations increases. Although this rules out their use within the core of the network, per-flow provisioning can still be used at the edge of the network where the concentration of flows is rather low. In the Internet, the IntServ (Integrated Services) architecture [4] offers a framework for per-flow QoS control which relies on RSVP ("Resource ReSerVation Protocol") [5][12] as the signalling protocol to carry resource requirements between the source and the destination(s) of a flow. Furthermore, several proposals, mainly flow aggregation techniques [10][2] and the DiffServ (Differentiated Services) architecture [3], have been put forward to overcome the state scalability problems in the core of the network. These latter proposals will have to be integrated with the "traditional" IntServ approach to provide some form of end-to-end QoS commitments acceptable to multimedia and real-time applications.

The above mentioned flow aggregation techniques do not necessarily result in any reduction in the number of control messages sent per individual flow in the core of the network and certainly add to the overhead by generating messages per aggregate. Therefore, such a message overhead may still create a computational bottleneck (depending on the number of aggregates supported) in core routers as well as consuming bandwidth. Consequently, the steady-state message overhead in RSVP represents a significant scalability challenge.

Although RSVP was originally designed for resource reservation, several proposals have now been tabled where RSVP is used to carry other types of control information in the network [7][9]. Another example is the possible use of RSVP within the DiffServ architecture [1]. Therefore, we believe that, whether it is for resource reservation or other control/signalling purposes, RSVP will have to operate over routes of various lengths and to satisfy demands exhibiting a broad range of dynamics. Consequently, RSVP's ability to carry control information efficiently across the network in any circumstances will be vital to the effective operation of the Internet.

That is why we propose, in section 2, a modified flow establishment mechanism aimed at improving the resource set-up capabilities of the protocol. In section 3, we also seek ways to improve the message overhead 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 in order to reduce the "steady state" overhead of RSVP at high loads of traffic. Section 4 presents comparative simulation results between RSVP and the improvements proposed in the paper. Section 5 concludes our discussion.

2 Improving Resource Set-up in RSVP

RSVP uses periodic messages to manage its states [5]:

- a sender sends Path messages (per flow) toward the receiver(s). These messages construct and maintain a "path state" and advertise the characteristics of the flow.

– a receiver sends Resv messages towards the senders. These messages construct and maintain actual reservations.
– every intermediate node periodically sends its *own* Path and reservation messages, that is there is no way senders or receivers can force a node to send copies of RSVP messages in the network.

The lapse of time between consecutive Path or Resv messages defines the refresh period of the protocol (in a refresh period, there is one Path and one Resv messages per flow on each link of the path). The default value R for the refresh period is 30 seconds[1]. Because RSVP messages are exchanged unreliably, such a lapse of time between similar RSVP 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 results in a very long latency at reservation establishment, as confirmed by the simulation results presented in section 4.

Simply reducing the value of the refresh period is not the right approach, however. Indeed, doing so would increase the control traffic associated with every flow, thus threatening to pose severe scalability problems. Consequently, reducing the refresh period *at establishment time only*[2] (including local repair conditions, see [5]) is considered a better solution.

In modern high speed networks, message losses are mostly due to buffer overflow and thus occur in bursts [6]. We therefore see that proper "inter-spacing" is required between consecutive control messages, to prevent them from encountering the same congestion conditions along their route. This observation rules out the use of a fixed, short establishment period for the sending of consecutive RSVP messages during the establishment phase. Furthermore, in order 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 acknowledgment message in RSVP. A straightforward introduction of such messages in the protocol (as proposed in [11]) would result in major changes to the protocol definition and specification. 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

[1] Each period is chosen randomly in $[R/2, 3R/2]$[5].
[2] Such shortened refresh periods are called *establishment periods* in the rest of the paper.

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", 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 (that is $R/2$), 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[3]. Of course, for this technique to be robust in the event of loss of Path messages, the periods used at establishment time must be quite a bit smaller than $R/2$.

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 (T_0). If by that time a Resv message has not been received, the node retransmits the Path message (this procedure is applied by all the nodes supporting our technique, so that the copy of the Path message is generate as close as possible to where the loss of the previous RSVP message occurred). In order to be adaptive to a wide range of congestion conditions, the value of the establishment period must be backed-off: we propose to multiply it by a factor $(1 + \Delta)$ at each retransmission of the 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 messages has been received, but the value of the establishment period has become greater than R. We therefore see that, in any case, the nodes "fall back" to the behaviour prescribed by the "classical" RSVP specification and FEM RSVP is backward compatible with "classical" RSVP.

With T_0 set to 3 seconds and Δ set to 0.3, this timer scheme is equivallent to the staged refresh timers described in [11]. It should be noted that for local repairs, a shorter value of T_0 would be acceptable, since we expect the new portion of the route to be fairly short. Furthermore, such a more aggressive behaviour of the protocol is justified by the fact that local repairs apply to existing flows.

[3] The period used by a node to send Resv messages is the refresh period defined in "classical" RSVP. The concept of establishment period timer does not apply to Resv messages.

3 Reducing the Overhead

The concept of soft-state was originally introduced in RSVP to deal easily with a number of conditions [12]. 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 (see section [5]) was then introduced to improve the protocol's responsiveness to such conditions. Furthermore, in section 2, we 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.

3.1 "Steady State" Overhead in "Classical" 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 the 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 inefficient. 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 route change or any network failure. We could therefore seek ways for any RSVP node to refresh simultaneously several flows, and indeed all the flows, shared with a direct neighbour. 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 was lost, the associated states or resources would be kept partially alive and would then waste resources indefinitely. If RSVP was modified to provide reliable teardown of flows, the risk of "resource leak" would be avoided and the steady-state message overhead of RSVP could then be dramatically reduced, solving the message overhead scalability problem.

3.2 New Focus for the Soft-State

As we saw in section 2, receipt of Resv messages indicates successful establishment of a reservation downstream of the node that received these messages.

Therefore, if a node implements local repair, the "exceptional" conditions that have still to be taken care of are network failures and the loss of control messages used to reclaim obsolete resources.

Once flows are established[4], 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 signalling 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 point-to-point links between nodes, there is only one neighbour per link, so the mechanism consists of a periodic check of each link. On broadcast links, the heartbeats could 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 changing 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 of a flow. 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 reservations, as illustrated in figure 1. This makes it more scalable than its per-flow counterpart while potentially providing much faster reaction times.

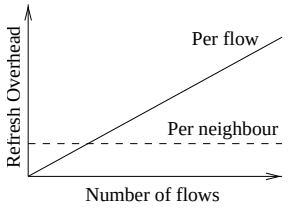


Fig. 1. Soft-State Overhead

We now need to devise a way to exchange teardown messages reliably. However, there is no need for complex end-to-end acknowledgment semantics: after all, a signalling protocol carries information hop-by-hop, and we can now rely on the heartbeats to detect the failure of a node (and therefore to react properly to any possible damage resulting from such failure conditions). Introducing the concept of hearbeats in RSVP is probably enough of a "revolution", so we strive to avoid changing the existing message types as well as defining any new

[4] Before reservation is completed, FEM or "classical" RSVP is enforced.

message types, and in particular *specific* acknowledgment messages such as TearAcks. On the other hand, nothing prevents the heartbeats from carrying some form of identification (i.e. 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 be guaranteed by having nodes piggybacking acknowledgment 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 has not been 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. It should also be noted that because the loss of a teardown message is only corrected, in "classical" RSVP, by the expiration of a lifetimer which will usually be several minutes long (see section [5]), the new teardown scheme proposed here will be much more efficient at resource reclamation than "classical" RSVP and will therefore improve resource usage in the network.

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 to communicate with that particular neighbour. Furthermore, as we will see in the next section, it is sometimes necessary for REDO nodes to revert to "classical" mode for certain flows, even when heartbeats are correctly exchanged.

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

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

At any time, either or both upstream or downstream classical modes can be applied to a flow by a REDO node.

REDO mode can only be applied to a flow between two REDO nodes when:

1. heartbeats are exchanged between these nodes (and the nodes have been synchronized, see below);
2. for the corresponding flow:
   - the node in upstream classical mode has received a Resv message acknowledging its Path messages; we say that node enters **upstream REDO mode** for that flow.
   - The node in downstream classical mode has received a Path message; we say that node enters **downstream REDO mode** for that flow.

The rules about the sending of Path and Resv messages in REDO mode, described in section 3.2, ensure correct operation of the protocol. However, when a node receives an RSVP message that changes the state (path state or reservation) associated with a flow, the node operates that flow in "classical" mode, until the above mentioned rules to enter REDO mode are met again.

However, in order to avoid inconsistent states in the network, the following rule must *always* be observed: when a REDO node starts, or re-starts, sending heartbeats to one of its neighbours, *synchronization* of these nodes must be completed before REDO mode can be applied to any flow between these nodes. In other words, during the synchronization period, all the flows between the nodes being synchronized *must* be operated in "classical" mode. The synchronisation is illustrated in figure 2. The synchronisation is only considered complete when the stabilisation period has expired. The role of the stabilisation is to prevent reincarnated 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. This synchronisation mechanism is applied to each direction of traffic between the nodes.

It should be noted that a REDO node that does not receive any heartbeat from any of its neighbours on a given interface will behave totally like a "classical"/FEM RSVP node on that interface and should therefore refrain from sending heartbeats. We therefore see that *backward compatibility* is guaranteed, with REDO nodes "bridging" the two RSVP worlds. This allows for a progressive deployment of REDO RSVP.

3.4 Exception Handling in REDO RSVP

With REDO RSVP, as soon as a reservation has been established, and as long as no "special" conditions appear in the network, nodes simply exchange "empty" heartbeat messages. We now turn our attention to the kind of "special" conditions the protocol has to deal with and describe how REDO RSVP handles them.

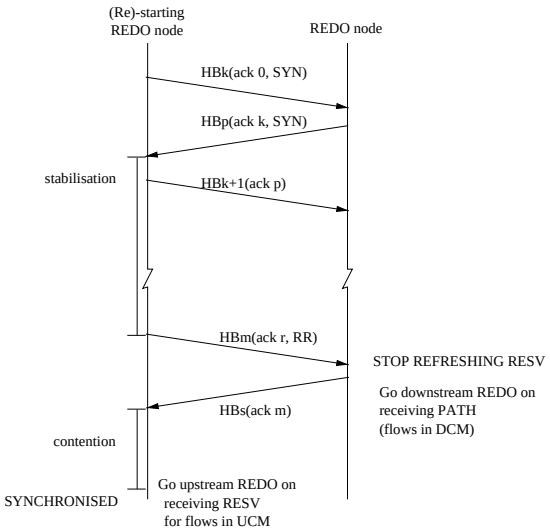


Fig. 2. Heartbeats (HB) during synchronisation between REDO nodes.

Route Change REDO nodes swiftly respond to route changes by using local repair and FEM on the new segment of route. We note that the use of FEM is essential in such a case: because any route change affects several sessions, there are potentially many flows to repair, so the newly visited nodes are likely to see a "burst" of control messages which could result in temporary congestion of the signalling traffic.

When a node which has initiated a local repair has received Resv messages from all its new neighbours downstream, it then knows that a flow has been repaired. This also means that it is now safe to tear down the old reservation on the old route. Because Path teardown messages follow normal IP routing, "classical" RSVP has no way of sending such a message down the old route. On the other hand, REDO RSVP can place such messages in the heartbeats destined for the old "next-hop" node and can therefore promptly reclaim the resources on the old route. Any node receiving a teardown from its neighbour registered as the *previous hop* in its path state copies the teardown in its heartbeats[5] downstream. If a node receives any teardown ("classical" or within a heartbeat) from a neighbour which is not the previous hop indicated by the path state, the teardown should be discarded, to avoid tearing down resources along portions of the old route which are still part of the new route. We therefore see that REDO RSVP can, in the event of route changes, reclaim resources faster than "classical" RSVP.

[5] In a unicast session, a node receiving a teardown in a heartbeat can alternatively *first* forward the information in a "classical" teardown message before propagating it within heartbeats.

On the other hand, if the initiator of the local repair has not received any Resv messages from some of the new next-hop nodes, one cannot conclude whether resources on the old route are still being used. This is because several conditions can prevent a Resv message from reaching the initiator of the local repair:

1. there are insufficient resources on the new route;
2. repair Path messages have been lost on new links;
3. repair Resv messages have been lost on new links.

The important point is that in the first and third cases above, the path state has been repaired, so that any teardown message will stay confined to the portion of the old route that is not used any more. However, in the second case, if a teardown was ever sent down the old route, it would propagate all the way to the receivers. To avoid such spurious release of resources, we propose that when sending the last Path message of the "establishment phase" of a flow (see section 2), the initiator of the local repair instructs, in its next heartbeat, its direct neighbours along the old route to revert to "classical" RSVP operation on the corresponding flows. Because hearbeats are exchanged reliably, so are these messages. The net result of this procedure is that, if during a route change, some flows are not repaired within the operation of FEM, REDO RSVP reverts back to "classical" RSVP for these flows, so that its performance is *in the worst case* equal to the performance of "classical" RSVP[6].

Network Failures Expiration of the soft-state timer associated with a neighbour is interpreted as a link or node failure. In such a situation,"classical" RSVP operations are reverted to for the flows handled by that neighbour. That is, upstream classical mode is applied to flows for which that neighbour is a downstream node, while downstream classical mode is applied to flows for which that neighbour is an upstream node.

In order to deal properly with "asymmetrical" link failures, a REDO node whose soft-state timer associated with one of its neighbours has timed out, should refrain from sending heartbeats towards these neighbours. REDO operations can only resume between these nodes after they have been re-synchronized (see section 3.3). Also, the rule of synchronisation ensures that, in case of a node or REDO RSVP process failure, the synchronisation will be triggered from the failure point on reset. However, in the event of a link failure, all REDO processes involved may refrain from sending heartbeats. In such a case, the two following techniques are suggested to discover the recovery from failure:

1. a synchronisation phase is triggered as soon as any "classical" RSVP message (with appropriate version number) is received from a neighbour involved in the failure;
2. periodical "synchronisation probes" are sent to the neighbours involved in the failure.

[6] Of course, as soon as the reservation for that flow is completed downstream of a node, that node enters normal REDO mode with its downstream neighbours.

On a broadcast link, where a well known multicast address may be used for the exchange of heartbeats (section 3.2), it is impossible to refrain from sending heartbeats to a specific neighbour. In such a case, synchronisation information concerning a particular neighbour must be present in every heartbeat sent to the multicast address.

The strategy of "going classical" was chosen in the event of a failure because REDO nodes do not know if, when and how the routing protocol is going to work around the fault. As a consequence, in the event of network failures, REDO RSVP performs at least as well as "classical" RSVP. Actually, in the event of network failures, REDO RSVP offers possibilities not supported by "classical" RSVP.

Indeed, REDO nodes directly connected upstream and downstream of a fault could propagate information about the failure towards respectively the senders and receivers, as well as possibly informing the local routing daemon. When used with advanced routing protocols offering alternative routes, this option would allow to by-pass a failure swiftly. In particular, for fault-tolerant systems having stand-by routes, REDO RSVP would enable fast recovery from faults.

Note that, because of the possibly relatively high frequency of the heartbeats, the values of the soft-state timers have to be chosen in such a way as to minimize the risks of erroneous failure detection. Furthermore, the scheme would benefit if nodes gave preferential treatment to heartbeats.

Transient Failures A node or REDO RSVP process failure which is detected neither by the heartbeat mechanism nor the routing protocol is called a transient failure. This can happen when recovery from the failure is achieved within the lifetime of the per-neighbour soft-state (which should be configurable per neighbour) or the response time of the routing protocol.

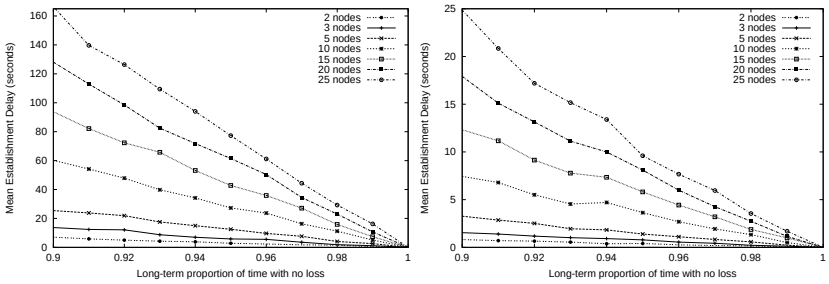
Upon reset, the REDO RSVP process will initiate a synchronisation phase with each of its neighbours (see section 3.3). This will have the effect of forcing "classical" operation (with FEM) in the neighbours for the flows that were handled by the REDO RSVP process that had failed, which of course results in a swift re-establishment of the reservations for these flows in the recovered node.

4 Simulation Results

4.1 FEM RSVP

We have simulated the external behaviour of both "classical" and FEM RSVP, in order to compare them. Our simulations consisted of repeated reservation establishments between a sender and a receiver, over routes of various length and under various loss conditions.

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



3.(a): MED with RSVP.

3.(b): MED with FEM RSVP.

Fig. 3. Mean Establishment Delay (MED)

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.

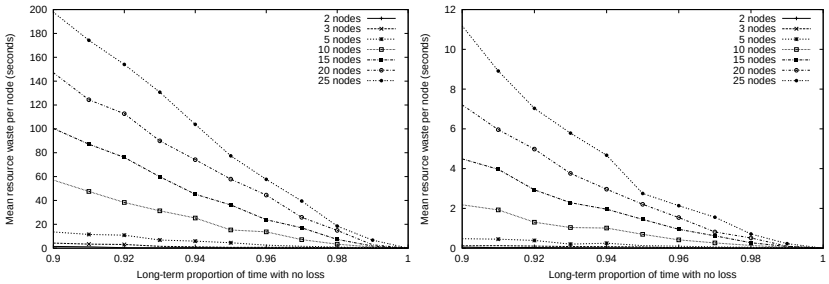
Configurations comprising respectively 2, 3, 5, 10, 15, 20 and 25 nodes (including the sender and the receiver) were considered. For every configuration, 1000 flows were established and no delay was introduced in nodes and links to isolate the time overhead introduced by the external (i.e. observable) operation of the protocol. Finally, the default of 30 seconds was used as the average value of the refresh periods in RSVP, while for FEM RSVP T_0 and Δ have the values proposed in section 2. Figures 3.(a) and 3.(b) show the measured mean establishment delay and clearly demonstrate the gain in performance obtained with FEM RSVP. Note the low establishment delay achieved by FEM RSVP over short routes with low per-hop success probability. This result is of particular interest in the case of a change of route, since local repairs produce bursts of control messages which can temporarily congest the signalling channel.

4.2 REDO RSVP

We have measured the waste of resources incurred by both "classical" and REDO RSVP. By "waste of resource", we mean the average time that a resource is reserved in a node while the corresponding flow is not in use by the application. Such a waste occur at both resource establishment and teardown. At establishment, the waste is due to the receiver-oriented nature of the protocol: the reservation has to make its way up towards the source while the reserved resources will only be used when the reservation actually reaches that source. REDO RSVP minimises this type of resource waste by using the FEM extension at flow establishment. At teardown, the waste is essentially due to losses

of teardown messages. REDO RSVP speeds up resource reclamation by implementing reliable exchanges of teardown messages within the use of its periodic heartbeats.

The simulations presented in this section were performed under the same conditions as the ones in the previous section, but with an average period H of 2 seconds for the heartbeats. Indeed, to avoid synchronisation of the heartbeats [8], the time between consecutive heartbeats is chosen randomly in $[H/2, 3H/2]$. In these circumstances, the overhead generated by the heartbeats is equivalent to the "steady-state" overhead of 15 "classical" RSVP flows. Figure 4 show the mean resource waste incurred per flow in any RSVP node along the route of that flow. The results show that REDO RSVP reduces resource waste in the nodes by an order of magnitude.



4.(a): With RSVP

4.(b): With REDO RSVP

Fig. 4. Mean per flow Resource Waste per Node.

5 Conclusions

We have presented FEM, a Fast resource Establishment Mechanism that is not only more robust to the conditions in the network than the establishment mechanism currently used in RSVP, but also establishes resources faster and without any unnecessary increase of control traffic. FEM is a very simple mechanism that 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 way to deal with some conditions in the network. Unfortunately, it does not provide a good response to every error condition (see section 3). It also has the drawback of incurring an important "steady-state" overhead that jeopardizes the scalability of RSVP. We have 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 per-flow, the steady-state overhead is reduced and is independent of the number of flows in the network.

REDO RSVP responds to each situation in the network in a specific way, including reverting to "classical" RSVP operation in conditions where per-flow soft-state is deemed the most appropriate and simple solution. This ensures that REDO RSVP consistently exhibits performance which is better than, or equal to, that of "classical" RSVP. No change to the messages currently used in "classical" RSVP is required in REDO RSVP, which instead relies upon a new message type. REDO RSVP can thus 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.

Almost the entire complexity of REDO RSVP resides in its synchronisation mechanism. Because the synchronisation is only performed occasionally, we believe that the added operational complexity in REDO RSVP is marginal compared to its demonstrated benefits. However, FEM, which is part of REDO RSVP, can also be deployed on its own as an amendment to "classical" RSVP.

References

1. Y. Bernet, J. Binder, S. Blake, M. Carlson, E. Davies, B. Ohlman, D. Verma, Z. Wang, and W. Weiss. A Framework for Differentiated Services. Internet Draft draft-ietf-diffser-framework-00, IETF, May 1998. Work in Progress.
2. S. Berson and S. Vincent. Aggregation of Internet Integrated Services State. Internet Draft draft-berson-rsvp-aggregation-00, IETF, Aug 1998.
3. D. Black, S. Blake, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An Architecture for Differentiated Services. Internet Draft draft-ietf-diffserv-arch-00, IETF, May 1998. Work in Progress.
4. R. Braden, D. Clark, and S. Shenker. Integrated Services in the Internet Architecture: an Overview. RFC 1633, IETF, Jun 1994.
5. R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource ReSerVation Protocol (RSVP)—Version 1 Functional Specification. RFC 2205, IETF, Sep 1997.
6. I. Cidon, A. Khamisy, and M. Sidi. Analysis of Packet Loss Processes in High Speed Networks. *IEEE Trans. Info. Theory*, 39(1):98–108, Jan 1993.
7. B. Davie, Y. Rekhter, E. Rosen, A. Viswanathan, V. Srinivasan, and S. Blake. Use of Label Switching with RSVP. Internet Draft draft-ietf-mpls-rsvp-00, IETF, Mar 1998. Work in Progress.
8. S. Floyd and V. Jacobson. The Synchronization of Periodic Routing Messages. *IEEE/ACM Trans. Network.*, 2(2):122–136, Apr 1994.
9. O. Fourmeaux and S. Fdida. Multicast for RSVP Switching. *Telecommunication Systems Journal*, 11(1-2):85–104, Mar 1999.
10. R. Guérin, S. Blake, and S. Herzog. Aggregating RSVP-based QoS Requests. Internet Draft draft-guerin-aggreg-rsvp-00, IETF, Nov 1997. Work in Progress.
11. P. Pan, H. Schulzrinne, and R. Guérin. Staged Refresh Timers for RSVP. Internet Draft draft-pan-rsvp-timer-00, IETF, Nov 1997. Work in Progress.
12. L. Zhang, S. Deering, D. Estrin, and D. Zappala. RSVP: A New Resource ReSerVation Protocol. *IEEE Network*, 7(5):8–18, Sep 1993.