Internet Engineering Task Force R. Guerin/S. Blake/S. Herzog
INTERNET DRAFT IBM/IBM/IPHighway
21 November 1997

Aggregating RSVP-based QoS Requests
draft-guerin-aggreg-rsvp-00.txt

Status of This Memo

Abstract

This document describes issues and approaches related to aggregation of QoS requests, when RSVP [BZB+97] is the protocol used to convey such requests. Aggregation is an important component to provide scalable QoS solutions, especially in the core of the backbone where the sheer number of flows mandates some form of aggregation. However, aggregation needs to be provided without impacting the ability to provide end-to-end QoS guarantees to individual flows. In this document, we review some of the main goals of aggregation and describe possible solutions, that do not preclude support for end-to-end QoS guarantees. Those solutions are targeted at unicast flows as we expect them to represent a large fraction of the flows requesting reservation, and hence to be the main contributors to potential scalability problems with RSVP.

Contents

1. Introduction

As described in [Bra97], there are several facets to the support of QoS in the Internet. The aspect of QoS aggregation with RSVP falls primarily in the areas of ‘‘Control Model’’, and to some extent ‘‘Scope’’, as they are identified in [Bra97]. Specifically, the focus of QoS aggregation is on both the granularity of QoS guarantees, and their extent, i.e., from where to where.

In RSVP, the granularity of a QoS request is determined through filters that specify destination address and port number, as well as source address and port number in some instances (see [BZB+97] for details). This corresponds to a very fine granularity of QoS guarantees, i.e., per flow, and while this provides end-users with accurate control, it can also translate into a substantial overhead for the network. This is particularly true for backbone links, where the sheer number of flows (there are 37,500 64kbps flows on an OC-48 link) can introduce a scalability problem. Similarly, the scope of RSVP QoS requests is end-to-end, i.e., from application to application, and while this does again provide end-users with maximum control, it can also impose substantial overhead. For example, a network administrator may want to reserve a certain amount of bandwidth to interconnect two sites across the network of an ISP. This is not readily feasible under the current RSVP specifications, which require that reservations be setup and managed between all pairs of end-users in the two sites. A possible alternative is to establish an RSVP ‘‘tunnel’’ between the two sites, and we discuss this option, but it has the disadvantage of additional encapsulation overhead and processing.

As a result, the issue of QoS aggregation in the context of RSVP has two major components. The first, is an extension to RSVP to support ‘‘aggregate’’ QoS requests, i.e., on behalf of a set of flows rather than individual flows. For example, the set of flows to which an aggregate request would apply, could correspond to traffic between a given source subnet and a given destination subnet. Support for such aggregate requests is not available from the current RSVP specifications, and would require the definition of new filter specifications. One possible example are the CIDR prefix based filters suggested in [Boy97]. The introduction of such extensions is certainly key to increasing the applicability of RSVP as a generic reservation protocol, but in this document we instead focus on the second and more immediate aspect of QoS aggregation for RSVP.

Specifically, we consider the problem of aggregating a large number of individual RSVP requests to improve scalability, e.g., on backbone links, without precluding support for individual QoS guarantees where feasible, e.g., on low speed links and local networks. In other words, the focus of QoS aggregation in this document, is to provide the means for ensuring individual end-to-end QoS guarantees, but without requiring that awareness of individual flows be maintained on each and every segment of their path. This is an important issue as the need for maintaining and updating a large number of individual RSVP flow states has been often mentioned as a major obstacle to the widespread deployment of RSVP. The goals of this document are, therefore, to review and address the potential scalability problems that have been identified with the CURRENT RSVP specifications, and propose possible solutions.

The rest of this document is structured as follows. In Section 2, we first describe a sample scenario illustrating the constraints and aspects of QoS aggregation with RSVP. In Sections 3 and 4, we identify specific goals when supporting QoS aggregation for RSVP, and propose possible aggregation solutions to achieve them.

2. Sample Scenario and Requirements for QoS Aggregation

Consider the network topology of Figure 1. It consists of three separate AS, with the two edge AS (AS1 and AS3) corresponding to local AS and the middle one (AS2) representing a backbone interconnecting the two. For the purpose of our discussion on QoS aggregation, we assume that scalability is of concern only in the backbone AS2, i.e., AS1 and AS3 are capable of maintaining RSVP state information for all the individual flows that originate and terminate in them. Furthermore, and without loss of generality, we focus on RSVP flows between AS1 and AS3 that cross AS2. In that context, QoS aggregation is of concern only for AS2.

```
     AS1                    AS2                      AS3
   ________           ________________           _________
  /        \         /                \         /         \
 /          \       /                  \       /           \
|   Access   |     |      Backbone      |     |   Access    |
|  Network   |-----|      Network       |-----|   Network   |
|            |     |                    |     |             |
 \          /       \                  /       \           /
  \________/         \________________/         \_________/
```

Figure 1: Sample Network Configuration

Aggregation of individual RSVP flows through AS2 must satisfy a number of requirements, which we briefly review.

$R1$ AS2 should not have to maintain awareness of individual RSVP flows between AS1 and AS3. Instead, AS2 should be able to map individual RSVP flows onto few internal service ''classes''.

$R2$ AS2 should ensure that it satisfies the QoS requirements of individual RSVP flows, e.g., the resources allocated to a service class in AS2 should at least be equal to the aggregate resources required by all the individual flows mapped onto it.

$R3$ Isolation between flows should be maintained in AS2, i.e., even when flows are aggregated into a common service class, the excess traffic of one flow should not affect the performance guarantees of another flow.

$R4$ Aggregation in AS2 should not prevent support for individual flow reservations in AS1 and AS3.

Requirement $R1$ is the core scalability requirement expressed by AS2. It basically states, that because QoS support within AS2 is provided through much coarser mechanisms than the control and allocation of resources to individual RSVP flows, of which there could be way too many, it is necessary for individual RSVP flows to be mapped onto one of the internal class-based mechanisms supported by AS2. Coarser class-based mechanisms are usually mandated by the speed of the backbone links, where the time available for making packet forwarding and scheduling decisions is often not sufficient to accommodate per flow operations. In addition to the constraint on forwarding and scheduling decision, there is a similar limitation on the amount of control information that a backbone node is capable of maintaining and updating. Specifically, maintaining path and reservation control blocks for individual flows may not be not practical in AS2.

Requirements $R2$ and $R3$ specify properties, that the mapping of individual RSVP flows onto the coarser ''classes'' of AS2 has to satisfy. First and foremost, requirement $R2$ expresses the need for some coupling between the resources (bandwidth and buffer) and level of service (priority) assigned to a class in AS2, and the aggregation of the individual RSVP flows mapped onto that class. For example, this means that the amount of bandwidth assigned to a class should be sufficient to accommodate the traffic of all the RSVP flows mapped onto it. This must remain true even as flows modify their reservations. Furthermore, requirement $R2$ also points to the fact that services classes in AS2 must be defined so as to ensure they can meet the QoS guarantees of any individual flow mapped onto them. This typically means that flows mapped onto the same service class must exhibit some level of homogeneity in their QoS requirements, or that the service class is dimensioned to meet the most stringent QoS requirements of the individual flows mapped onto it.

Requirement $R3$ is a direct result of the aggregation of individual flows. The QoS guarantees provided to an individual RSVP flow are limited to its conformant packets, i.e., packets that comply with the advertised TSpec of the flow. Checking compliance with a flow TSpec is readily achieved when per flow awareness is maintained, but is lost after flows have been aggregated. In particular, violation of an aggregate TSpec (‘‘sum’’ of individual TSpec’s) can be caused by a single non-conformant flow, but can impact the QoS guarantees experienced by all the flows that have been aggregated. As a result, some mechanism is needed to identify non-conformant packet even after flows have been merged. One possible approach is to use a ‘‘tagging’’ capability as suggested in [CW97].

Requirement $R4$ expresses the important constraint that satisfying scalability in AS2, should not come at the expense of functionality in AS1 and AS3. Specifically, the aggregation of control and data path information in AS2 should be reversible, so that reservations in AS1 and AS3 can default back to individual flows after crossing AS2. In other words, the hiding of individual flow information in AS2 should not prevent reservations at a finer level of granularity in AS1 and AS3, so that end-to-end RSVP reservations can be supported. This means that for RSVP flows, AS2 should essentially behave as a single RSVP ‘‘node’’. Reservation of resources within a node are transparent to RSVP, but should not affect end-to-end operation.

In the next sections, we qualify how the above requirements translate into specific goals to support aggregation, and also describe possible approaches to satisfy these requirements.

3. Data Path Aggregation

On the data path, the main issue is the classification of data packets to determine the level of QoS they are eligible to receive. Performing this classification on the basis of bit patterns that are specific to individual flows, i.e., source and destination addresses and port numbers, may not scale. Specifically, storing all the patterns corresponding to individual flows holding a reservation and extracting the corresponding patterns from all incoming packets, can represent a substantial per packet processing overhead. As a result, the goal of an aggregation solution is to map all the bit patterns used to classify individual flows with reservations onto a much smaller number of patterns. There are several possible approaches to achieve such a mapping.

3.1. Tunnel Based Aggregation

A first solution is to rely on RSVP tunnels. In other words, an RSVP tunnel is created between any two ingress and egress points for which there exists at least one RSVP flow across AS2. At the ingress, packets (data and control) belonging to the corresponding RSVP flows are encapsulated in IP packets with an IP destination address identifying the egress point from AS2. The egress point is then responsible for the reverse decapsulation process before forwarding packets towards their next hop. As a result of encapsulation, routers on the path in AS2 only require a single entry to classify packets from all the associated RSVP flows. The main disadvantages of this solution are the data and processing overheads associated with encapsulation, as well as the need for close synchronization with routing. Specifically, the mapping of individual RSVP flows onto a given egress point from AS2 depends on routing information, and route changes need to be closely monitored to determine if and when they affect this mapping.

In addition to these disadvantages, tunnels *alone* do not easily address the above requirement $R3$ concerning flow isolation. This is because after encapsulation, conformant packets from one flow cannot be distinguished from non-conformant packets of another flow. As a result, it is necessary to discriminate between conformant and non-conformant packets at the ingress point of a tunnel, e.g., send non-conformant packets as regular (non-encapsulated) packets through AS2. While this satisfies requirement $R3$, it does so at the cost of potentially unnecessary penalization of RSVP flows, e.g., out-of-order delivery, even in the absence of congestion in AS2.

3.2. TOS Field Based Aggregation

A number of other approaches for aggregation have been brought forward, [CW97, BV97, Hei97, Kil97], with several [CW97, Hei97, Kil97] proposing the use of fewer bits in the IP header for classification purposes. In particular, it has been suggested to use the TOS octet field [Pos81] to specify different service classes as well as drop precedence. From the point of view of aggregation of RSVP flows, this means that RSVP data packets are assigned a value for the TOS field in their IP header, that is a function of both their service class, e.g., Controlled Load [Wro97a] or Guaranteed Service [SPG97], and whether the packet is conformant or not.

Specifically, several (the exact number is tbd and a function of the number of distinct service classes that are deemed necessary) TOS bits are used to specify service classes. Data packets from RSVP flows entering AS2 then have the TOS field in their IP header set accordingly to reflect the service class they have requested. As a result, routers in AS2 can classify packets using TOS bit patterns instead of full filters. Conceptually, on a link at a router in AS2, each service class is identified through its assigned TOS bit pattern and mapped onto a separate transmission queue, that has been allocated sufficient resources (bandwidth and buffers) to satisfy the QoS requirements of the aggregation of flows it carries.

In addition to the TOS bits identifying the service class to which the data packets of RSVP flows belong to, one more bit from the TOS field is needed to indicate conformance of packets from each flow to their corresponding TSpec. This explicit indication of non-conformant packets is key to enforcing flow isolation (requirement $R3$) and ensuring that the QoS guarantees of individual flows are met in spite of their aggregation into a common class. For example, as has been suggested, the conformance bit can be used by routers in AS2 to implement a ''drop precedence'' policy to preferentially discard non-conformant packets in case of congestion.

In general, ensuring that the aggregate resources allocated to each service class are adequate to satisfy the QoS guarantees of individual RSVP flows, i.e., requirements $R2$ and $R3$, requires coupling to the RSVP control path and this aspect is discussed in the next section. However, before addressing this issue, it should be noted that the above approach offers a number of benefits above those afforded by the previous tunneling solution. First, it avoids the overhead of encapsulation. Second, the ingress and egress processing required is minimal, i.e., update of the TOS field in the IP header (note that this could even be performed in the end-stations themselves). Third, it does not require any interactions with routing above and beyond what is normally required by RSVP. In other words, aggregation is supported in a manner that is essentially transparent to RSVP.

4. Control Path Aggregation

The aggregation of control information associated with individual RSVP flows is just as important for scalability as its data path counterpart. Specifically, maintaining PATH and RESV states for individual RSVP flows can represent a substantial burden in backbone routers which need to support a large number of flows. The goal of QoS aggregation is then to eliminate or at least minimize the amount of per flow control information that needs to be maintained. As with data path aggregation, this needs to be done while maintaining the QoS guarantees requested by individual flows. In particular, the resources allocated to a set of aggregated flows must reflect the ''sum'' of the reservation requests conveyed in individual RESV messages. Making sure this and the several requirements identified earlier (requirement $R2$, $R3$, and $R4$) are met, varies according to the data path aggregation method used. Note that aggregation also offers the opportunity for greater efficiency because of the potential benefits of statistical multiplexing. However, some care must be applied to avoid under-provisioning of resources in the backbone, e.g., aggregation may affect measurement based call admission rules in backbone routers.

4.1. Tunnel Based Aggregation

When QoS aggregation is achieved through the use of an RSVP tunnel, all RSVP control messages for individual flows are encapsulated and, therefore, not seen by any of the intermediate routers (in AS2). However, because those messages are carried across the tunnel, after decapsulation at the egress router, they will be forwarded as usual so that reservations for individual RSVP flows can still be established on the rest of the path, i.e., in AS1 and AS3. Note that because individual PATH messages are encapsulated, their ADSPEC is not updated as they cross the backbone. At the egress router of the tunnel, updating the ADSPEC in the PATH messages of individual flows is carried out using the corresponding ADSPEC fields from the PATH messages of the tunnel itself. Specifically, hop count, path latency, service specific quantities such as Guaranteed Service error terms, etc., are all updated as if the ADSPEC values for the tunnel were those of a single ‘‘node’’ (AS2 is considered as one node).

As far as the tunnel is concerned, its establishment is the responsibility of the ingress and egress routers at its end-points, which generate new RSVP control messages with their address as the source and destination addresses (1). In addition, the traffic specification (TSpec) and reservation levels (FLOWSPEC) specified in these messages need to adequately reflect the requirements of the flows aggregated into the tunnel. In particular, the type of service used for a tunnel should match that of the flows being aggregated on the tunnel, e.g., Controlled Load flows should be aggregated onto a Controlled Load tunnel.

----

1. Note that a possible alternative is to use a layer 2, e.g., ATM, tunnel, which would then be setup using the available layer 2 signalling.

4.1.1. Setting of Aggregate Reservations

In the case of a Controlled Load tunnel, the aggregate TSpec used in the PATH messages for the tunnel, needs to be selected so as to accommodate the TSpec's of all the flows it aggregates. A natural selection is to choose the sum of the TSpec's of all the individual flows being aggregated (see [Wro97b] for a discussion on how TSpec's are to be summed). Similarly, the TSpec specified in the FLOWSPEC of the RESV messages for the tunnel should again be chosen to ensure that the aggregated flows receive a level of service consistent with their individual requests. One option is to again select the sum of individual FLOWSPEC's, although as mentioned above the potential benefits of statistical multiplexing may allow a lower reservation level.

Irrespective of the aggregate reservation level specified, satisfying the QoS guarantees of individual flows is also predicated on the ''proper'' handling of excess traffic, i.e., packets from each flow that do not conform to their individual TSpec. Specifically, excess traffic MUST NOT be forwarded onto the RSVP tunnel, unless some form of explicit identification of excess traffic is provided. For example, this could be achieved through the use of a bit from the TOS field in the IP header of packets as suggested in Section 3.2.

The case of a Guaranteed Service Tunnel is somewhat more involved. There are two issues that need to be addressed. The first is the update of the ADSPEC in the PATH messages of individual flows at the egress router (see [SPG97] for details on the use of ADSPEC). The second is the selection of appropriate TSpec and RSpec for the tunnel, so that the delay bounds of all individual flows can be guaranteed. The handling of these two issues are not independent, and there are many possible solutions. In this document, we outline only one of several alternatives.

The update of the ADSPEC can be done as described before, using the ADSPEC values of the tunnel. The determination of appropriate TSpec and RSpec values for the tunnel, essentially follows the method described in [RG97]. Specifically, the TSpec used for the tunnel needs to be at least the ''sum'' of the TSpec's of the individual flows. Similarly, the reserved rate R of the RSpec is determined using eqs. (6) and (7) of [RG97], with the only difference that the individual delay bounds used in eq. (7) are only for the portion of the flows paths that coincide with the tunnel. This partial delay bound for individual flows is readily computed from the TSpec of individual flows, their RSpec, and the error terms for the portion of their path that corresponds to the tunnel.

It should be pointed out that as mentioned in [RG97], the resulting aggregate reservation rate for the tunnel can be either smaller or bigger than the sum of the individual reservation rates. Another point worth noting concerns the possible use of the slack term, in particular when individual flows specify a non-zero slack and a reservation rate R equal to their token rate r, i.e., they could tolerate a higher delay but cannot ask for a lower rate. In the case of a tunnel, the slack could be used to increase the individual delay bound for that flow used in eq. (7), provided that the SUM of the token rates of individual flows remains smaller than or equal to the aggregate reservation rate.

4.2. TOS Field Based Aggregation

The case of TOS based QoS aggregation is different from that of a tunnel because the egress point associated with a particular flow is not identified *a priori* at the ingress router. This has the advantage of eliminating the need for an ingress router to continuously interact with routing to monitor possible changes in egress routers and mapping of individual flows into tunnels. However, this means that some other mechanisms are needed to ensure that the appropriate amount of resources is reserved for RSVP flows between the associated ingress and egress routers.

There are many possible approaches that one can follow, and in this document we describe two, which we feel represent reasonable trade-offs between simplicity and minimization of backbone overhead. Other alternatives are clearly possible. In both approaches, as in the tunneling case, a key goal is to avoid or at least minimize awareness and/or processing of individual flows in the backbone. Satisfying this goal has several requirements and implications:

- Disable processing of (most) individual RSVP messages in the backbone, while still allowing their identification when they arrive at egress or ingress routers.

- Identify transparently, i.e., without relying on interactions with routing, the egress routers corresponding to individual flows entering the backbone at an ingress router.

- Reserve the appropriate amount of resources on backbone links to satisfy the requirements of individual flows routed over them.

- Properly update RSVP PATH messages of individual flows at egress routers.

In what follows, we describe two possible approaches to achieving those goals.

4.2.1. Ingress-Egress Aggregation: Method 1

Next, we describe a first approach, and the steps performed at ingress and egress routers to both identify each other and ensure proper aggregation of flows and allocation of resources between them.

- For new flows, the ingress router starts forwarding ‘‘individual’’ PATH messages carrying a Policy Object containing its IP address. Receipt of those individual PATH messages provides the associated egress routers with the identify of the ingress router for the flow. The individual PATH messages are initially processed by the backbone routers, and reach the egress router with updated ADSPEC information.

- Upon receiving an ‘‘individual’’ PATH message with a policy object specifying a new ingress router, the egress router logs the association between the flow and the ingress router and forwards the PATH message.

- Upon receipt of a RESV message (2) for a flow, the egress router forwards the ‘‘individual’’ RESV message with a Policy Object specifying its IP address. The Policy Object will eventually be delivered to the ingress router, and inform it of the identity of the egress router associated with the flow.

- Upon receipt of a RESV message identifying a new egress or when the ingress router deems there are sufficient flows to a given egress to consider aggregating them, it starts sending PATH message destined to this egress and representing the aggregation of all flows destined to it. At the same time, the ingress router starts sending the PATH messages corresponding to individual flows, in a format that ‘‘hides’’ them from backbone routers (more on this below) but not the egress router.

- Upon receipt of a PATH message destined to itself, the egress router sends a RESV message with an aggregate reservation for all the flows it has logged as coming from the associated ingress

----

2. Alternatively, the egress router could generate a ‘‘fake’’, e.g., near zero reservation, RESV
message immediately after receiving the first PATH message. This has the benefit of faster awareness about the egress at the ingress.

router. At the same time, it starts sending RESV messages for the individual flows directly to the ingress router. This ensures that they will not be processed by backbone routers, and any existing reservations for individual flows in the backbone will time out. Note that to lower the potential for call admission failure, the egress router may want to progressively increase the reservation level in its aggregate RESV message. This may give it a better chance of recapturing bandwidth as it is being released, when reservation states of individual flows time out.

- Upon receipt of ‘‘hidden’’ PATH messages for individual flows, the egress router changes them back to ‘‘standard’’ PATH messages and updates them with the ADSPEC information from the PATH message originated by the associated ingress router before forwarding them downstream.

- Upon receipt of RESV messages for individual flows from a known egress router, the ingress router simply forwards them upstream.

The above steps ensure that ingress and egress routers become aware of each other without having to directly query routing, and also ultimately removes awareness of individual flows in backbone routers. However, it is still necessary to describe how route changes within the backbone are handled. This is tightly coupled to the approach used to ‘‘hide’’ RSVP PATH messages in the backbone, and we therefore describe this next.

In the case of tunnels, individual RSVP messages were ‘‘hidden’’ on backbone links because they were encapsulated within another IP header. As a result backbone routers would forward them as regular IP packets. Furthermore, because the destination address in the encapsulating IP header was that of the egress (ingress) router, decapsulation would be performed and ensure proper identification and processing of the RSVP messages. Such a solution is not applicable in the case of TOS based aggregation, because of the decoupling from routing, i.e., identity of egress or ingress, if known, cannot be used to ensure delivery of RSVP messages

There are several possible options to overcome those problems while avoiding processing of RSVP messages from individual flows in the backbone. Processing of RSVP (PATH) messages from individual flows in the backbone can be avoided simply by hiding the information used to trigger RSVP processing, i.e., turn the router-alert option [Kat97, KAPJ97] off at the ingress router. The problem is then that without the router-alert option on, the egress router will also fail to identify, and therefore intercept and process those PATH messages.

There are several possible solutions to this problem. One is to use some other bit pattern in the IP header, that can be used by egress routers to identify RSVP PATH messages from individual flows. For example, a TOS bit combination could be assigned to indicate ''aggregated control information.'' Routers responsible for de-aggregating control information, e.g., egress routers, would then intercept such packets, while other routers (backbone routers) would ignore them. Another option is to require that egress routers examine the protocol number of all arriving packets, even when the router alert option is not set. This may, however, impose a significant performance penalty. A third option is to keep the router alert option set, but use a different protocol number inside the backbone. Backbone routers would still intercept RSVP PATH messages from individual flows, but not need to process them any further, i.e., upon identifying the new protocol number they would simply forward the packet on. A last option is to define a *new* router alert option for ''Unaggregated RSVP'' messages, which would be silently ignored by backbone routers, but recognized by access (ingress/egress) routers.

This last alternative (see Appendix A for additional details) appears to provide a reasonable trade-off, that ensures the required functionality at egress routers while keeping the backbone overhead reasonable.

Assuming that one of the above mechanisms is being used, PATH messages for individual flows are now being automatically delivered directly from ingress routers to the appropriate egress routers. However, note that PATH messages are not being processed at any of the backbone routers they traverse. The main implication for the egress is that the ADSPEC field of the PATH messages has not been updated to reflect the characteristic of the backbone path they have traversed. As a result, they cannot be readily propagated forward by the egress router, unless the information needed to properly update their ADSPEC is *already* available at the egress router. This is one of the motivations for the above choice of initially sending individual PATH messages into the backbone, as this enables the egress to first acquire the necessary information to update the ADSPEC of ''hidden'' PATH messages. However, this approach does not address the problem in case of route changes in the backbone.

Route changes in the backbone result in ''hidden'' PATH messages being delivered to a *new* new egress, without being preceded by corresponding ''clear'' PATH messages. As a result, the new egress does not have the necessary information to update the ADSPEC of the ''hidden'' PATH messages it starts receiving. Hence, those messages cannot be propagated forward. In order to address this problem, the ingress router needs to become aware of the route change. The simplest approach is to rely on RSVP soft states. Basically, the ingress router will detect that it stops receiving RESV messages from the old egress routers (at least for the flows affected by the route change). It can then use that information as a trigger to start forwarding the PATH messages of those flows again as *regular* RSVP PATH messages. As a result, they will be processed by intermediate backbone routers, and we are back to the initial case described above.

4.2.2. Ingress-Egress Aggregation: Method 2

In this section, we describe a second alternative, which is mostly a variation on the general method described in the previous section. The main motivation for this variation is to avoid *all* processing of individual RSVP flows in the backbone. This is desirable as even the limited processing of individual RSVP flows required from backbone routers by method 1, can represent a substantial processing load when flows are of short duration. In addition, this second method can avoid reliance on Policy Objects.

The main difference with the previous method is that the PATH messages from individual flows are not sent directly in the backbone. Instead, they are always forwarded as ‘‘hidden’’. The main issue is then to determine how to inform the ingress router of the identity of the egress router associated with each individual flow, without relying on explicit queries to routing. We describe next, the different steps involved in addressing this issue.

- Upon receipt of a new PATH message, the ingress router forwards it as ‘‘hidden’’ into the backbone.

- On receipt of a hidden PATH message for a new flow, the egress router immediately notifies the ingress router of its existence (the identity of the ingress is carried in the PHOP of the PATH message). This notification can take several forms. One possibility is for the egress router to generate a PATH_ERR message (with some appropriate new error code) directly destined to the ingress router. Another possibility is for the egress router to generate a ‘‘fake’’ RESV message with near-zero reservation (FLOWSPEC). Note that as discussed earlier, ‘‘hidden’’ PATH messages cannot be forwarded until the information needed to update their ADSPEC is available (more on this below).

- On receipt of a ‘‘fake’’ RESV or a PATH_ERR from a new egress, the ingress proceeds to send a ‘‘regular’’ aggregate PATH message to that egress.

- On receipt of an aggregate PATH message (destined to itself), the egress now has the information necessary to update the ADSPEC of the individual PATH message and can start forwarding it. The main disadvantage here is the latency incurred in forwarding the individual PATH message. However, this latency is typically only incurred by the first flow from a given ingress. The egress can use the ADSPEC from the aggregate PATH message to update and immediately forward the PATH messages of subsequent flows from that ingress.

- On receipt of a new RESV message for an individual flow, the egress sends a RESV message associated with the aggregate PATH from the corresponding ingress (or updates an existing RESV). The individual RESV messages are then forwarded directly to the ingress router.

As mentioned earlier, the method embodied in the above steps avoids any processing of individual flows in the backbone. The cost is an increased latency in propagating the first PATH message of the first flow from the associated ingress.

4.2.3. Setting of Aggregate Reservations

Proper selection of appropriate aggregate reservation levels requires some care, especially for Guaranteed Service flows. For Controlled Load flows, it is only necessary that in backbone routers the queue assigned to Controlled Load traffic, be allocated the proper service rate. Since rate is an additive quantity, aggregate reservations can be based on the sum of the FLOWSPECs of individual flows. The situation is again more complex for Guaranteed Service flows.

The main difference with the tunnel-based case, is that on any link in the backbone the overall aggregation of packets/flows with the same TOS value (corresponding to the Guaranteed Service) is not know to either the ingress or egress routers associated with individual RSVP flows whose route through the backbone includes that link. As a result, the egress router cannot use the approach of Section 4.1.1 to determine an appropriate aggregate service rate, that will ensure that all individual delay bounds are met.

In order to support aggregated Guaranteed Service flows in this setting, it is necessary to change the ‘‘node model’’ used to represent the backbone. Specifically, an approach similar to the one used in the ISSLL drafts to account for ATM networks, can be used. It amounts to representing the backbone as a delay only node. In other words, the backbone only contributes to the D error term of the ADSPEC and not the C term. The main difference with an

ATM network is that, contrary to ATM switches, individual backbone routers will update the ADSPEC in PATH messages. In order to ensure a behavior consistent with that of a delay-only node, each individual router needs to only update the D error term of the ADSPEC of PATH messages it processes. The implication of this behavior is that the scheduling and call admission support for Guaranteed Service flows in backbone routers, will be based on ensuring a fixed delay upper bound for the TOS queue assigned to Guaranteed Service packets. This delay upper bound will then be the quantity used to update the D error term in the ADSPEC field of PATH messages.

5. Conclusion and Recommendations

In this draft we have outlined issues and proposed possible approaches to allow aggregation of individual RSVP flows, without precluding support for individual reservations where available. This can enable delivery of the end-to-end and per flow QoS guarantees supported by RSVP and the Int-Serv Services, while avoiding possible scalability limitations.

As a result of this exercise, several requirements emerged to support the different aggregation methods that were discussed. These requirements are summarized below':

- Allocation of one bit from the TOS field of the IP header to specify in-profile and out-of-profile packets.

- Allocation of one bit pattern from the TOS field that can be mapped to the Controlled Load service, and at least one bit pattern from the TOS field that can be mapped to the Guaranteed Service (two would be preferable to provide some granularity in the delay bounds for Guaranteed Service flows).

- Support for a mechanism to selectively ‘‘hide’’ RSVP control messages. Specifically, the preferred mechanism is through the introduction of an new Router Alert option, that can be selectively recognized or ignored in routers.

References

[Boy97] J. Boyle. RSVP extensions for CIDR aggregated data flows, (draft-rsvp-cidr-ext-00.txt). Internet draft (work in progress), Internet Engineering Task Force, February 1997.

[Bra97] S. Bradner. Internet protocol quality of service problem statement, (draft-bradner-qos-problem-00.txt). Internet draft (work in progress), Internet Engineering Task Force, September 1997.

[BV97] S. Berson and S. Vincent. A ‘‘classy’’ approach to aggregation for integrated services, (draft-berson-classy-approach-00.txt). Internet draft (work in progress), Internet Engineering Task Force, March 1997.

[BZB+97] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource reSerVation Protocol (RSVP) version 1, functional specification. Request for comments, rfc 2205, (proposed standard), Internet Engineering Task Force, September 1997.

[CW97] D. Clark and J. Wroclawski. An approach to service allocation in the Internet, (draft-clark-diff-svc-alloc-00.txt). Internet draft (work in progress), Internet Engineering Task Force, July 1997.

[Hei97] J. Heinanen. Use of the IPv4 TOS octet to support differential services, (draft-heinanen-diff-tos-octet-00.txt). Internet draft (work in progress), Internet Engineering Task Force, October 1997.

[KAPJ97] D. Katz, R. Atkinson, C. Partridge, and A. Jackson. IP router alert option, (draft-ietf-ipngwg-ipv6-router-alert-03.txt). Internet draft (work in progress), Internet Engineering Task Force, July 1997.

[Kat97] D. Katz. IP router alert option. Request for comments, rfc 2113, (proposed standard), Internet Engineering Task Force, February 1997.

[Kil97] K. Kilkki. Simple integrated media access (SIMA). (draft-kalevi-simple-media-access-01.txt). Internet draft (work in progress), Internet Engineering Task Force, June 1997.

[Pos81] J. Postel. Internet protocol. Request for comments, rfc 791, (standard), Internet Engineering Task Force, September 1981.

[RG97] S. Rampal and R. Guérin. Flow grouping for reducing reservation requirements for Guaranteed Delay service, (draft-rampal-flow-delay-service-01.txt). Internet draft (work in progress), Internet Engineering Task Force, July 1997.

[SPG97] S. Shenker, C. Partridge, and R. Guérin. Specification of guaranteed quality of service. Request for comments, rfc 2212, (proposed standard), Internet Engineering Task Force, September 1997.

[Wro97a] J. Wroclawski. Specification of the controlled-load network element service. Request for comments, rfc 2211, (proposed standard), Internet Engineering Task Force, September 1997.

[Wro97b] J. Wroclawski. The use of RSVP with IETF integrated services. Request for comments, rfc 2210, (proposed standard), Internet Engineering Task Force, September 1997.

A. Router Alert Options for Concealing ‘‘Individual’’ PATH Messages

As discussed in Section 4.2, the scalability of RSVP is improved when using TOS field based aggregation if the PATH messages from individual applications are concealed from the interior routers in the backbone. PATH messages are addressed either to a destination host or multicast group and are transmitted with the IP router alert option as defined in [Kat97] or [KAPJ97]. This allows routers along their transit path to intercept the packets for RSVP processing. To prevent the backbone routers from intercepting and processing the PATH messages from individual applications, while allowing the aggregating egress routers to recognize and intercept them, a new router alert option value may be used.

The syntax of the IPv4 router alert option is defined as follows [Kat97]:

A.1. IPv4 Syntax

The Router Alert option has the following format:

```
+--------+--------+--------+--------+
|10010100|00000100|  2 octet value  |
+--------+--------+--------+--------+
```

Type:
  Copied flag: 1 (all fragments must carry the option)
  Option class: 0 (control)
  Option number: 20 (decimal)

Length: 4

Value: A two octet code with the following values:
0 - Router shall examine packet
1-65535 - Reserved

The specification states that ''Unrecognized value fields shall be silently ignored''.

The syntax of the IPv6 router alert option is defined as follows [KAPJ97]:

A.2. IPv6 Syntax

The router alert option has the following format:

```
+--------+--------+--------+--------+
|00| TBD | Len= 2 | Value (2 octets)|
+--------+--------+--------+--------+
```

''TBD'' is the Hop-by-Hop Option Type number (To be allocated by the IANA).

Nodes not recognizing this option type SHOULD skip over this option and continue processing the header. This option MUST NOT change en route. There MUST only be one option of this type, regardless of value, per Hop-by-Hop header.

Value: A 2 octet code in network byte order with the following values:

0	Datagram contains ICMPv6 Group Membership message.
1	Datagram contains RSVP message.
2	Datagram contains an Active Networks message \cite{ANEP97}.
3-65535	Reserved to IANA for future use.

New value fields must be registered with the IANA.

This specification states that ''Unrecognized value fields MUST be silently ignored and the processing of the header continued''.

There are two alternatives which will satisfy the requirement to ''hide'' application PATH messages (when necessary) from the backbone routers:

- Define a 2 octet router alert option value for both IPv4 and IPv6 which signifies that the datagram contains an ''Unaggregated RSVP Message''. The router should silently ignore this router alert option and continue to forward the packet unless specifically configured to recognize and intercept it.

- Define a 2 octet router alert option value for both IPv4 and IPv6 which signifies that the router should ''Ignore by Default''. The router should silently ignore this router alert option and continue to forward the packet unless specifically configured to recognize and intercept it.

PATH messages from individual applications would be transmitted by the aggregating ingress router using either router alert option value (whichever is defined) whenever it employs TOS field based aggregation to a particular egress router. Aggregated PATH messages to that router would be transmitted with the default router alert option value used for RSVP. The backbone routers would be configured to ignore router alert options using this new option value. The aggregating egress routers would be configured to intercept packets transmitted with the new router alert option value.

Authors' Address

Roch Guerin
IBM T.J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598
Phone: +1 914 784-7038
Fax: +1 914 784-6205
Email: guerin@watson.ibm.com

Steven Blake
E95/664
IBM Corporation
800 Park Offices Drive
Research Triangle Park, NC 27709
Phone: +1-919-254-2030
Fax: +1-919-254-5483
Email: slblake@raleigh.ibm.com

Shai Herzog
IPHighway
Email: herzog@iphighway.com