

# QoS Path Management with RSVP

Roch A. Guérin, Sanjay Kamat, and Shai Herzog,

**Abstract**—This paper addresses the issue of QoS path management in IP networks. We describe a proposal aimed at allowing management through the RSVP [3] protocol of paths selected by a QoS routing algorithm such as those of [1], [6]. The goals of the proposal are to allow efficient management of such QoS paths with a minimal impact to the RSVP protocol and the existing routing infrastructure. Basic features of the approach include leveraging of RSVP soft state mechanisms, and simple extensions to enable soft pinning (sticking) of paths selected by the QoS routing algorithm. In addition, the proposal addresses the issue of preventing the formation of data path loops, and of avoiding potential race conditions.

**Keywords**—QoS Routing, RSVP, Path Management.

## I. INTRODUCTION

The goal of QoS routing is to select paths for flows with QoS requirements, in such a manner as to increase the likelihood that the network will indeed be capable of satisfying them. For example, in IP networks, QoS routes can be computed using extended versions of existing link state algorithms such as OSPF [1], [6]. In general, though, the use of QoS routing algorithms has a number of implications above and beyond what is required when using standard Internet routing algorithms.

First, a specific mechanism needs to be used to identify flows with QoS requirements, so that they can be assigned to the corresponding QoS routing algorithm. In this paper, we assume that the RSVP protocol [5], [3] is used for that purpose.<sup>1</sup> Specifically, RSVP PATH messages serve as the trigger to query QoS routing. Second, because of variations in the availability of resources in the network, routes between the same source and destination and for the same QoS, may often differ depending on when the request is made. However, it is important to ensure that such changes are not always reflected on existing paths. This is to avoid potential oscillations between paths, and limit changes to cases where the initial selection turns out to be inadequate.

As a result, some state information needs to be associated with a QoS path to determine its current validity, i.e., should the QoS routing algorithm be queried to generate a new and potentially better route, or does the current one remain adequate. We say that a path is “pinned” when its state specifies that QoS routing need not be queried anew, while a path is considered “unpinned” otherwise. The main issue is then to define how, when, and where route pinning and unpinning is to take place. In our context, where the RSVP protocol is used as the vehicle to request QoS routes, we also want this process to be as synergistic as possible with the existing RSVP state management. In particular, our goal is to support pinning and unpinning<sup>2</sup> of routes in a manner consistent with RSVP soft states while requiring minimal changes to the RSVP processing rules.

R. A. Guérin is with the IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598. [guerin@watson.ibm.com](mailto:guerin@watson.ibm.com).

S. Kamat is with the IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598. [sanjay@watson.ibm.com](mailto:sanjay@watson.ibm.com).

S. Herzog is with the IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598. [herzog@watson.ibm.com](mailto:herzog@watson.ibm.com).

<sup>1</sup>Readers are assumed to be familiar with the RSVP protocol.

<sup>2</sup>Note that we use the terms pinning and unpinning in a generic sense, and in particular do not specifically associate them with a hard state concept.

It should be noted that some changes are unavoidable, especially to the interface between RSVP and routing. Specifically, QoS routing requires, in addition to the current source and destination addresses, at a minimum, knowledge of the flow’s traffic characteristics (TSpec), and possibly also service types (as per the information in the AdSpec), PHOP, IP TTL value, etc. In this paper, we assume that the information provided by RSVP to QoS routing includes at least the sender TSpec in addition to the source and destination addresses. While such changes seem unavoidable, our goal is again to keep them as small as possible and also to avoid any change to the existing RSVP message format.

Specifically, we assume interactions between RSVP and routing that are very similar to what is currently defined. During the processing of RSVP PATH messages, RSVP queries (QoS) routing to obtain the next hop(s) for forwarding the PATH message. The PATH message is then forwarded on the interface(s) returned by (QoS) routing. As mentioned before, the sender TSpec is part of the information made available to routing, and a QoS routing algorithm such as in [1], [6] selects the next hop along a path to the destination that is most likely to support the flow specified by the sender TSpec. Thus, forwarding the PATH message along this next hop should improve the likelihood of the reservation request succeeding during RESV message processing. In particular, RESV messages, if any, propagate as before in the reverse direction of the PATH messages and attempt to reserve the required resources along the path delineated by PATH messages.

In this context of hop-by-hop routing, there are two main issues associated with the pinning and unpinning of QoS paths.

1. Detection of loops that may be caused by inconsistencies in the QoS routes returned by QoS routing at different nodes. Such inconsistencies are typically transient, but it is important that the pinning of a path does not result in the formation of permanent loops.
2. Query of a new QoS route in case of failures. An example of such failures is a reservation failure because the RESV message arrived substantially later after the QoS route was initially selected. Other failures include the usual link failures, and in general it is important to allow QoS routing to become aware of the failure and select a better route if one is available.

The QoS path management scheme proposed here addresses the above two issues based on the following two design rules:

(i) What is pinned is a “path” taken by a specific RSVP flow and not a “route” as computed by the routing algorithm. Hence pinning and unpinning could be considered as RSVP domain operations, and be completely independent of the specific QoS routing algorithm used.

(ii) Path pinning and unpinning is kept “soft” by tying it to the existing RSVP soft state mechanism. In other words, we rely on existing RSVP refreshes and time-out mechanisms to detect the state changes that trigger pinning and unpinning of paths. In addition, such changes are triggered only on the basis of current

RSVP state information.

In the rest of this paper, we review how the above two rules translate into the conditions and processing rules for pinning and unpinning paths, that address the problem of loops while also enabling reactions to failures. Some examples that illustrate the operation of these rules in different configurations are also provided.

## II. PATH MANAGEMENT (PINNING AND UNPINNING) STATE AND RULES

The state of a QoS path as maintained by RSVP consists of a flag that is used to indicate whether the path is currently pinned or not. Specifically, a pinned path means that QoS routing need not be queried for a new path (next hop) for forwarding a PATH refresh. The rules for pinning and unpinning paths are as follows:

1. Paths get pinned during processing of PATH messages.
2. Paths get unpinned when
  - (a) corresponding path states are removed (time-out or PATH\_TEAR),
  - (b) some of the parameters received in PATH messages change,
  - (c) a local admission control failure error is detected after receiving a RESV message,
  - (d) a PATH\_ERR with a specific error code is received, or
  - (e) failure notification of a local link belonging to the path is received.

The above rules translate into a number of small changes to the existing RSVP processing rules. These changes can be grouped in two categories and are described next.

(i) Modifications to RSVP/Routing interface to make use of QoS routing:

Currently, RSVP acquires routing entries using its asynchronous query-response interface to routing [4]. Route query is of the form

*Route\_Query*( [SrcAddress], DestAddress, Notify\_flag )

and Routing responds with OutInterface (or OutInterface\_list in case of a multicast connection).

In order for RSVP to interact with a QoS routing algorithm, QoS\_Route\_Query needs to also include (at a minimum) the sender\_TSpec, so that it is now of the form

*Route\_Query*( [SrcAddress], DestAddress, TSpec, Notify\_flag )

and again Routing responds with OutInterface (or OutInterface\_list in case of a multicast connection).

Another small difference with the current interface is that the Notify\_flag should always be set to True. This is because there will be no Route\_Query to QoS routing in the case of pinned paths. Hence, it is important that a trigger be provided to unpin the path in case of failure. However, note that QoS routing will only generate an asynchronous Route\_Change callback to RSVP in the case of the failure of a local (to the router) link currently used by the QoS path.

(ii) Modifications to message processing rules in the context of pinning/unpinning of paths.

– PATH message processing: When receiving the *first* PATH message, RSVP determines that no PATH\_state exists for the flow. It then queries QoS routing to obtain the next hop along

the “best” available path. This next hop is stored as part of the PATH state with its pinned flag set.

Upon receiving a PATH refresh, RSVP checks for changes in PATH state that are of relevance to QoS routing. In particular, it checks for changes in PHOP and the IP TTL value. If there are no changes and the current next hop is indicated as pinned, it will be used to forward the next PATH refresh. If the PATH state has changed or the current next hop is marked as unpinned, RSVP queries QoS routing again to obtain (and pin) a new next hop that is to be used when forwarding the next PATH refresh. Similarly, at the time when a PATH refresh is to be sent, RSVP checks if the current next hop is pinned or not. If it is, it is used to forward the PATH refresh. Otherwise, QoS routing is again queried to obtain (and pin) a new next hop.

The unpinning of the path upon detecting changes in either the PHOP or the IP TTL value of an incoming PATH message is used to infer that significant route changes have taken place. These state changes alert a router of possible unpinning actions at upstream routers or of potential looping conditions. This is further explained in Section III-B.

– PATH\_TEAR Processing: Processing is similar to what is currently done. PATH and RESV states are removed.

– RESV Processing: The only change needed is for the case when the resource reservation attempt fails. As currently specified, a RESV\_ERR message with “admission control failure” error code is still sent downstream in such instances. However, some additional processing is needed in order to enable selection of a better path in case one exists. This starts with the unpinning of the current next hop, and then proceeds in either one of two ways: attempt *local* repair of the QoS path or not.

In case local repair is attempted, RSVP queries again its local QoS routing table. If a different next hop is returned, i.e., the reservation may now succeed, then local repair is attempted by pinning the new next hop and sending a PATH message along the new route. If the same next hop is returned, then local repair has failed. In this case or when local repair is not attempted, the current next hop is then unpinned in the PATH state (but kept). Furthermore, a PATH\_ERR message is sent upstream with a new QoS\_Path\_Failure Error Code and an associated Error Value specifying that the type of error was “Requested QoS unavailable”. As described below, the receipt of a PATH\_ERR message with the QoS\_Path\_Failure Error Code triggers unpinning of the next hop information at upstream router. This ensures that QoS routing will be queried at the time of the next PATH refresh, so that a better path, if one exists, can be identified.

– Route\_Change Notification processing: A Route\_Change notification is triggered when QoS routing detects that a local link currently used by a QoS path failed. Upon receiving such a notification, RSVP immediately unpins the current next hop. As in the case of reservation failure, RSVP can then first attempt local repair, i.e., query QoS routing for a new next hop. If a new next hop is returned by QoS routing, RSVP uses it to replace the previous next hop, marks it as pinned, and forwards the PATH message towards the new next hop. If QoS routing responds that no path to the destination is available or if local repair is not attempted, RSVP sends upstream a PATH\_ERR message with the QoS\_Path\_Failure Error Code and an Error Value specifying

“Link failure”.

– **PATH\_ERR Processing:** The only modification is, as mentioned above, to recognize the new `QoS_Path_failure` Error Code and unpin the associated next hop. This forces a fresh QoS route query during the processing of the next PATH refresh.

– **RESV\_ERR Processing:** There are no changes to RESV\_ERR processing.

### III. DISCUSSIONS AND EXAMPLES

#### A. Impact of QoS routing on the data path

The use of QoS routing only affects the choice of a data path and not how the actual forwarding of data packet takes place. Nevertheless, there is an important aspect that needs to be noted. Specifically, while PATH messages are immediately forwarded onto the next hop returned by QoS routing, the same need not apply to data packets. This is because of the potential for transient loops in QoS paths. Forwarding PATH messages on a QoS path that may contain loops has minimal impact on the routers and is actually useful to detect and eliminate loops (more on this below). However, depending on how fast loops can be resolved, forwarding data packets on a QoS path may be best deferred until the absence of a loop has been verified.

As a result, it is proposed that modification of the packet classifier in the forwarding loop that will result in data packets being sent towards the next hop specified by QoS routing, be deferred until the time a RESV message is received. As discussed below, the receipt of a RESV message also implies that loops are not present in the QoS path. Note that the update of classifiers at the time of receipt of a RESV message is consistent with when this is done using the current default routing. The main difference is that the actual flow of data packets may not start following the QoS path until after the classifier has been updated in the first node where the default and the QoS paths start differing. Such a scenario is illustrated in Figure 1, where the best-effort path between the sending host SH and the receiving host RH is SH...R1-R2-R4-R5-R6...RH, while the QoS path is SH...R1-R2-R3-R5-R6...RH. In that case, data packets will not be sent over the QoS path until after the classifier at router R2 has been updated to forward them towards R3 instead of R4.

There are some drawbacks with the above approach, e.g., inability to take advantage of partial reservations in some instances. For example, if the reservation fails on the link between R2 and R3, then the reservation in place on the link between R3 and R5 will be of no use to the data packets from SH since they will not be forwarded along the path through R3. However, note that they will be able to benefit from the reservations on the links between R5 and R6 and R6 and RH since this portion of the QoS path coincides with the the best-effort path.

There are a number of ways to address the above problem. One possibility, that may be acceptable if transient loops are detected and removed quickly, is to actually update classifiers upon receipt of a PATH message (or a certain number of PATH messages, when it appears that the QoS path is stable and loops are not present) instead of a RESV message. Another more comprehensive alternative is to couple this process with the handling of policy information. Such a coupling is a natural step as the ability for users to specify how much of a partial reservation is

acceptable to them, i.e., does one need to look for another path, is really a policy issue. In order to support such a coupling, policy data objects would have to be included in PATH, RESV, RESV\_ERR, and PATH\_ERR messages, in order to enable the local policy control module [2] to assess the suitability of a QoS path. The discussion and description of such an approach is the subject of future work.

#### B. Handling and prevention of loops

As mentioned before, pinning of a QoS path introduces the possibility of loops when nodes use inconsistent information to make QoS routing decisions. Therefore, it is important to detect such occurrence and unpin the selected QoS path (next hop), so that loops can eventually be eliminated once nodes have converged to consistent information.

We now illustrate how the processing rules described earlier are successful at detecting and eliminating loops. Recall that a change in the PHOP or the IP TTL field in the received PATH message of a flow indicates to the router that an upstream router has effected a path change for the particular flow. By performing local unpinning of the flow on seeing these changes, it is ensured that routing will be queried afresh to use the most current routing information before forwarding the PATH message. Additionally, as the following examples show, such unpinning rules help prevent pinning of looping paths.

The first scenario below illustrates a transient looping scenario where the change in PHOP alone would be sufficient to detect the loop.

In scenario 1 shown in Figure 2, we assume that R2 (and possibly other routers along RX...RY) has routing (metrics) information that is not consistent with that of R1. Router R1 identified R2 as the next hop on the best QoS to RH on the basis that this QoS path was R1-R2-R3...RH. However, because of inconsistent metrics information, R2 identifies RX as the next hop on the best QoS path to RH, so that the PATH message forwarded to RX eventually comes back to R2 from RY. As a result, a temporary loop is created and would remain in effect for as long as the next hop information remains pinned. However, because the PATH message arriving from RY to R2 comes with a different PHOP value, this triggers unpinning of the next hop information at R2. QoS routing is, therefore, queried anew the next time R2 has to forward a PATH message for the flow, so that the loop is eventually (once the routing/metrics information at R2 becomes consistent with that at R1) removed.

Note that how quickly the loop is removed depends on the RSVP timer values and also how they are synchronized between the different routers. For example, R2 may receive a PATH refresh from RY just after having received one from R1. This would not affect the unpinning of the next hop (RX), but means that RY will be kept as the PHOP for the flow, at least until the next refresh from R1 (if the routing information at R2 is now consistent, PATH messages will be forwarded to R3 and not RX, so that RY will not receive PATH refreshes and, therefore, not send any itself). As a result, a RESV from RH may initially be forwarded to RY instead of R1. However, this will not create a reservation loop, since when the RESV returns to R2 from RX, R2 determines that RX is not the correct NHOP for the flow.

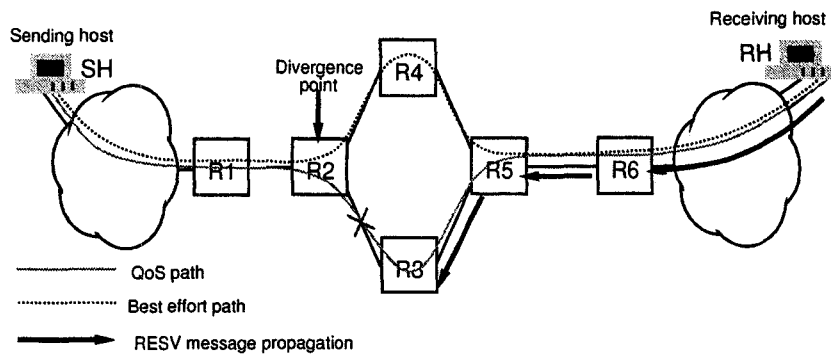


Fig. 1. Impact of differences between best-effort and QoS paths in the case of partial reservations.

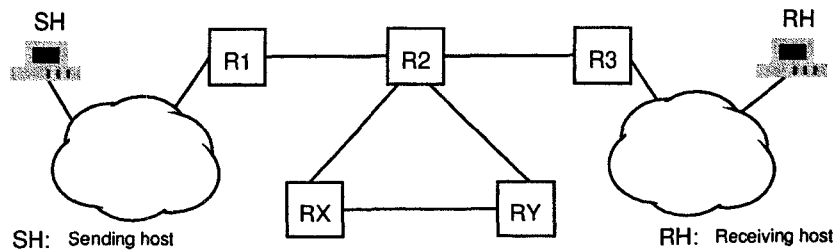


Fig. 2. Scenario 1.

Scenario 1 showed the case where the router which caused the loop was also the one where the loop closed. Scenario 2 shown in Figure 3 considers the case where the loop does not close at the router that caused it in the first place.

Here the “correct” QoS path from SH to RH is SH...R1-R2-RX-R3....RH. However, router RX has inconsistent routing information based on which it identifies RY as the best next hop. This eventually creates a loop closing at R2 as RY identifies R2 as the best next hop to RH. In this case, RX would not see any change in PHOP, and therefore this criterion cannot be relied upon to unpin the next hop state at RX. However, RX will receive PATH messages with varying IP TTL values, and this will then trigger unpinning of the path so that when RX routing information eventually gets updated, it can then identify the correct next hop, namely R3.

Clearly, any looping condition is characterized by a change in IP TTL and hence unpinning based on this change is sufficient to break loops. However, unpinning on seeing a change in PHOP is also a useful mechanism by which a downstream router can be alerted of a path change effected by an upstream router.

### C. Handling of race conditions in resources allocation

The problem of race conditions, i.e., having several paths competing for the same resources, is intrinsic to the distributed nature of QoS routing decisions. This is not unique to the model assumed in this paper, e.g., PNNI routing in an ATM network faces a similar quandary, but the problem can be exacerbated by the potential time lag between the selection of a path and when reservation of resources is actually attempted, i.e., upon receipt of a RESV message. As a result, it is important to ensure that QoS routing and the associated mechanisms for establishing QoS paths is sufficiently flexible to handle such race conditions and change paths if and when required.

In this section, we illustrate how the mechanisms described in this paper handle such situations in the context of a representative example.

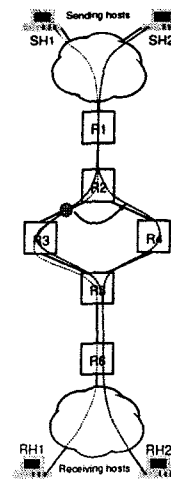


Fig. 4. Competing Flows.

Consider the topology shown in Figure 4 and assume that link R2-R3 can accommodate only one of the two flows at a time, and that the segment R1-R2-R3-R5-R6 is pinned for both the flows before either RH1 or RH2 has generated a RESV message. Next, assume that RH1 is the first to generate a RESV message and that its reservation succeeds on the entire path. When RH2 generates its own RESV message, its reservation will succeed at R6, R5, and R3, but fail at R2. Based on the processing rules outlined above, R2 will then unpin its next hop state in addition to forwarding a RESV\_ERR downstream. Note, that the path and reservation states for the flow remain intact on the downstream segment.

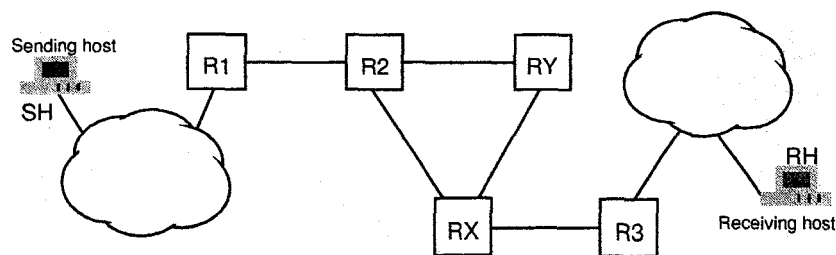


Fig. 3. Scenario 2.

If R2 attempts local repair next, it will then query QoS routing. If the same next hop, R3, is returned, R2 determines that local repair has failed and sends a PATH\_ERR message upstream with Error Code QoS\_Path\_failure and Error Value "Requested QoS unavailable". Receipt of this PATH\_ERR message then triggers unpinning of the next hop state at all upstream nodes, i.e., at R1 (and SH1). This in turn will trigger querying of QoS routing at the time of the next PATH refresh. This ensures that if at some time resources become available on the path through R4 and this information is provided to QoS routing, then the flow from SH2 to RH2 will ultimately be routed over that path, i.e., R2 will identify R4 as its next hop for the flow. As a result, a PATH message will then be sent out on the interface to R4 so that RESV messages from R5 will eventually be forwarded towards R4, and the flows' reservation will then likely succeed. A similar process would have been followed if QoS routing had returned R4 as the best next hop, when R2 attempted to perform local repair.

Note that R2 did not issue a PATH\_TEAR to R3 at the time of the reservation failure. There are a number of reasons for this. First, the path through R3 may indeed be the best one currently available, and any reservation that has succeeded on it so far should probably not be taken down (a PATH\_TEAR would) unless the receiver itself decides so. Second, keeping the reservations in place also avoids the problem of having the resources on links from R5 to R6 and R6 to RH2 be taken by some other flows, while the flow from SH2 to RH2 is in the process of switching paths. Instead, the existing reservations on the downstream segment R5-R6...RH2 will be reused, and the path and reservation states for the flow at R3 will time out. On the other hand, keeping the reservations in place may affect the outcome of QoS routing. This is because QoS routing typically won't know, that on some links the resources actually available to the flow are higher than what is currently advertised. However, this should be mitigated by the fact that advertised resources are always inaccurate and also because the QoS routing will always return the best possible path, even if it does not think that the necessary resources are available on it.

#### IV. SUMMARY AND EXTENSIONS

In this paper, we have described a possible approach to provide QoS routing while using RSVP as the signaling protocol to setup QoS routes. The goal was to minimize changes to RSVP processing rules, rely as much as possible on the existing RSVP soft states, and do so without limiting the benefits of QoS routing. The approach taken relies on hop-by-hop routing decisions

similar to those used by the current best-effort routing. Potential routing loops are detected and eliminated through simple monitoring of RSVP state information. Similarly, reservation and link failures are handled using existing RSVP error messages with new error codes specifying the type of routing failure.

It should be noted that while the approach described in this paper is not dependent on the use of a particular QoS routing protocol, it has been essentially targeted at unicast routing and possibly multicast routing protocols such as MOSPF, where consistent routing decisions can be made at all nodes based on common global knowledge. Other multicast routing protocols may have additional requirements that necessitate additional or different processing rules.

In the approach presented in this paper, the bulk of the responsibility for QoS path management, i.e., pinning and unpinning of next hop information, lies with RSVP. This was motivated in part by the need to couple path management with the RSVP soft state management, and by the close relation to existing RSVP processing rules. However, it can be argued that the path management functionality belongs to routing rather than RSVP. Such function placement can be realized by broadening the RSVP-Routing interface and at some increased cost of replicating some of the RSVP information in routing. Extending RSVP to carry opaque routing objects can also simplify some of the QoS path management issues as it would facilitate reservation set up along explicit routes. We are currently exploring these extensions.

**Acknowledgments** The authors would like to thank Ariel Orda, Ping Pan, and Dimitrios Pendarakis for helpful feedback on early versions of this proposal.

#### REFERENCES

- [1] R. Guérin, A. Orda, and D. Williams. QoS Routing Mechanisms and OSPF Extensions. In *IEEE-GLOBECOM*, Phoenix, Arizona, November 1997.
- [2] S. Herzog. Local Policy Modules (LPM): Policy Control for Resource Reservation Protocols. (draft-ietf-rsvp-policy-lpm-01.[ps,txt]). INTERNET-DRAFT, November 1996. Work in progress.
- [3] R. Braden (Ed.), L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource reSerVation Protocol (RSVP) version 1, functional specification (draft-ietf-rsvp-spec-13.ps). INTERNET-DRAFT, July 1996. Work in progress.
- [4] D. Zappala. RSRR: A Routing Interface for RSVP (draft-ietf-rsvp-routing-01.ps). INTERNET-DRAFT, November 1996. Work in progress.
- [5] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala. RSVP: A New Resource Reservation Protocol. *IEEE Network*. (U.S.A.), September 1993.
- [6] Z. Zhang, C. Sanchez, B. Salkewicz, and E. Crawley. Quality of Service Extensions to OSPF or Quality of Service Path First Routing (QOSPF). INTERNET-DRAFT, June 1996. Work in progress.