ELSEVIER

Computer Communications 25 (2002) 1782–1798

computer communications

www.elsevier.com/locate/comcom

A performance study of RSVP with proposed extensions

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

Computing Department, Lancaster University, LA1 4YR Lancaster, UK

Received 9 June 2000; revised 27 February 2002; accepted 27 February 2002

Abstract

Resource ReSerVation Protocol (RSVP) was developed as an intended key component for the evolving Internet, and in particular for the Integrated Services architecture. Therefore, RSVP performance is crucially important; yet this has been little studied up till now. In this paper, we target two of the most important aspects of RSVP: its ability to establish flows and its steady-state overhead. We first identify the factors influencing the performance of the protocol by modelling the establishment mechanism. Then, we propose the principles of a Fast Establishment Mechanism (FEM) aimed at speeding up the set-up procedure in RSVP. We analyse FEM by means of simulation, and show that it offers improvements to the performance of RSVP over a range of likely circumstances. We also present the principles of a simple mechanism aimed at reducing the steady-state (i.e. refresh) message overhead of RSVP.

Keywords: Resource ReSerVation Protocol; Internet; Traffic

1. Introduction

It is now widely recognized 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 for applications, such as distributed multimedia applications, that require a variety of qualities of service. In an ideal world, this evolution should depend on the evolution of the traffic mix in the network (that is the ratio best-effort and guaranteed traffic). Unfortunately, the evolution of the traffic mix is very hard to forecast.

If best-effort traffic clearly dominates, then a well provisioned network, possibly enhanced with some simple form of traffic differentiation [5], can probably satisfy the occasional requests for stringent Quality of Service (QoS) guarantees [8]. In this case, adequate bandwidth is the key to QoS. On the other hand, if the proportion of guaranteed traffic becomes significant, more advanced resource management mechanisms are likely to be needed to meet the level of service expected by the users. One such mechanism considered here is resource reservation.

Studies [16] have found that in today's Internet, which is dominated by best-effort traffic, congestion occurs mainly at the edge of the network (e.g. in ISP access networks, links from campus networks, etc.). However, it has also been shown that some backbone links (especially some trans-continental links) are saturated for a substantial part of the day. These observations suggest that to support applications—such as interactive multimedia application or real-time applications—with stringent QoS requirements, resource management mechanisms will have to be provided at the edge of the network at least. This argument is reinforced by the fact that, as such applications appear, the traffic mix in the network may shift towards traffic requiring more resource usage control.

Among resource management mechanisms, those offering the finest grain of traffic control operate on a per-flow basis. These mechanisms, however, suffer 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 Integrated Services (IntServ) architecture [6] offers a framework for per-flow QoS control which relies on Resource ReSerVation Protocol (RSVP) [7,18] 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 [4,14] and the Differentiated Services (DiffServ) architecture [5], have been put forward to overcome the state scalability problems in the core of the network. A

* Corresponding author. Fax: +44-1524-593608.

E-mail addresses: dh@comp.lancs.ac.uk (D. Hutchison), laurent@comp.lancs.ac.uk (L. Mathy), sschmid@comp.lancs.ac.uk (S. Schmid), ss@comp.lancs.ac.uk (S. Simpson).

0140-3664/02/$ - see front matter
PII: S0140-3664(02)00102-0

proposal for a combination of IntServ and DiffServ has been considered within the IETF [3], in an effort to combine the scalability advantages of flow aggregation with the fine grain control advantages of resource reservation. The framework provides for end-to-end quantitative QoS by applying the IntServ model end-to-end across a network containing one or more DiffServ regions. IntServ enables hosts to request per-flow resources along end-to-end data paths and to obtain feedback regarding the admissibility of these requests. DiffServ enables scalability across large networks.

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. Therefore, such a message overhead may create a computational bottleneck 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 [1,11,13]. Another example is the possible use of RSVP within the DiffServ architecture [5]. 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, in this paper, we study some of RSVP's performance aspects. The lack of experiments in 'real conditions' leads us to develop, in Section 3, a mathematical model of the flow establishment phase in RSVP. The results yielded by our model clearly show the need to revise the flow establishment procedure of RSVP. The principles of a modified flow establishment mechanism are then presented in Section 4. Simulation results comparing the establishment procedure currently used in RSVP with our proposal are given in Section 4.2. In Section 5, we also outline a simple method to reduce the steady-state (i.e. refresh) message overhead in RSVP. Some relevant related work is discussed in Section 6, and Section 7 concludes our discussion.

It should be noted that the primary context of resource reservation has influenced the naming of the control messages used in RSVP and it is therefore easier to describe the operations of RSVP in this context. The reader should however bear in mind that the results presented in this paper equally apply to RSVP as a 'general' signalling protocol. Moreover, in this paper, we are only concerned with performance aspects of RSVP: state scalability issues are not addressed.

The work presented in this paper is part of a wider effort at Lancaster University aimed at improving the support for distributed multimedia applications in the Internet, and specifically investigating the viability of resource management mechanisms. The present paper is a more fully developed version of Ref. [15].

2. A brief overview of RSVP

RSVP is based on the concept of session [7]. A session is composed of at least one data flow and is defined in relation to a 'destination' (more precisely as the triplet (destination address, destination port, protocol id)). As the destination address can be a multicast address, the destination can thus be a group of receivers as well as a single receiver.

In RSVP, a flow is defined as any sub-set of the packets in a session, or in other words, as a sub-set of the packets sent to a given destination. A flow is therefore simplex. Theoretically, the sub-set of packets making up a flow may be arbitrary, but in the current state of the RSVP specification, a flow is defined as the set of packets emitted from a given 'source' (identified by the pair (source address, source port)).[1]

RSVP works as follows [7,18]:

- Path messages are periodically[2] sent towards the destination and establish a 'path state' per flow in the routers.
- Resv messages are periodically[2] sent towards the sources and establish the required reservations along the path followed by the data packets. The style of reservation in RSVP is thus 'receiver oriented', since it is the receivers that initiate the requests for resources to be reserved.
- In order to reduce the overhead associated with RSVP, any Path or Resv message that does not have an net effect on the states held by a router is not forwarded immediately by that router. Instead, each router periodically issues its own Path and Resv messages carrying information about the flows it holds.
- A lifetime L is associated with each reserved resource. This timer is reset each time a Resv message confirms the use of the resource. If the timer expires, the resource is freed. This principle of resource management based on timers is called 'soft-state'. Soft-state is also applied to the path state in the routers (in this case, the timer is reset upon reception of a Path message). By default, L is 2 min 37.5 s [7].
- To improve RSVP responsiveness to network dynamics, the mechanism called 'local repair' has been introduced. When an RSVP entity detects a change of route, it sends Path messages down the new route for the flows whose

[1] This definition of a flow could, and should, be updated in future versions of the protocol to exploit the possibilities offered by the flow label field in the IPv6 header.

[2] Each period is chosen randomly in $[R/2, 3R/2]$, with $R = 30$ s by default [7,10].

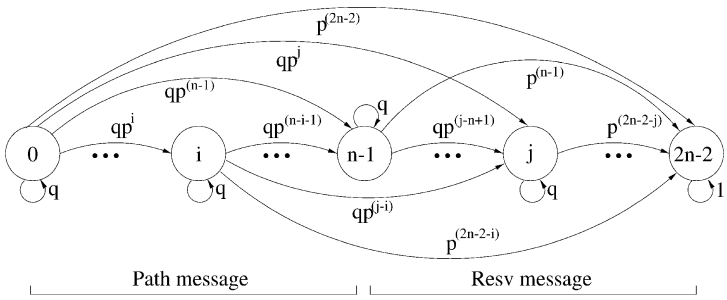


Fig. 1. Markov chain modelling RSVP, with $p + q = 1$.

route has changed. When the downstream RSVP entity, situated at the junction of the old and new routes, receives these Path messages, it updates its path states accordingly and immediately sends a Resv message upstream along the new segment of route for the corresponding flows.

- Teardown messages (resp. PathTear and ResvTear) are available for immediate release of the corresponding states (resp. path state and reservations). Teardown requests can either be initiated by a sender, a receiver, or any intermediate RSVP router (upon state timeout or service preemption).

It is worth noting that all the messages described above are delivered unreliably: because of the protocol reliance on soft-states, the concept of acknowledgement is not used in RSVP.

3. A model for flow establishment in RSVP

Although its core ideas appeared a few years ago [18] and both research and commercial implementations are now available, to the best of our knowledge, no large-scale experiment has been done with RSVP yet. This lack of experimentation means that we do not know how RSVP will perform when used in real conditions, as encountered in the Internet. In this section, we develop a mathematical model of the establishment phase of RSVP in order to gain some insight of its performance. We are actually interested in quantifying RSVP's ability to make a successful reservation over a route where resources are plentiful. Although such a question may at first glance seem superfluous, we think it is of paramount importance to address it in order to assess RSVP's viability in the Internet, because of the unreliable character of the delivery of RSVP messages. In other words, we are interested in RSVP's external behaviour at reservation establishment as well as in dealing with network dynamics (local repair [7] may be seen as simply establishing a new reservation on a new portion of route).

In the rest of this section, we label as *sender* an RSVP node that initiates; forwards (as opposed to forward) the first Path message on a route where no (path) state has been established for the corresponding flow yet. A sender can either be an end-system (in the case of a reservation establishment) but could also be a router detecting a change of route (in the case of a local repair). We label as *receiver* an RSVP node that initiate; forwards (as opposed to forward) the first Resv message, in response to the sender's Path message, on the 'reverse route' where no reservation has been made for the corresponding flow yet. Again, the receiver can either be an end-system or a router. Any other node treating (i.e. creating state and reservation) and forwarding the messages along the route are called RSVP routers. Although our model will be developed considering only one sender and one receiver, it is nevertheless applicable to the multicast case by applying it to the (sub-)branches of multicast trees.

The central parameter in our model is p, the *per-hop success probability*, which is the probability that an RSVP message sent by an RSVP node is correctly received by the RSVP process in the next node. We therefore see that p takes into account not only transmission errors but also overflow conditions at the different levels of the protocol architecture (i.e. link, IP and RSVP layers). In a well dimensioned network, routers should be provisioned with enough resources to accommodate most of the control traffic. We therefore expect the value of p to be high (i.e. close to 1). Consequently, in our model, we will ignore state timeouts because such events occur with a probability $(1 - p)^K \approx 0$ (with $K = 3$ by default [7]). Therefore, our model will yield slightly overestimated results.

We know that to establish a reservation for a flow:

1. The sender issues a Path message to the receiver.
2. Upon receipt of that Path message, the receiver issues a Resv message describing the resources required.
3. Every intermediate node periodically sends *its own* Path and Resv messages, that is there is no way to force a node to send copies of RSVP messages in the network.

It is only when the Resv message reaches the sender that the reservation is fully established (i.e. considered successful). Furthermore, because we ignore state timeouts, if any of the RSVP messages is ever lost along the way, an equivalent message is re-emitted *from the last node where it was last correctly received* at the beginning of the next refresh period. The establishment of a reservation thus

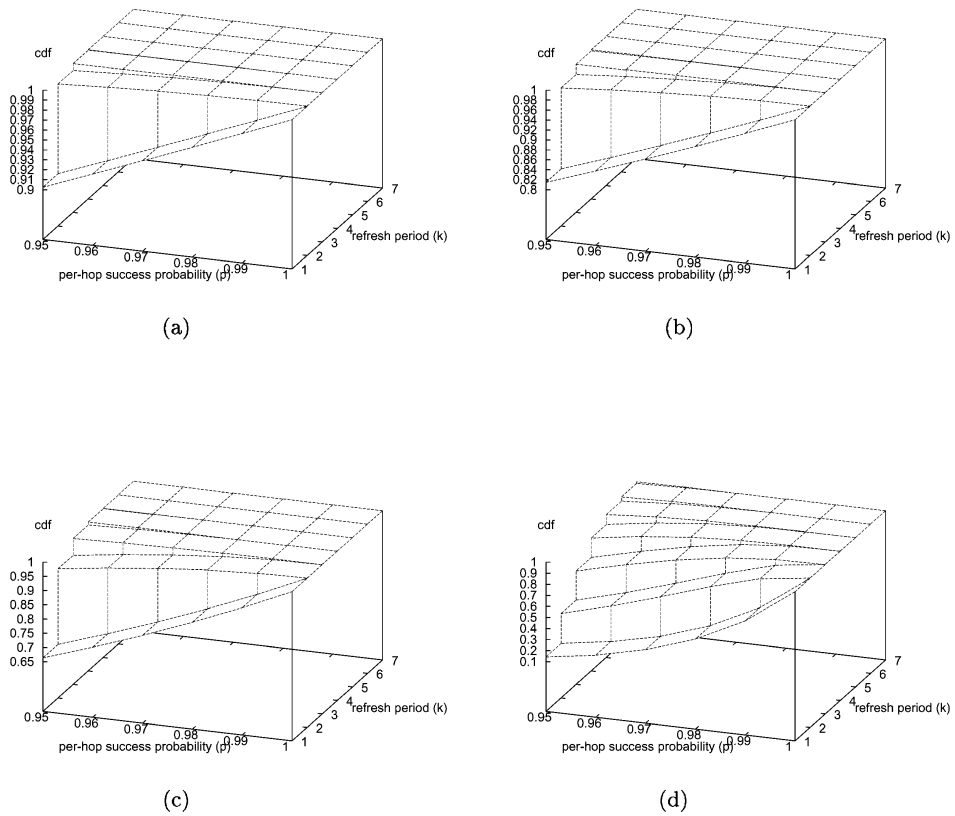


Fig. 2. CDFs for the success probabilities. (a) Along two nodes. (b) Along three nodes. (c) Along five nodes. (d) Along 20 nodes.

appears to be 'incremental': from a refresh period to the next, the number of nodes holding proper state/reservation for the flow cannot decrease. Therefore the refresh messages exchanged between nodes where the corresponding states/-reservations have already been established have no influence on the rest of the establishment procedure and can thus be ignored. In other words, we always consider the control message which is 'ahead' of the others.

Another way to describe successful reservation establishment, is to note that on a route involving n RSVP nodes (including the sender and the receiver), the Path and Resv messages we consider must collectively travel $2n-2$ hops. This is because the initial Path message is 'generated' at the sender while the initial Resv message is 'generated' at the receiver: these messages do not need to 'travel' to reach these nodes.

Considering the number of hops the RSVP messages have travelled by the end of each refresh period, naturally leads to a *discrete time semi-Markov process* with $2n-2$ states (see Fig. 1), whose embedded Markov chain (representing the process at the instants of state transitions) has the following transition probabilities:

$$p_{i,j} = 0 \qquad 0 \le i < 2(n-1),\ \ 0 \le j < i \tag{1}$$

$$p_{i,j} = p^{j-i}(1-p) \qquad 0 \le i < 2(n-1),\ \ i \le j < 2(n-1) \tag{2}$$

$$p_{i,2n-2} = p^{2(n-1)-i} \qquad 0 \le i \le 2(n-1). \tag{3}$$

Eq. (1) is the mathematical expression for the incremental establishment simplification. Eqs. (2) and (3) are based on the fact that a transition from any state of the chain to any other (including itself), is equivalent to a control message travelling a number of hops equal to the distance between the states. Eq. (2) simply expresses that if the state reached is not the last one, then the control message must have been lost between two nodes. Eq. (3) states that when the Resv message reaches the sender, no more control traffic is required. Also, note that the last state of the chain is *absorbing*, stating that the establishment of the reservation is complete. Eqs. (1)–(3) unambiguously describes the *transition probability matrix* $\mathbf{P}$ of the Markov chain.[3]

We note $\boldsymbol{\pi}^{(k)}$ the state probability vector at the end of the kth refresh period (the kth refresh period is represented by the kth state transition):

$$\boldsymbol{\pi}^{(k)} \triangleq \left(\pi_0^{(k)}, \pi_1^{(k)}, \ldots, \pi_{2n-2}^{(k)}\right).$$

The state probability vector is obtained by:

$$\boldsymbol{\pi}^{(k)} = \boldsymbol{\pi}^{(0)}\mathbf{P}^k, \tag{4}$$

where $\boldsymbol{\pi}^{(0)}$ is the initial state probability vector, with $\boldsymbol{\pi}^{(0)} = (1, 0, \ldots, 0)$, since we start with a Path message at the sender.

[3] It is easy to verify that $\sum_{j=0}^{2(n-1)} p_{i,j} = 1,\ 0 \le i \le 2(n-1)$.

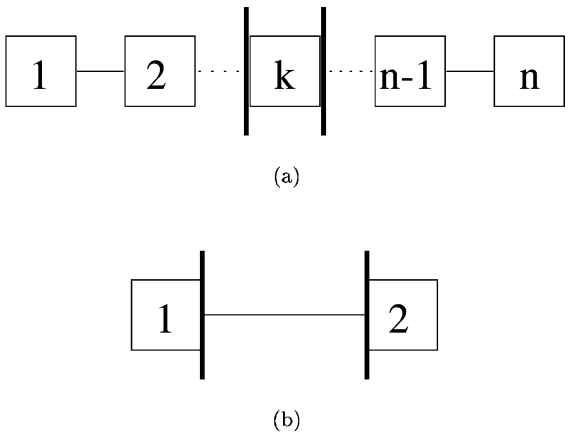


Fig. 3. Example of mapping of a real scenario to model. Bold interfaces are lossy. (a) Scenario. (b) Modelisation.

We now express RSVP's ability to make a successful reservation. Let S be the number of periods required to establish a flow with reservation. In the context of the model, S is the number of transitions required to reach state $2n-2$.

Relations (1) and (3) imply that all the states but the last one are *transient*.[4] The last state of the chain is absorbing and so it is *recurrent*. The last state of the chain is eventually entered, and once it has entered it, the random process never leaves it. Consequently, $\pi_{2n-2}^{(k)}$—the probability of being in state $2n-2$ at the end of period k—can be interpreted as the probability that a reservation has been established by the end of the kth refresh period:

$$\pi_{2n-2}^{(k)} = P[S \leq k] = F_S(k). \tag{5}$$

The transient behaviour of $\pi_{2n-2}^{(k)}$ thus represents the Cumulative Distribution Function (CDF) of the success probability P_S.

Fig. 2 shows the values of $F_S(k)$ for different route length (n) in terms of different per-hop success probabilities (p) and various number of periods (k).

In Fig. 2, it clearly appears that, even for short routes, RSVP will perform reasonably well only for very high per-hop success probabilities. Indeed, the probability of success within the first period is in accordance with Eqs. (5) and (3)

$$P_S^{(1)} = P[S=1] = \pi_{2n-2}^{(1)} - \pi_{2n-2}^{(0)} = p^{2(n-1)}, \tag{6}$$

where $\pi_{2n-2}^{(0)}$ is 0 because the chain is always started in state 0. $P_S^{(1)}$ is an important quantity because it expresses the chances that a reservation is established without any loss of control messages.

The behaviour of Eq. (6) when p is in the neighbourhood of 1, is given by

$$\lim_{p\to 1^-} \frac{\partial}{\partial p} P_S^{(1)} = \begin{matrix} 2(n-1) \\ \geq 2, \end{matrix} \qquad \text{for } n \geq 2, \tag{7}$$

which shows that RSVP is very sensitive to dynamic condition changes in the network.

In Appendix A, we derive the average number of refresh periods needed to establish a reservation:

$$E[S] = (2n-2)\frac{1-p}{p} + 1. \tag{8}$$

The previous result allows us to obtain the average contribution of the external behaviour of RSVP to the establishment time T of a reservation, or in other words, the average establishment time when the queueing, transmission, propagation and internal processing delays are neglected:

$$E[T] \approx (E[S]-1)R = R(2n-2)\frac{1-p}{p}. \tag{9}$$

Both Eqs. (8) and (9) confirm the sensitivity of RSVP to the values of the per-hop success probability.

The model presented in this section may seem a little pessimistic since we use the same value of the per-hop success probability on every link of a route. However, by extending the concept of an 'RSVP node' to encompass the idea of a 'lossless RSVP cloud', that is a contiguous region of the network where losses of RSVP messages do not occur or can be neglected, the model can be used to describe more realistic situations. For example, the scenario with two nodes can model a route of any length with one bottleneck, that is a route where control messages are only lost at one congested router. Indeed, Fig. 3(a) and (b) representing, respectively, the 'real scenario' and its modelisation, has the same number of interfaces where messages can be lost.

As already pointed out at the beginning of this section, the value of the per-hop success probability p strongly depends on the rate of control traffic generated in the network and the associated resources needed to absorb such traffic. In RSVP, this rate of control traffic depends on both the rate at which new reservation requests are issued (either by end-systems or following route changes) and the average number of existing flows with reservations (because of the periodic refresh associated with the soft-state). The results exposed in this section strongly suggest the need for a dedicated 'signalling channel'[5] in order to keep the per-hop success probability as high as possible.

This is not only true to ensure good performance at flow establishment, but also to improve resource utilization in the network. Indeed, for a resource to be released within a short delay, a teardown message must travel the path followed by a flow *without being lost* [7]. It is so because any loss of a teardown message can only be corrected when a lifetime expires, which can take several minutes [7] and thus induce inefficient resource utilization. For a route with n nodes (sender and receiver included), the probability of 'immediate' release of the resources of a flow is p^{n-1}. This value

[4] Indeed, for $0 \leq i < 2(n-1)$, we have $\sum_{n=1}^{\infty} p_{i,i}(n) = \sum_{n=1}^{\infty} (1-p)^n = \frac{1-p}{p} < \infty$.

[5] This signalling channels includes the RSVP processes in the routers, and hence the associated queues.

shows that, although the release of resources is less sensitive to the value of the per-hop success probability than the establishment (see Eq. (6)), this sensitivity may nonetheless become a problem over medium-length or long routes.

As a consequence, for RSVP to give acceptable results as the signalling protocol of the Internet, a carefully provisioned signalling channel will be required. Obviously, in the parts of the network where RSVP will be deployed, such a channel will be built by reserving resources for the control traffic; in non-RSVP networks (connecting 'RSVP clouds' together), mechanisms such as traffic differentiation [5] or prioritization will be required.

4. Improving reservation establishment

RSVP uses periodic messages to manage its states. 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 message per flow on each link of the path). The default value for the refresh period R is 30 s. From the results of our model presented in Section 3, 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. It therefore seems natural to reduce the length of the refresh periods to improve RSVP's performance at establishment time.

Simply reducing the value of the refresh period is not the right approach, however. Indeed, this would substantially increase the control traffic associated with every flow, thus increasing the required capacity of the signalling channel of the network while threatening to pose severe scalability problems. Consequently, reducing the refresh period *at establishment time only*[6] (including local repair conditions) is considered a better solution. In Ref. [7], it is suggested that a node could, at establishment, temporarily send control messages more often than dictated by the refresh period. However, the question of how many, as well as how often, such messages should be sent has not been addressed. This is precisely what we propose to do in this section.

4.1. Fast establishment mechanism

In modern high speed networks, message losses are mostly due to buffer overflow. As a consequence, such losses occur in bursts [9]. 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 only way to discover the end of the establishment phase of a flow is somehow to use the concept of acknowledgment. In order to keep our discussion as clear as possible and focus on principles, we present, in this section, a simple solution that only relies on the use of the Path and Resv messages, and hence does not require the introduction of explicit acknowledgment messages in RSVP.

It is clear that the role of an initial Path message is to 'prepare' for a subsequent Resv message. A Resv message can therefore be considered as an acknowledgment for a Path message. This Resv message also 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 $R/2$ [7]) then the flow is most 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.[7] 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 lot smaller than $R/2$. Also, the difference between $R/2$ and any establishment period should be at least an order of magnitude larger than delay variations in the network.

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

[6] Such shortened refresh periods are called *establishment periods* in the rest of the paper.

[7] 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.

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 he 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 suspects a loss of control messages and re-transmits the Path message (this procedure is applied by all the nodes supporting Fast Establishment Mechanism (FEM), so that the copy of the Path message is generates 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 re-transmission 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. We therefore see that, in any case, the nodes 'fall back' to the behaviour prescribed by the classical RSVP specification. We therefore see that FEM RSVP is backward compatible with classical RSVP.

With T_0 set to 3 s and Δ set to 0.3, this timer scheme is equivalent to the staged refresh timers described in Ref. [17]. 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.

The simple solution presented above requires a receiver to be able to send a Resv message immediately on receipt of a Path message. Although it will always be so in the case of local repairs, the reservation requirements might not be readily available if interaction with the end user is needed to determine these requirements. In this latter case, the solution proposed here would result in much unnecessary overhead and would fail to correct swiftly the loss of a Resv message. One way to overcome such a problem would be to define acknowledgment message for Path messages (e.g. PathAck) and Resv messages (e.g. ResvAck). An immediate Resv message would be generated at a receiver whenever possible (and FEM would be applied as presented in this section), otherwise an immediate PathAck would be sent and FEM applied on Path–PathAck pairs. As soon as the reservation requirements would be known, a Resv message would be sent and the FEM mechanisms could be applied, in the 'reverse' direction, on Resv–ResvAck message pairs. Using FEM separately on Path and Resv messages would then ensure prompt recovery from losses of control messages.

In the multicast case, two strategies can be adopted for FEM:

1. As soon as the first Resv message is received by a node, that node forwards it upstream without delay. This has the advantage of quickly propagating reservations along the multicast tree. However, although any Resv message subsequently received by the node and increasing the reservation demands would be immediately forwarded upstream (according to the message forwarding rules of RSVP [7]), losses of such messages would not be corrected by FEM but by later refreshes. In such a case, FEM speeds up the initial establishment but cannot reduce the latency of increasing reservation demands.
2. A node could hold the reservation requirements received in Resv messages either for a small lapse of time or until it has received Resv/PathAck messages on all the output ports of the multicast tree, before it forwards its own Resv message upstream. This has the advantage of establishing a more complete reservation at once, but has the risk of potentially increasing the overall establishment latency. We say 'more complete reservation' because a node does not necessarily know exactly how many 'next hop' nodes are reachable through each of its output ports.

Further work is needed to study and evaluate these possible strategies in the multicast case.

Finally, in order to avoid unnecessary overhead, FEM RSVP nodes should try to discover the capabilities of their neighbours (this could be done by recording the protocol version in the received messages) and refrain from using FEM when the next hop node does not support it. Furthermore, in multicast, the usual state/message merging should be applied.

4.2. Simulation results

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 lengths and under distinctly different 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 distributions set so that the mean congestion period is 200 ms and the loss process spends a long-term proportion of time equal to the per-hop success probability 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 with values of the per-hop success probability ranging from 99 to 100% inclusive. Such values for the per-hop success probability were chosen because they are likely

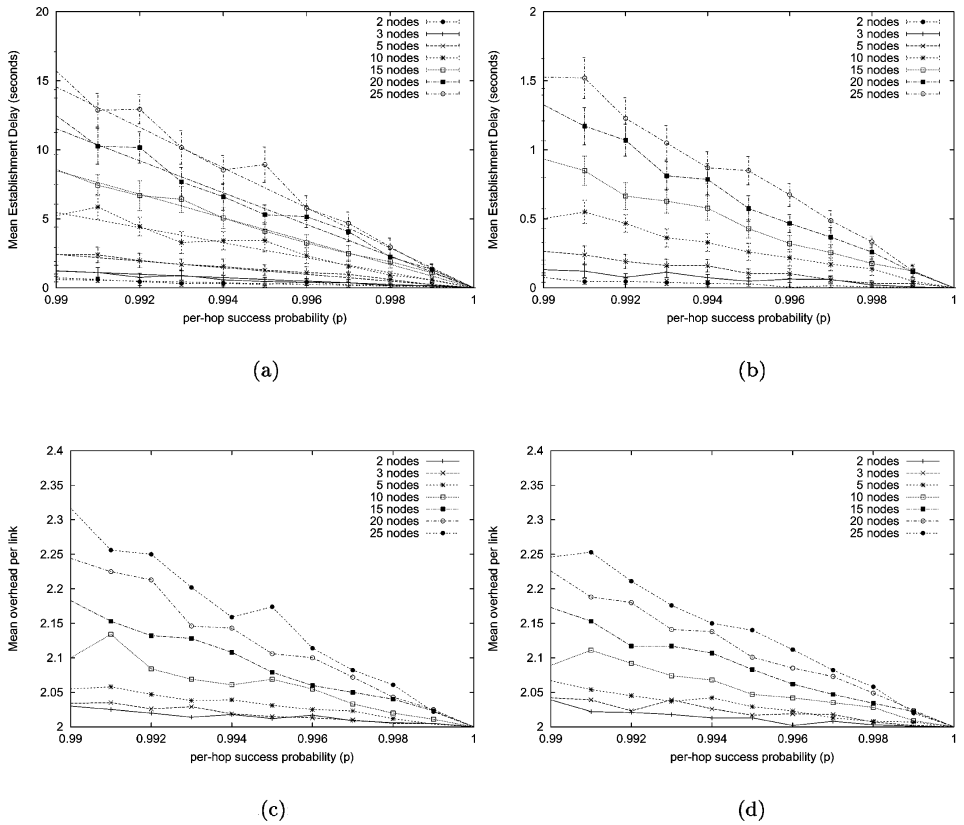


Fig. 4. MED and MOPL. (a) MED with RSVP. Simulation and theoretical (Eq. (9)) results are represented. (b) MED with FEM RSVP. (c) MOPL with RSVP. (d) MOPL with FEM RSVP.

to be encountered in a well dimensioned network. 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 s was used as the average value of the refresh periods[8] in RSVP, while for FEM RSVP, T_0 and Δ have the values proposed in Section 4.1.

The measured quantities were the Mean Establishment Delay (MED) and the Mean Overhead Per Link (MOPL) (i.e. the mean number of control messages per link per reservation). For the MED, 95% confidence intervals were computed using the method of batch means [12] (p. 293) on 40 batches of 25 samples each. The results are given in Fig. 4. In interpreting the results in this section, special attention must be paid to the meaning of the per-hop success probability which essentially represents the chances of survival of a control message from one RSVP process to the next. Therefore, the corresponding per-hop loss probability (i.e. the probability that a control message does not reach the next RSVP process) is expected to be greater than usual packet loss probabilities, because it encompasses possible losses due to overflows of the queue holding messages awaiting to be treated by the RSVP process which usually resides in the slow path of a router.

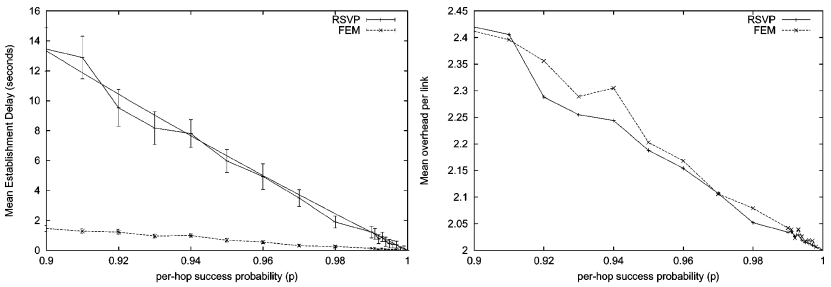
Apart from the obvious gain in performance, Fig. 4 also confirms the more predictable (or more stable) behaviour of FEM RSVP (the 95% confidence intervals are about an order of magnitude smaller in FEM RSVP than in RSVP). The message overhead (Fig. 4(c) and (d)) is fairly similar in both cases. There is however a slight trend showing a better effectiveness of FEM as reliability decreases. This property could prove very valuable in the case of local repairs, where bursts of repair messages could result in congestion of the signalling channel (including queues to the RSVP processes in routers).

Fig. 4(a) validates the predictions of our mathematical model, despite fundamental differences in the loss processes[9] assumed in Section 3 and these simulations.

Fig. 5, by contrast, specifically compares RSVP and FEM RSVP over routes of three nodes with per-hop success probabilities ranging from 90 to 100%. This scenario is important since it represents a case where the sender and receiver are wireless terminals, with the wired network in-between considered lossless. Again, the stability and gain

[8] In RSVP, each period is chosen randomly in [R/2, 3R/2] [7,10].

[9] Uniform versus two-state loss model.



(a) (b)

Fig. 5. Comparison of RSVP and FEM over routes of three nodes in a wireless scenario. (a) Establishment delay. Simulation and theoretical (Eq. (9)) results are represented. (b) Message overhead.

in performance of FEM, with no significant increase in message overhead, are clearly demonstrated.

Another equally important scenario is the case of a route of any length with a single bottleneck. As mentioned in Section 3, such a scenario is modelled as a route comprising two nodes connected by a lossy link (see Fig. 3). The MED is given in Fig. 6, for different ranges of per-hop success probabilities. In this case, although the mean time penalty introduced by the external behaviour of RSVP is quite small and would not be considered unacceptable as long as the message loss probability does not exceed a percent. Once more, this confirms RSVP's sensitivity to the values of the per-hop success probability and their variations, even when only facing a single point of congestion.

Finally, in view of the results presented in this section, it could be argued that if the per-hop success probability was kept very close to 1, the performance of RSVP would be satisfactory, and hence the FEM extension would not be necessary (especially over short routes). It should however be noticed that our results consist of mean values, averaged over a large number of flows and that on any particular occasion, the loss of any control message at flow establishment, is penalised by a delay of at least $R/2$ seconds (i.e. 15 s by default) with RSVP, but only by a delay of at least T_0 seconds (i.e. 3 s in the context of our simulations) with FEM RSVP. This fact alone probably justifies the use of FEM RSVP, even when the probability of losing a control message is extremely low.

5. Reducing the overhead

The concept of soft-state was originally introduced in RSVP to deal easily with a number of conditions [18]. 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 2) was then introduced to improve the protocol's responsiveness to such conditions. Furthermore, in Section 4.1, 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. On the other hand, nodes relying on local repairs can reduce the amount of overhead by using a longer refresh period R. But this 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.

5.1. 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, 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 change of route 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 were lost, the associated states or resources would be kept partially alive and would then

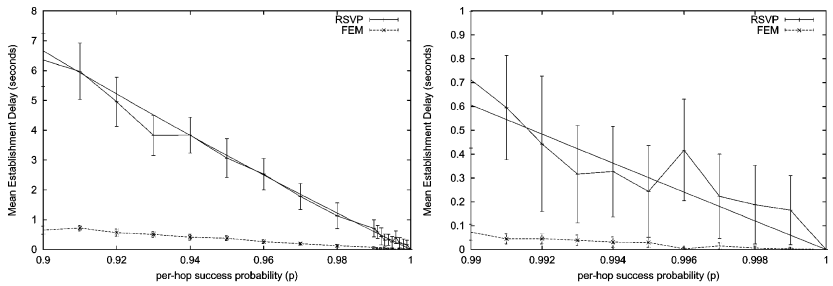


Fig. 6. Comparison of RSVP and FEM over routes with a single bottleneck. (a) Low per-hop success probabilities at bottleneck. Simulation and theoretical (Eq. (9)) results are represented.

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.

There are several ways in which RSVP can be modified to exchange teardown messages reliably. If we assume that RSVP has been explicitly modified to provide reliable teardown, then the mechanism presented in Section 5.2 is merely concerned with reducing the steady-state (i.e. refresh) overhead in RSVP. However, as we will see in Section 5.2, if the teardown procedure of RSVP has not been modified, teardown reliability can be easily embedded in the mechanism used to reduce the message overhead.

5.2. New focus for the soft-state

As we saw in Section 4.1, 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 with reservations are established,[10] 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.

As in Section 4.1, in order to concentrate on principles, we assume that Resv messages acknowledge Path messages. If explicit acknowledgments are introduced in RSVP, then the following discussion can easily be updated to reflect the introduction of such messages in the protocol specification. The point here is that the introduction of explicit acknowledgment for existing RSVP messages can improve and simplify the operation of the protocol, but is not mandatory.

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 (because per-flow refreshes of reservations are suppressed, such a Resv message should be re-sent until reception of a ResvConf message from the previous hop). Similarly, after having received a Resv message, a node only forwards new Path messages or Path messages modifying the path state of a flow (again, to ensure proper state synchronisation between nodes, a mechanism such as FEM must be used when propagating all Path messages). 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 shown in Fig. 7. This makes it more scalable than its per-flow counterpart while potentially providing much faster reaction times (because reasonably short periods are not a scalability threat anymore). As mentioned in Section 5.1, if no reliable teardown mechanism has been introduced in RSVP, 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

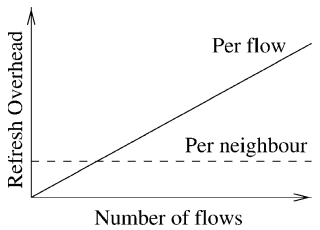


Fig. 7. Soft-state overhead.

[10] Before reservation is completed, FEM or classical RSVP is used.

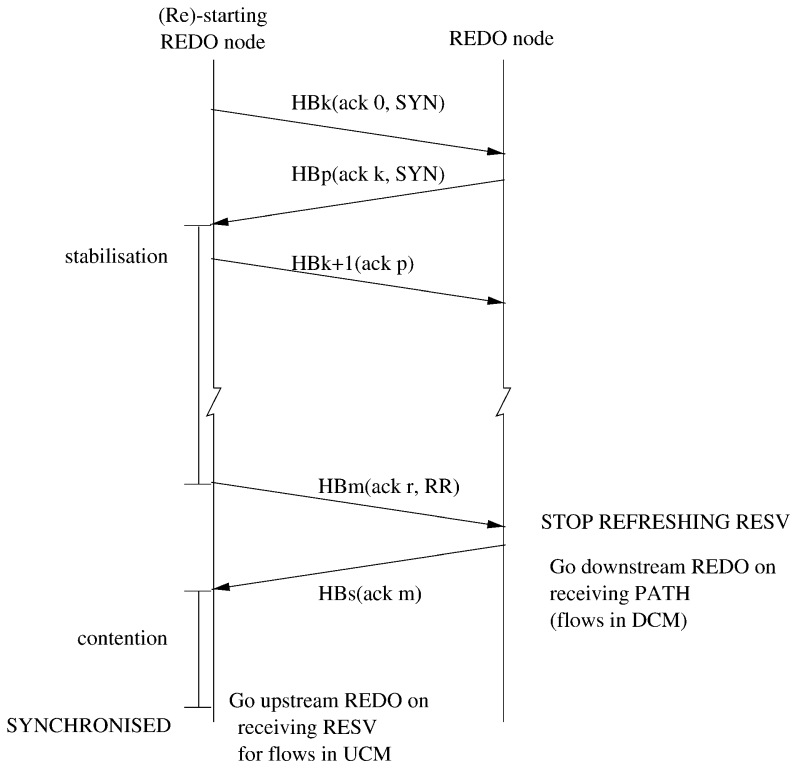


Fig. 8. Synchronisation between REDO nodes.

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). If teardown is unreliable, 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 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. 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 2), 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.

In the rest of the paper, a protocol specification including local repair, FEM and per-neighbour soft-state (including reliable teardown messages) will be called REDuced Overhead RSVP (REDO RSVP).

5.3. Compatibility of REDO RSVP 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 neighbour, then classical/FEM RSVP must be used to communicate with that particular neighbour. Furthermore, as we will see in Section 5.4, it is sometimes necessary for REDO nodes to revert to classical mode for certain flows, even when they correctly exchange heartbeats.

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** is applied to the concerned flows:
  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** is applied to the concerned flows:
  per-flow soft-state is applied to path states;
  periodical Resv messages are sent upstream.

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

REDO mode can only be entered between two REDO nodes for a flow when:

1. Heartbeats are exchanged between these nodes.
2. For the corresponding flow:
   - the node in upstream classical mode has received a Resv message acknowledging its Path messages;
   - the node in downstream classical mode has received a Path message.

The rules about the sending of Path and Resv messages in REDO mode, described in Section 5.2, ensure correct operations of the protocol.

However, in order to avoid inconsistent states in the network, the following rule must *always* be observed:[11] 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 can be applied to any flow between these nodes. In other words, during the synchronisation period, all the flows between the nodes being synchronised *must* be operated in classical mode. This synchronisation consists of a three-way handshake between the nodes, followed by some period of time for temporization (see Fig. 8). The synchronisation is only considered complete when the temporization period has expired. This temporization is necessary to prevent 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 on some flows. The length of the temporization should therefore be greater than the Maximum Packet Lifetime (MPL) in the network. A temporization of 30 s to 2 min is proposed.

It should be noted that a REDO node that does not receive any heartbeat from any of its neighbour 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. We believe that such a deployment will be mostly profitable in the core of the network (where the load on routers can be very high). On the other hand, at the edge of the network (e.g. in a LAN), we do not expect to encounter a very high volume of flows per node. Consequently, classical/FEM RSVP is likely to be used in end-systems and small routers, while REDO RSVP should gear up 'top of the range' routers.

5.4. Exceptions handling in REDO RSVP

With REDO RSVP, as soon as a reservation has been established, and as long as no 'special' condition appears 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.

5.4.1. Change of route

REDO nodes swiftly respond to route changes by using local repair and FEM on the new portion of route. We note that the use of FEM is especially beneficial in such a case: because any route change potentially 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 channel.

When a node which has initiated a local repair has received Resv messages on all its new downstream interfaces, it knows that the corresponding 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 to send such a message down the old route. REDO RSVP, in contrast, 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[12] 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, in order to avoid tearing resources down 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.

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 of any use or not. This is because several conditions can prevent a Resv message from reaching the initiator:

1. There is not enough 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 last case 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 were 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 message of the 'establishment phase' of a flow (see Section 4.1), the initiator of the local repair instructs, in its next heartbeat, the neighbours down

[11] This rules is only necessary if *no explicit message identification and acknowledgment* have been introduced in the protocol specification for the Path and Resv messages.

[12] 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.

the old route to revert to classical RSVP operation on the corresponding flow. Such 'go classical' messages are forwarded in the same way as the teardown messages (see above), so that unnecessary propagation is avoided. Also, because heartbeats 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.[13]

In the case of a route change affecting unicast sessions, special attention must be paid to possible *path state instability*. Such a phenomenon can appear because, on the old portion of the route, classical nodes or REDO nodes operating in classical mode for some unicast sessions keep sending Path messages down the outdated route. This could result in the RSVP node at the 'junction' of the old and new routes receiving Path messages from multiple neighbours for the same flows. In the worst case, this instability can last as long as a state lifetime L (that is several minutes), with a node on the old route sending up to 11 'misleading' Path messages![14]

If the junction node operates in classical mode with all its upstream neighbours, this situation can result in the path state 'flipping' among different neighbours (although data is only received on the new route). However, if REDO mode is in operation with the neighbour up the new route, the situation is more serious: because no more path messages are sent down the new route after the local repair, any Path message sent from the previous neighbour on the old route could result in the Path state being wrongly reinstalled on the old route. When the path state eventually times out on the old route upstream the junction node, the states/reservations are wrongly torn down. This problem is overcome by ensuring that any state or reservation, between nodes operating in REDO mode, can only be uninstalled *following a time out or a teardown message* (*which is reliable*). Therefore, when receiving path messages for the same flows from different neighbours, a REDO node *must* at least maintain simultaneous path states for those neighbours with whom it operates in REDO mode for the concerned flows.

It should be noted that because REDO RSVP relies on local repairs to detect route changes, it cannot be applied to 'adjacent' REDO RSVP nodes connected across a non-RSVP 'cloud'. This is because a change of route within such a non-RSVP cloud would not necessarily be detected by the ingress REDO node while that route change could physically result in a change of next-hop neighbour.

[13] As soon as the reservations for that flow is completed downstream of a node, that node enters normal REDO mode with its downstream neighbours.

[14] $L \geq 3/2(K + 1/2)R$, with the minimum time between Path messages being $R/2$ and $K = 3$ by default [7].

5.4.2. Network failures

Expiration of the soft-state timer associated to 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. 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 to one of its neighbours has timed out, should refrain from sending heartbeats towards this neighbour. REDO operations can only resume between these nodes after they have been re-synchronised (see Section 5.3). 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 the 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.

On broadcast links, where a well known multicast address may be used for the exchange of heartbeats (Section 5.2), it is impossible to refrain from sending heartbeats to a particular 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 network 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. Additionally, 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, respectively, towards the senders and the 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 to faults.

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

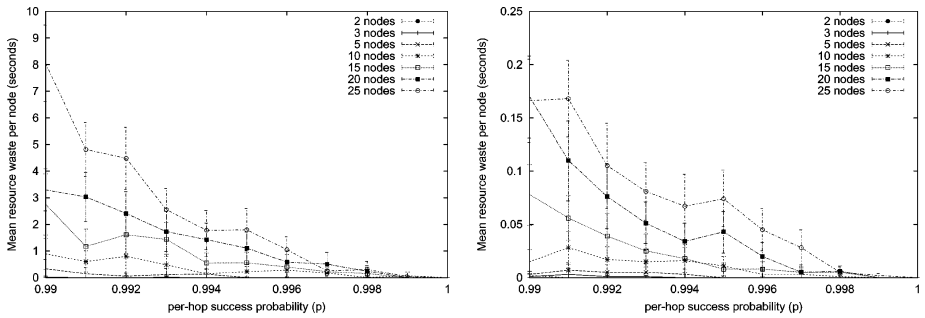


Fig. 9. Mean per flow resource waste per node. (a) With RSVP. (b) With REDO RSVP.

5.4.3. Transient failures

We call transient failure a node or REDO RSVP process failure which is neither detected by the heartbeat mechanism nor the routing protocol. This can happen when the 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 5.3). This will have the effect to force 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.

5.5. Simulation results

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 waste occurs 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 its periodic heartbeats.

The simulations presented in this section were performed under the same conditions as the ones in Section 4.2, but with an average period H of 2 s for the heartbeats. To avoid synchronisation of the heartbeats [10], 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. Fig. 9 shows 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 and is more predictable. We have also compared waste of resources on routes with a single bottleneck (Fig. 10(a)) and in the case of two wireless terminals connected by a lossless wired network (Fig. 10(b)). In all cases, we see that when the per-hop success probability is sufficiently high, REDO RSVP stops outperforming classical RSVP at resource reclamation. This was expected as flow teardown is less sensitive to the values of the per-hop success probability than flow establishment (see Section 3). Unless the per-hop loss probability is greater than 1%, unreliable teardown exhibits acceptable performance.

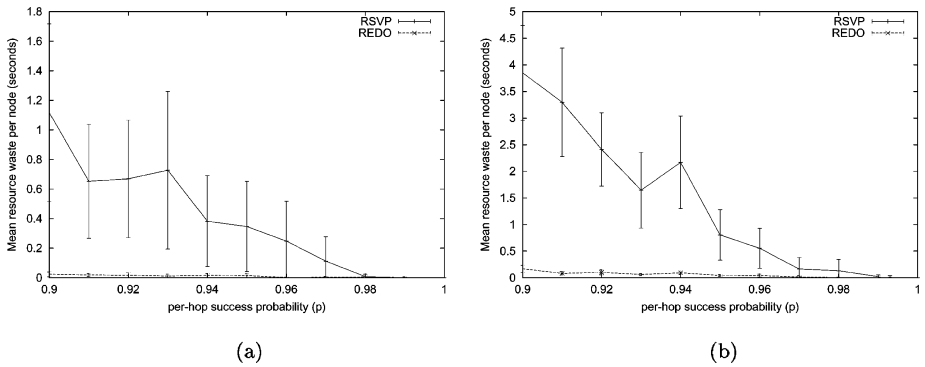


Fig. 10. Mean per flow resource waste over routes with a single bottleneck and in a wireless environment. (a) Routes with one bottleneck. (b) Wireless scenario.

6. Related work

In Ref. [17], it is proposed to define and use explicit hop-by-hop acknowledgment messages for every control message in RSVP. To improve the responsiveness of the protocol, a procedure similar to the one described in Section 4.1 is used for the re-transmission of the control messages that have not been acknowledged. Once states or reservations have been acknowledged, it is then proposed to use long refresh periods (of the order of quarter of an hour) in order to reduce the steady-state overhead.

Although this approach seems similar to ours, there is a major difference in the use of acknowledgments: the acknowledgments are used hop-by-hop. A node that has correctly received a message from one of its neighbours acknowledges it. Therefore, the semantic of these acknowledgments is weak, because, unless hop-by-hop acknowledgments are coupled with a mechanism (such as per-neighbour soft-state) to detect failures, the receipt of an acknowledgment does not mean that the initial message has reached, or will reach, its final destination. Finally, the long refresh periods will result in performance far worse than the one of classical RSVP in the following circumstances: path state instability after route changes, transient failures undetected by the routing protocol and loss of state (due to soft-state time-out) in nodes that have acknowledged the corresponding control messages.

In contrast, FEM RSVP is based on a mechanism where any form of acknowledgment is generated by the receiver of the original message (this provides a form of end-to-end notification), which covers the conditions cited above. Furthermore, as outlined in Section 4, FEM RSVP can avoid the use of explicit acknowledgment messages when reservation requirements are readily available. As this is always the case for local repairs, FEM RSVP helps in reducing the size of the message bursts that occur in those circumstances.

Very recently, an Internet Draft [2] has been introduced at the IETF describing changes to RSVP aimed at reducing its steady-state message overhead. As in our work, the basic principle of the proposal is also the use of per-neighbour soft-state. However, the details of the proposals are quite different from REDO RSVP, because along with the per-neighbour soft-state, the authors introduce fundamental changes to the original philosophy of RSVP by proposing the adoption of message identification for every message, as well as explicit acknowledgment, and acknowledgment request, for every message type. Furthermore, before suppressing refreshes on a flow, a node indicates its intention through a new explicit notification.

In contrast, REDO RSVP was designed under the constraint not to change the format of existing messages and not to introduce changes to the specification of the protocol unless absolutely unavoidable. As a result, REDO RSVP appears slightly more complex than the per-neighbour soft-state mechanism in Ref. [2] where, for example, a synchronisation phase is not required. However, it should be noted that the simplification of that particular mechanism has been achieved by 'redistributing' part of the complexity throughout the protocol with the introduction of the explicit message identifications, acknowledgments and notifications.

However, the use of explicit acknowledgment in Ref. [2], coupled with the per-neighbour soft-state mechanism, allows for the use of hop-by-hop acknowledgments at flow establishment, which should yield performance equivalent to our FEM mechanism but with a smaller increase in the number of control messages sent (because re-transmissions only occur on one hop at a time). Furthermore, the use of explicit teardown acknowledgments avoids the coupling of the reliable teardown mechanism with the per-neighbour soft-state mechanism. These properties are achieved through a thorough, and hence more complex, revision of the protocol specification than REDO RSVP.

7. Conclusion

We have modelled the resource reservation establishment mechanisms in RSVP and have shown that it is very sensitive to the values of the per-hop probability measured between RSVP processes. We have also shown that, to a lesser extent, this sensitivity affects resource release too. Consequently, there is a need for a signalling channel in the Internet, to protect as much as possible the value of the per-hop probability experienced by RSVP messages from being adversely influenced by data traffic. Furthermore, because even the best provisioned signalling paths are never totally lossless, we have presented the principles of FEM, a Fast 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 in most circumstances. In the case of local repairs, FEM can even achieve better performance without any increase of message overhead.

To achieve the very high values of the per-hop success probabilities required for RSVP to provide acceptable performance, routers would have to treat RSVP messages as high priority, to minimise losses and their dramatic effects. Thanks to the robustness of FEM, such a constraint can be relaxed.

FEM introduces a slight increase in protocol state. However, we anticipate that RSVP will only be operated on a 'per-flow' basis in areas of the Internet (in particular at the edges) where the concentration of flows is low. Elsewhere, RSVP will be operated in an aggregation context which greatly reduces the state scalability problem. Consequently, the small state increase in FEM RSVP should be of little consequence.

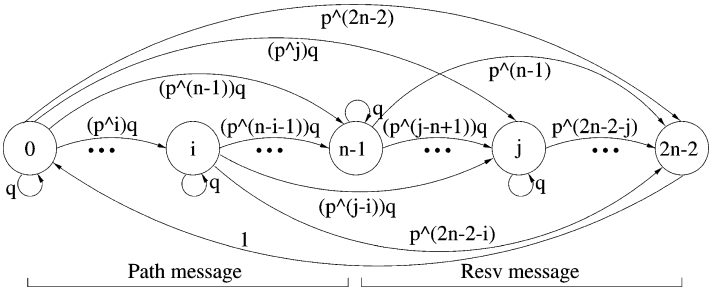


Fig. 11. Modified chain.

The underlying principles of FEM are very simple. The resulting modest increase of protocol complexity is negligible compared with the achieved gains in performance. Furthermore, the value of the initial establishment period can be chosen depending on the packet drop rate observed between adjacent FEM RSVP nodes to minimize the associated message overhead. Indeed, on a (wired) LAN where packet loss is a rare event, T_0 could be set to a few seconds, while on an often congested WAN link, this value could be set to just a couple of hundreds of milliseconds.

The soft-state mechanism in RSVP is a simple way to deal with some exceptional conditions in the network. Unfortunately, it does not provide a good response to every error condition that can be encountered (see Section 5). 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, for instance, have 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 steady-state overhead that jeopardizes 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 per-flow, the steady-state overhead is reduced and is independent of the number of flows in the network.

REDO RSVP actually 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. As 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 the synchronisation mechanism described in Section 5.3. Because such a synchronisation must only be performed occasionally, we believe that the added operational complexity in REDO RSVP is marginal compared to its demonstrated benefits. However, FEM, which is an integral part of REDO RSVP, can also be deployed on its own as an amendment to classical RSVP.

Finally, our goal was to study principles for solutions, not to redesign RSVP completely. We hope to have demonstrated that careful analysis and understanding of RSVP can lead to the introduction of minor extensions that nevertheless bring major performance improvements. Of course, more thorough modifications can lead to even better performance, but in each case, the extent of the work to be carried out should always be balanced against the added benefit.

Acknowledgments

The authors wish to express their gratitude to Dimitris Pezaros for his valuable help and support during the course of this work.

Appendix A. Average number of periods to reservation

To obtain the average number of periods required to establish a reservation for a flow, we consider the modified Markov chain depicted in Fig. 11. The only difference between this chain and the one discussed in Section 3 is that the last state is no longer absorbing and instead we have a sure transition back to the beginning of the chain. Consequently, the chain is now *ergodic* and we can thus obtain a steady-state solution for the long-term behaviour of the chain.

From this steady-state solution, the mean recurrence time of state $2n - 2$ is readily available. The average number of periods to establishment is then obtained by noting that this quantity is the average number of transitions to move from state 0 to state $2n - 2$, which is the mean recurrence time of state $2n - 2$ minus the average number of transitions from state $2n - 2$ to state 0 (that is 1).

The transition probability matrix is now:

$$\mathbf{P} = \begin{pmatrix} q & pq & p^2q & p^3q & \cdots & p^{2n-3}q & p^{2n-2} \\ 0 & q & pq & p^2q & \cdots & p^{2n-4}q & p^{2n-3} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 0 & \cdots & q & p \\ 1 & 0 & 0 & 0 & \cdots & 0 & 0 \end{pmatrix}.$$

The steady-state solution is the normalised solution of the following system of linearly dependent equations

$$\mathbf{xP} = \mathbf{x},$$

where $\mathbf{x} = (x_0, x_1, \ldots, x_{2n-2})$.

If we let $x_0 = 1$, then we get $x_{2n-2} = p$. Furthermore, by inspection of matrix **P**, we see that for $0 < i < 2n - 2$, we have

$$x_i = x_{i-1}p + x_iq = x_{i-1}.$$

Since

$$x_0pq + x_1q = x_1,$$

it follows that $x_1 = q$, and we have a solution for the system of equations.

The steady-state probabilities are then given by

$$\pi_j = \frac{x_j}{\sum_k x_k}, \quad j = 0, \ldots, 2n - 2.$$

In particular,

$$\sum_{k=0}^{2n-2} x_k = 1 + (2n - 3)q + p,$$

and we have

$$\pi_{2n-2} = \frac{p}{1 + (2n - 3)q + p}.$$

The mean recurrence time for state $2n - 2$ is then [12] p. 482:

$$E[T_{2n-2}] = \frac{1}{\pi_{2n-2}} = (2n - 2)\frac{q}{p} + 2.$$

As explained above, the average number of periods to establish a reservation is then:

$$E[S] = E[T_{2n-2}] - 1 = (2n - 2)\frac{1 - p}{p} + 1.$$

References

[1] D. Awduche, L. Berger, D.-H. Gan, T. Li, V. Srinivasan, G. Swallow, RSVP-TE: extensions to RSVP for LSP Tunnels, Internet Draft draft-ietf-mpls-rsvp-lsp-tunnel-07, IETF, August 2000, Work in progress.

[2] L. Berger, D.-H. Gan, G. Swallow, P. Ping, F. Tommasi, S. Molendini, RSVP refresh overhead reduction extensions, Internet Draft draft-berger-rsvp-refresh-reduct-05, IETF, June 2000, Work in progress.

[3] Y. Bernet, P. Ford, R. Yavatkar, F. Baker, L. Zhang, R. Speer, M. Braden, B. Davie, J. Wroklawski, E. Felstaine, A framework for integrated services operation over Diffserv networks, RFC 2998, IETF, November 2000.

[4] S. Berson, S. Vincent, Aggregation of Internet integrated services state, Proceedings of Sixth International Workshop on Quality of Service (IWQoS'98), Napa, CA, USA, May 1998, pp. 26–28.

[5] D. Black, S. Blake, M. Carlson, E. Davies, Z. Wang, W. Weiss, An architecture for differentiated services, RFC 2475, IETF, December 1998.

[6] R. Braden, D. Clark, S. Shenker, Integrated services in the Internet architecture: an overview, RFC 1633, IETF, June 1994.

[7] R. Braden, L. Zhang, S. Berson, S. Herzog, S. Jamin, Resource ReSerVation Protocol (RSVP)—Version 1, Functional specification, RFC 2205, IETF, September 1997.

[8] L. Breslau, S. Shenker, Best-effort versus reservations: a simple comparative analysis, ACM Comput. Commun. Rev. 28 (4) (1998) 3–16.

[9] I. Cidon, A. Khamisy, M. Sidi, Analysis of packet loss processes in high speed networks, IEEE Trans. Info. Theory 39 (1) (1993) 98–108.

[10] S. Floyd, V. Jacobson, The synchronization of periodic routing messages, IEEE/ACM Trans. Network. 2 (2) (1994) 122–136.

[11] O. Fourmeaux, S. Fdida, Multicast for RSVP switching, Telecommun. Syst. J. 11 (1–2) (1999) 85–104.

[12] A. Leon-Garcia, Probability and Random Processes for Electrical Engineering, Second ed., Addison-Wesley, Reading, MA, 1994.

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

[14] A. Mankin, F. Baker, B. Braden, S. Bradner, M. O'Dell, A. Romanow, A. Weinrib, L. Zhang, Resource ReSerVation Protocol (RSVP)—Version 1, Applicability statement: some guidelines on deployement, RFC 2208, IETF, September 1997.

[15] L. Mathy, D. Hutchison, S. Schmid, G. Coulson, Improving RSVP for better support of internet multimedia communications, IEEE Multimedia Systems'99, vol. 2, IEEE Computer Society, Silver Spring, MD, 1999, pp. 102–106.

[16] A. Odlyzko, The Internet and other networks: utilization rates and their implications, Proceedings of 26th Telecommunications Policy Research Conference, October 1998, Available from www.research.att.com/~amo/doc/networks.html.

[17] P. Pan, H. Schulzrinne, Staged refresh timers for RSVP, Proceedings of Second Global Internet Conference, Phoenix, AZ, USA, November 1997.

[18] L. Zhang, S. Deering, D. Estrin, D. Zappala, RSVP: a new Resource ReSerVation Protocol, IEEE Network 7 (5) (1993) 8–18.