A case for dynamic sender-based reservations in the Internet

Paul Patrick White
Department of Computer Science
University College London
Gower Street
London WC1E 6BT
England

email: p.white.cs.ucl.ac.uk
phone: +44 171 419 3701

Jon Crowcroft
Department of Computer Science
University College London
Gower Street
London WC1E 6BT
England

email: j.crowcroft.cs.ucl.ac.uk
phone: +44 171 380 7296

Abstract

In this paper we discuss the need for resource reservation in the Internet and examine some of the strengths and weaknesses of RSVP, which is currently the most popular of Internet reservation protocols that have been developed. We also discuss some alternative reservation protocols for packet networks, in particular the ATM Block Transfer(ABT) reservation protocol that has been designed for use in Asynchronous Transfer Mode(ATM) networks and which uses 'in-line' control packets to modify reservations 'on the fly' to achieve very efficient bandwidth utilisation. Finally we present a proposal for a new reservation protocol, known as DRP(Dynamic Reservation Protocol) which combines many of the strengths of RSVP and ABT with few of the weaknesses to achieve a highly bandwidth-efficient reservation mechanism with excellent scalability with regards to round trip time, data rate and number of hosts.

This paper was published in The Journal of High Speed Networks in late 1998.

Contents

1 Introduction

It is clear that the current Internet which was founded upon the concept of ‘best-effort’ datagram delivery must be enhanced in some way in order to accommodate the changing communications environment. In particular there is a growing demand for real-time applications which have specific Quality of Service(QoS) requirements, especially with regard to end-to-end delay and minimum bandwidth, both of which cannot be guaranteed in the current Internet using traditional connectionless best-effort delivery. Furthermore as the World-Wide-Web is increasingly used for business there is a growing number of users for whom delay bounded access of information is important.

In response to the changing requirements of Internet users, much attention has focussed on the use of resource reservation as a means of providing selected data flows with special QoS commitments in accordance with their needs. Under such a framework it is likely that special QoS delivery would be the exception rather than the rule with the majority of Internet traffic continuing to receive the ‘default’ best-effort mode of delivery. The special QoS required by a specific data flow can be realised by reserving resources(bandwidth, buffer space) and installing appropriate scheduling behaviour in each router along the end-to-end path followed by the data flow. Such mechanisms require admission control at the individual intermediate nodes to ensure that the request for reservation is only accepted and installed provided sufficient resources are available. In addition, per-flow state[1] in the intermediate nodes will usually be required in order to identify the flows to receive special QoS as well as the QoS to be received.

In order to allow users to invoke special QoS delivery on demand for a data flow several protocols have been developed to enable users to communicate their QoS needs to the intermediate routers along the data path in an IP internetwork. The majority of these protocols initiate the set up of flow-specific reservation state in intermediate routers, a notable exception being the approach described in [1] whereby no per-flow reservation state is set up in routers which instead record their reservation commitments as a whole per output port. While this approach potentially offers very good scalability characteristics for a large number of flows, it is dependent upon a certain degree of trust among end hosts not to exceed their indicated traffic levels unless per-flow policing is applied at the network access point. In addition, the approach is only able to offer end applications an approximate minimum bandwidth without any quantitative guarantees on loss or delay and so may not be suitable for applications with stringent QoS requirements such as Distributed Interactive Simulation[15].

Of the reservation protocols that set up flow-specific reservation state, an early example in the Internet is the Stream Protocol, ST[7] which was limited to unicast reservations. Although its successors, ST-II[16] and the more recent ST2+[6] can handle both multicast and unicast reservations as well as possessing many improvements over ST, the ST group of protocols has attracted little commercial interest. By contrast, another reservation protocol, RSVP[4] has received significant industry support and with good reason. Unlike the ST protocols, RSVP reservation state is soft-state and will time-out in the absence of any refresh reservation requests within a certain time period. This so-called soft-state nature of RSVP provides a very simple failure recovery mechanism over a wide range of fault scenarios and helps to retain much of the robustness that has helped to make IP so successful. The soft-state approach where the end applications are responsible for maintaining the flow-specific router state leads to a

[1] The introduction of per-flow state is a significant departure from the initial Internet design philosophy of a pure connectionless network with no per-flow state in the intermediate routers.

significant reduction in complexity compared to a hard-state approach where the network is responsible for maintaining the flow-specific router-state. RSVP has other notable architectural differences compared to the ST protocols such as receiver-initiated rather than sender-initiated reservations[2]. The initial design of RSVP was to a large extent influenced by the needs of multicast conferencing applications although its intended use is now much broader.

While RSVP is concerned merely with signalling the end application’s reservation requests to the intermediate nodes, it is the special QoS delivery models that define the node behaviour required to meet the signalled special QoS objectives. The Integrated Services Working Group(intserv) of the IETF[8] has standardised two special QoS delivery models, both of which offer applications an end-to-end minimum bandwidth albeit with different assurances. First, Guaranteed Service which offers applications a loss-free service with an end-to-end delay bound. Second, Controlled-Load Service which does not provide any quantitative guarantees on delay or loss, although qualitatively these parameters can be expected to be the same as for best-effort delivery under low network load. The network layer QoS achieved via these special QoS delivery models must be mapped onto specific link layer technologies such as ATM, IEEE 802 and Ethernet. Responsibility for these mappings has been assigned to the Integrated Services over Specific Lower Layers(issl) Working Group of the IETF[9].

In parallel with the recent Internet growth much interest has been generated by Asynchronous Transfer Mode(ATM), a technology designed from the outset with end-to-end QoS in mind. A necessary component in ATM networks for achieving QoS on demand is a signalling protocol in order to request resource reservations in the intermediate nodes of the end-to-end path. More traditional ATM signalling protcols such as ITU’s Q.2931 standard for public networks[10] or ATM Forum’s UNI standards for private networks[2] use end-to-end handshaking to set up an end-to-end reservation before data transfer can take place. A more dynamic and flexible approach is that provided by the ATM Block Transfer/Immediate Transfer mode(ABT/IT) signalling protocol which sends reservations in-line with data and as such is more conducive to efficient bandwidth utilisation than the more static end-to-end handshaking approach.

In the next few sections we review the RSVP and ABT/IT protocols as well as identifying some of their limitations. We then show in sections 6 and 7 how these limitations can be minimised or eliminated by combining many of the features of these two protocols to produce a new QoS signalling protocol known as DRP which can be used to setup the IETF’s integrated services models ‘on-the-fly’. Following this we present details of packet formats and processing rules for DRP before presenting our conclusions.

[2] ST-II+ permits both sender and receiver-initiated reservations, ST-II and ST permit sender-initiated reservations only.

2 Overview of RSVP

The primary messages in RSVP are the Path message and the Resv message. The Path message travels downstream from sender(s) to receiver(s) of the communications session[3] and is processed at each RSVP-capable node that it encounters. The Path message serves 3 main functions. First, it installs the address of the previous hop(s) at each node in order to facilitate correct reverse routing of ResV messages in the upstream direction for source-based trees[4]. Second, a Path message contains the sender traffic specification(sender Tspec) which is required by receivers when deciding the reservation to ask for, and by intermediate routers to clip the requested reservation Tspec in certain circumstances[5]. Third, a Path message may optionally include a block of path-specific information which is updated at each hop in order to present the receivers with certain end-to-end path characteristics which may be used along with the sender traffic characteristics to calculate the reservation required in order to achieve a specified end-to-end QoS required by end applications. A receiver invokes special QoS delivery for a dataflow by sending a Resv message to the previous hop(s) in the delivery tree. The reservation message indicates the level of resources to be reserved in each router on its journey towards the sender(s). Also each reservation has an associated style which can be either fixed filter, in which case the reservation applies to a single specific sender, or a shared reservation style in which case the reservation is shared among multiple senders usually on the understanding that only one of them will be transmitting at once. RSVP uses so-called 'soft-state' whereby the reservations timeout in the absence of refresh Resv messages from end systems within a certain timeout period. In the steady-state[6], a Resv message received by a node is not propagated upstream immediately. Instead it is held until the end of some refresh period epoch whereupon it is merged[7] with Resv messages that were received from other downstream nodes during the last refresh period epoch to create a single refresh Resv message to be propagated upstream. This merging mechanism ensures that the number of refresh Resv messages received by the sender in the steady state is determined by the number of its next hops rather than the number of receivers. Consequently the merging mechanism is essential to allow RSVP to scale to a large number of receivers.

[3] The communications session is defined by the 3-tuple (IP destination address, transport layer protocol, transport layer destination port).

[4] The IP destination address in a Resv message is set to the previous hop address at each hop. This ensures that the set of routers that install a reservation in the case of a successful reservation attempt is identical to the set of routers traversed by the data packets captured by the reservation request. Moreover this will be true even for asymmetrical routes.

[5] The actual Tspec used by admission control in intermediate routers is given by MIN(Tspec in reservation request, sum of sender Tspecs captured by request). The sender Tspecs are obtained from Path messages. This procedure safeguards against a receiver inadvertently overspecifying the Tspec in a reservation request as well as minimising over-reservations that may occur with shared reservation styles as described in section 3.3.

[6] The steady state means that the reservations for all receivers are in place and the Resv messages are simply refreshing existing reservation state rather than altering its value or setting up new reservation state. In the case of reception at a node of a Resv message that causes an alteration to existing reservation state and that alteration is such that it causes a change in the merged Resv message to be sent upstream, the new merged Resv message will be propagated upstream immediately.

[7] Merging is performed according to the rules specified in the RSVP specification[4].

3 Limitations of RSVP

In this section we identify what we believe to be significant weaknesses of RSVP.

3.1 Limited QoS dynamics

Currently RSVP only allows receiver-initiated reservations. Although receivers are permitted to change their QoS requests within the lifetime of the data flow the end-to-end signalling nature of RSVP prohibits renegotiation over a timescale less than the round-trip time of the connection path. In circumstances where the characteristics of the traffic stream can change over a short time-scale these limited QoS dynamics can preclude RSVP from achieving optimal reservations. One example is the case of a shared-reservation in an audio session for senders with different Tspecs. Here RSVP would be unable to alter the reservation quick enough to reflect the current speaker and so would have to use a reservation that satisfied the most demanding requirements of all speakers and did not change to reflect any change in the speaker. This scenario is covered in more detail in section 3.3. Another example is a multimedia teaching seminar multicast in real-time. This seminar might combine audio, video and data in real-time while allowing the student end-hosts to communicate with the lecturer end-host during the course of the seminar. Each media component could have unpredictable active and quiet periods and the encoding and bit rate of each component might change between active periods. In such a scenario, RSVP would be unable to respond fast enough to accurately track each component's characteristics and QoS requirements on a per-activity burst basis. Consequently, in order to avoid high risk of user-perceived QoS disruption, RSVP would need to err on the side of caution meaning that for a large portion of the time the reservation for each component was over-specified.

3.2 Path message overhead in large-scale multipoint-multipoint applications

In RSVP in the steady state each sender to a session sends Path messages at periodic intervals in order to install/refresh Path state in intermediate routers and receiver end-nodes. In the basic RSVP specification, no merging of Path messages from different senders occurs as they progress through the distribution tree in which case the overhead[8] associated with Path messages on each link of a multicast shared-tree[9] in the steady state will grow in proportion to the number of senders as shown in Figure 1a. If RSVP aggregation is used as shown in Fig 1b, then multiple Path messages within the core can be aggregated into a single Path message. Regardless of whether core aggregation is used, the number of Path messages received in the steady state per refresh period by each on-tree router and host outside the core will be equal to the number of senders. In addition each on-tree router and host outside the core must maintain a separate Path state entry per sender. These burdens will become significant in very large-scale multipoint-multipoint applications of the future such as interactive gaming where the number of senders could run into hundreds or even thousands.

[8] The overhead associated with Path messages is two-fold. First, bandwidth consumption. Second, and perhaps of greater concern, processing load at routers and end-nodes.

[9] A shared-tree builds a single multicast tree per multicast group. Each host connected to the tree can use it to both send and receive. Source-based multicast routing builds a separate multicast tree for each sender to a multicast group. A source-based tree has a single sender host and multiple receiver hosts.

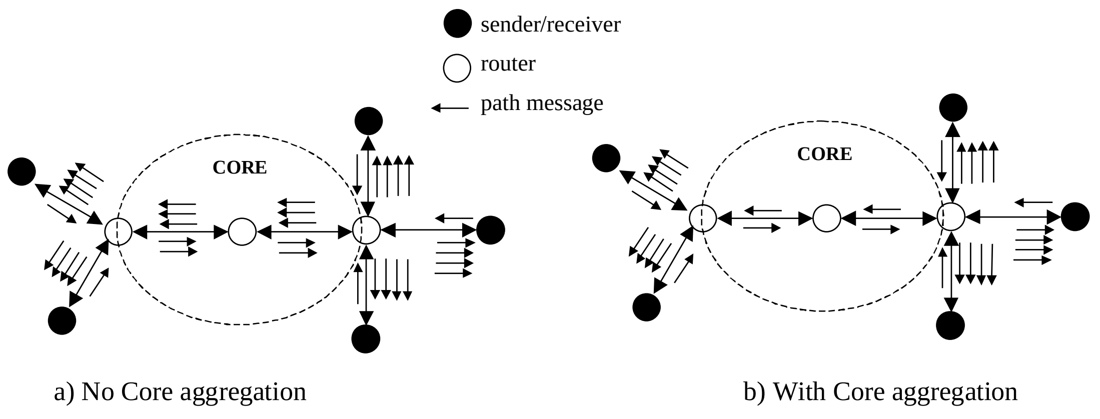


Figure 1: Number of RSVP Path messages per refresh period in the steady state on a shared multicast tree with and without core-aggregation.

3.3 Suboptimality of shared reservations

RSVP supports shared reservation styles which would typically be used in scenarios where only one source is likely to be active at once. In such cases, a shared reservation can achieve lower overall reserved bandwidth than a number of sender-specific reservations. However as we will illustrate, the bandwidth reserved can still be suboptimal, that is the reserved bandwidth might be permanently or intermittently greater than is actually required to provide the QoS guarantee. With regard to the controlled-load service, which is described entirely by a reservation Tspec, we define two different types of suboptimality as follows

Static 'Tspec suboptimality' - reserved Tspec > maximum Tspec of all sources captured by the reservation.

dynamic 'Tspec suboptimality' - reserved Tspec > Tspec of at least one of the sources captured by the reservation.

The Tspec contained in the Resv message arriving at an interface may exhibit static suboptimality for 2 reasons. Firstly, one or more receivers may have requested more than the maximum Tspec of all sources contained in the request's filter spec, in order to allow for a certain amount of overspeaking. Secondly, even if none of the receivers allow for overspeaking in their requests, static Tspec suboptimality can be introduced when a reservation request splits at a source branch point on its way upstream. We will illustrate this latter case shortly. Now if RSVP made no allowance for reservation requests that reflected some degree of overspeaking, then the routers could completely eliminate static suboptimality by limiting the installed Tspec to the maximum Tspec of all senders captured by the reservation in accordance with equation (1) where the sender Tspecs are obtained from Path state installed in the router.

$$Tspec = MIN(MAX\{\text{Sender Tspec}\}, \text{Tspec of reservation request}) \qquad (1)$$

However, because RSVP does allow for requests that reflect an amount of overspeaking, equation (2) is used instead and static suboptimality of the reserved Tspec may sometimes be evident.

$$Tspec = MIN(\sum \text{sender Tspec}, \text{Tspec of reservation request}) \qquad (2)$$

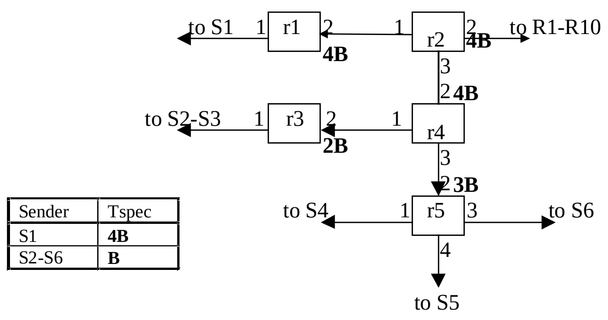


Figure 2:Shared-explicit, Controlled-Load Service reservations at RSVP-capable routers.

We now present an example in Figure 2 to illustrate static and dynamic suboptimality of reserved Tspecs. R1-R10 represent receivers of a multicast session that have requested shared-explicit[10] style reservations such that the union of the associated filter specs of these reservations gives the set of senders S1-S6. Each request generated by R1-R10 is for Controlled-Load Service with a Tspec equal to the maximum Tspec of all senders contained in the reservation's filterspec, that is none of the requests allow for overspeaking. In addition suppose senders S2-S6 each have Tspecs equal to some base quantity B but sender S1 has a Tspec of 4B. Such an example of different sender Tspecs might occur in an audio-conference if the senders used different coding schemes for their audio signals. For our example, the Tspec of the installed reservation at r2 interface 2 will be 4B. This is an example of dynamic reservation suboptimality and when any other source apart from S1 is transmitting, this reservation is 4 times larger than necessary assuming only one source is active at once. At r4 interface 2 the sum of all path state entries will be equal to 5B(S2-S6) and the installed reservation will be equal to 4B in accordance with equation (2). This is an example of static suboptimality since the reserved Tspec of 4B will always be 4 times greater than the Tspec of the active source assuming only one source is transmitting at once. At r3 interface 2 the sum of all Path state entries will be equal to 2B and so will the installed reservation Tspec. Similarly at r5 interface 2 the sum of Path state entries will be equal to 3B and so will the installed reservation Tspec. Assuming only one source is transmitting at once, the reservation Tspecs at r3 interface 2 and r5 interface 2 will always be 2 and 3 times greater than necessary respectively and are therefore statically suboptimal.

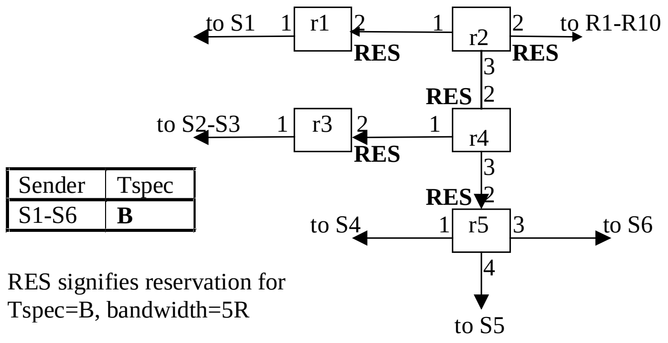


Figure 3:Shared-explicit, Guaranteed Service reservations at RSVP-capable routers

[10] RSVP permits two kinds of shared reservations. First, shared-explicit style where the filter spec (set of senders to which the reservation applies) refers to an explicit list of senders, and second, wildcard filter style where the filter spec refers to all senders.

The potential for suboptimality of shared reservations is even greater in the case of guaranteed service where over-reservations can exist even when the Tspecs of the senders are the same and no overspeaking has been allowed for in receivers' reservation requests. This is because Guaranteed Service reservation requests contain a bandwidth to be reserved, Rspec in addition to a Tspec. The minimum required Rspec for each (sender-receiver) logical path that is to receive a reservation is a function of the sender Tspec, the desired end-to-end delay bound and the path characteristics between the sender and receiver. These last two parameters can vary among (sender, receiver) pairs and consequently so can the required Rspecs even when the Tspecs of the senders are the same. As we shall soon illustrate, this leads to further suboptimality, 'Rspec suboptimality', in addition to Tspec suboptimality which can also occur in guaranteed service shared sessions. We define 'Rspec suboptimality' as follows:

Static Rspec suboptimality – the installed Rspec is greater than the minimum required to satisfy the end-to-end delay bounds for all of the senders.

Dynamic Rspec suboptimality – the installed Rspec is greater than the minimum required to satisy the end-to-end delay bound of at least one of the senders.

While RSVP provides a mechanism, equation (2), for reducing static Tspec suboptimality, no such mechanism exists to reduce static Rspec suboptimality. This situation is illustrated in the example of Figure 3 where the sender Tspecs are all equal to some base quantity B. In this example let us assume that each receiver makes a shared reservation request without allowing for any overspeaking. Let us further assume that no receiver requires an Rspec greater than bandwidth R in order to achieve the desired delay bound for packets from S1. Now suppose that the maximum Rspec required by any receiver is 5R and this just happens to be determined by a particular receiver's desired end-to-end delay bound for S4. In this scenario the Rspec of the installed reservation at r2 interface 2 will be equal to 5R. This is an example of dynamic Rspec suboptimality since, for example, when source S1 alone is transmitting, the reserved Rspec is 5 times greater than necessary to meet the end-to-end delay bounds of all receivers. Moreover as this reservation propagates to each of the senders S1-S6 the Rspec will remain unchanged at 5R which will introduce static Rspec suboptimality. For example the installed Rspec at r1 interface 2 will be 5R even though only R is required to satisfy the end-to-end delay bound for all upstream senders which in this case is simply S1.

Theoretically it is possible to modify RSVP to prevent the static Rspec suboptimality. This could be achieved if each receiver's shared-reservation request was to include a list of the contributory (sender, bandwidth) pairs that the shared reservation Rspec was obtained from. These (sender,bandwidth) pairs could then be taken into account during the merging process at intermediate nodes. For example suppose a receiver calculates that its flowspec to S1, S2 and S3 should be (RSpec1, Tspec1) , (Rspec2, Tspec2) and (Rspec3, Tspec3) respectively in order to satisfy the desired end-to-end delay bounds for each of these senders. Further suppose that Rspec1>Rspec2>Rspec3 and Tspec1>Tspec2>Tspec3 and that the receiver wishes to make a shared reservation without allowing for any overspeaking. The generated reservation request would then have filterspec=(S1, S2, S3) and flowspec=(Rspec1, Tspec1). The receiver's list of contributory (sender, bandwidth) pairs would then be (S1, Rspec1: S2, Rspec2 : S3, Rspec3). The reservation installed at a router interface would be obtained by merging of incoming guaranteed service reservation requests in the usual manner. In addition, for each adjacent node sending reservation requests to that interface a separate (sender, bandwidth) list state entry would be installed at the interface. The main components of the reservation request to be propagated upstream out of interface, *i* would then be obtained as follows:

- The filterspec, *fspecout* and Tspec, *Tspecout* to be included in the reservation request would be obtained in the usual RSVP manner for shared-explicit reservations.
- For each sender in *fspecout*, a router would include an entry in the (sender, bandwidth) list,

listout in the reservation request to be sent out of interface *i*. The bandwidth value for a particular sender in *listout* would be given by the maximum of bandwidth values for all (sender, bandwidth) entries that the router has installed for that sender.

- Rspec, Rspecout to be included in the reservation request would be obtained by taking the maximum of all bandwidth values contained in *fspecout*.

Although, this modification to RSVP would prevent the static Rspec suboptimality problem in the case of shared-explicit reservations it would increase the size of the reservation messages by at least 4 bytes per sender contained in the filterspec of the reservation message. In addition, routers would need to store extra state and their processing demands would increase.

3.4 Synchronization of reservation styles

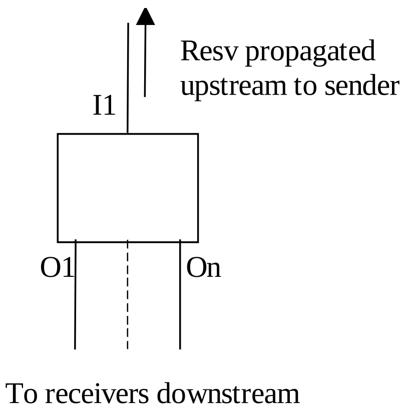
RSVP requires that all receivers of a session use the same reservation style when making a reservation request. Currently, RSVP supports 3 styles, one of which is sender-specific while the other two are shared reservation styles. The sender-specific style is known as fixed-filter and its filterspec contains a single sender. One of the shared reservation styles is known as shared-explicit style and its filterspec contains an explicit list of senders. The other shared reservation style is known as wildcard filter style since its filter spec is wildcard, i.e. matches on any sender to the session. Merging between these reservation styles is prohibited since it might affect the service experienced by some of the receivers. For example a fixed-filter reservation style is made by a receiver that wishes a dedicated bandwidth to be allocated to the packets it receives from a specific sender. If this fixed-filter reservation was merged with a shared–explicit reservation then this dedicated bandwidth could no longer be guaranteed. Because of such problems, RSVP disallows merging between different reservation styles. In addition, RSVP disallows co-existence of different reservation styles at the same interface as this can lead to reservation suboptimalities. For example suppose there are 10 senders, S1-S10 to a multicast session and the sender Tspec of each of these senders is equal to some base quantity, B apart from S1 whose Tspec is equal to 5B. Suppose receiver A makes a wildcard filter reservation with Tspec equal to 5B, while receiver B makes a shared-explicit reservation with Tspec equal to 5B and filterspec equal to S1-S5. If both of these reservations are installed at the same interface then in effect the shared-explicit reservation is redundant since its reservation Tspec is the same as that of the wildcard filter reservation while its filtersspec is a subset of the wildcard filter reservation’s filterspec.

3.5 Receiver complexity and synchronization of reservation classes

With RSVP, it is the responsibility of each receiver to negotiate the level of resources to be reserved for any data flow that it wishes to receive special QoS delivery. One consequence of assigning this responsibility to receivers is that different receivers in the same session might not use the same Tspec in their reservation requests. This scenario would be especially common in the case of shared-explicit reservation requests since each request might refer to a different set of senders, that is have a different filterspec. Merging of a guaranteed-service reservation with a controlled-load service reservation can be non-trivial if the Tspecs of the reservations are dissimilar. Because of this, RSVP prohibits co-existence of controlled-load and guaranteed service requests within the same session. Consequently, synchronization of reservation styles between receivers of the same session is required which can result in suboptimalities. For example, a receiver which might otherwise be satisfied with the controlled-load class might be forced to use the more resource-intensive guaranteed-service class if that was the reservation class designated for the session.

3.6 Reservation Processing Load at Intermediate Nodes

The receiver-initiated reservation mechanism of RSVP is inefficient with regard to control messages transferred and processing load at intermediate nodes for the initial setup of reservations when the number of receivers is large. To illustrate this consider a source-based multicast tree with a large number of receivers, each of which wishes to receive Guaranteed Service delivery for the data packets sent. In the case of RSVP, each receiver will independently calculate the bandwidth, R that it needs to request to achieve its desired end-to-end delay bound. The requests sent by the receivers will be merged as they are sent upstream. Propagation of a reservation request upstream will cease in 3 cases. First when the reservation reaches the sender. Second, when the reservation fails admission control at any node. Third, when the request reaches a node where the bandwidth reservation at one of the node's on-tree outgoing interfaces is greater than or equal to that contained in the request.



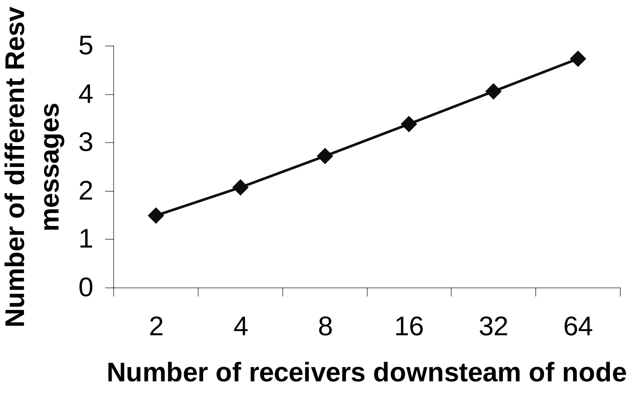


Figure 4: The mean number of Resv message propagated upstream by a node out of interface I1 before steady state is achieved.

Figure 4 shows a node in the source-based tree. This node will receive guaranteed service reservation requests over outgoing interfaces O1-On and will propagate merged reservation requests out of incoming interface I1. The mean number of merged requests, all of which contain progressively larger Rspec values, sent out of interface I1 before steady state is achieved is shown in Figure 4 where it is assumed that each receiver has calculated a different Rspec for its reservation flowspec in order to achieve its target end-to-end delay. Each of these merged requests sent upstream prior to the achievement of steady state will modify the upstream reservation. This is clearly not the optimal approach for setting up reservations. The optimal approach would set up a reservation at each outgoing interface of the intermediate nodes once only for each sender data flow block. We define the start of a sender data flow block as being a step change in either the sender Tspec or the sender-suggested QoS for the data flow[11]. In the case of RSVP, the sender can suggest a change in the QoS class by changing the class of the single service-specific fragment included in the Adspec of Path messages that it generates. The sender could suggest a change in the QoS level within a class(e.g. end-to-end delay within Guaranteed Service class) by some appropriate application layer mechanism.

[11] In order to ensure that the end-to-end QoS is maintained while minimising over-reservations the receiver must decide whether to readjust its reservation whenever it detects a new sender data flow block.

4 Overview of ATM Block Transfer(ABT)

The ITU-T[12] has recently specified block transfer methods[11] for ATM that differ from the traditional ATM connection signalling approach in that they allow QoS to be negotiated in-call on a block by block basis. In these schemes the traffic on a connection is treated as a contiguous series of blocks, each of which contains a number of ATM cells. Successive blocks are delimited by a resource management cell which specifies the Block Cell Rate(BCR) of the block that immediately follows it. The service received by a block at a given BCR is equivalent to a Dedicated Bit Rate(DBR) connection of peak rate equal to the BCR. With ATM Block Transfer(ABT), as with conventional ATM signalling, an end-to-end connection setup phase must take place before transfer of data can occur. During this connection setup phase certain parameters are negotiated between the sender and network; in particular maximum block cell rate and sustainable cell rate(SCR). The maximum block cell rate represents the largest value of BCR that a resource management cell should contain. The sustainable cell rate is a reservation that indicates the long-term average of reserved bandwidth that the network can guarantee.

Once a BCR has been accepted for a given block the service received by the block will be guaranteed provided the block remains within the negotiated traffic envelope as indicated by conformance testing at both the cell-level and block-level. Cell-level conformance testing simply involves checking that the measured peak rate of the connection does not exceed the BCR for the current block. By contrast block-level conformance testing is more concerned with the long-term average of measured traffic which each intermediate node analyses using a leaky bucket type mechanism such that for BCR>SCR credits decrease with time while for BCR<SCR credits increase with time. When the credits fall below a certain threshold the block is deemed to be non-conforming. Under conditions of cell-level non-conformance the network is under no obligation to honour the BCR reservation, although it may continue to provide a reservation at the accepted BCR to the subset of the connections cells that do not exceed it. Non-conformance at the block level may cause the network to initiate renegotiation of the BCR.

ITU-T actually specifies two variants of ATM Block Transfer(ABT), namely Immediate Transmission(IT) and Delayed Transmission(DT). ABT/DT is more akin to traditional ATM signalling in that the sender will not transmit cells of a new block until it has received a positive acknowledgement from the preceding resource management cell associated with that block. By contrast, with ABT/IT network nodes do not generate acknowledgments in response to reservation requests and the sender may transmit cells of a new block immediately after sending the resource management cell containing the requested BCR for that block. With ABT/IT, assuming the connection is conforming at the block-level, a request for an increase in BCR has a specified probability of succeeding. However, should the request fail, all cells of the block are discarded.

5 Limitations of ABT/IT

Although ABT/IT represents a significant improvement over traditional ATM signalling in terms of utilising bandwidth in high speed multimedia environments it is still restrictive in a number of ways, several of which are directly attributable to the inherent characteristics of the underlying ATM network. Here we highlight some of these restrictions.

[12] The International Telecommunications Union is an international organisation which among other things is responsible for the coordination of telecommunication standards.

5.1 Connection establishment phase

With ABT, an initial end-to-end connection establishment phase is required before data transfer can take place. This adds complexity and latency to the data transfer which may prove significant for short-lived data flows.

5.2 Limited QoS control

With ABT, the QoS control is restrictive in 2 ways. First the only class of service offered is an emulated DBR service. Second, although the sender may control the bandwidth received on a block by block basis, other QoS parameters such as end-to-end delay are outside the control of the sender since they are fixed parameters associated with the DBR service provided by the underlying ATM network.

5.3 Idle Overhead

The price ABT/IT pays for offering a specific probability of reservation acceptance when a connection is conforming is the fact that a connection will consume resources in accordance with the SCR traffic descriptor negotiated at connection-establishment even if the current BCR is zero

5.4 Rigid Filter Specification

In ABT/IT, the filter specification of an RM cell is always taken to be the VPI/VCI combination in the ATM cell header of the RM cell. Given that ATM signalling does not facilitate the setup of either multipoint-to-point or multipoint-to-multipoint virtual circuits, ABT/IT is thus unable to support shared reservations, that is reservations where the reserved bandwidth is shared between a number of senders. In principle ABT/IT could be used over point-to-multipoint ATM VCs although the ITU standard I.371 does not specify this at present.

5.5 Hard failure

In ABT/IT when a request for an increase in BCR fails all cells of the block are discarded. While such action may be acceptable if the cells contain data, it is unsuitable for real-time traffic as it will result in a complete loss of service as perceived by the user.

6 Dynamic Reservation Protocol (DRP)

We now present our scheme called DRP(Dynamic Reservation Protocol) for setting up reservations along end-to-end paths in communications networks. DRP has been designed to minimise or eliminate certain limitations of previous reservation mechanisms which were outlined in the previous sections. It combines many principles of RSVP such as soft-state, OPWA and merging of messages with the dynamic sender-initiated reservation concept of ABT/IT to achieve the following goals:

- High control dynamics to achieve efficient bandwidth usage.
- Optimal bandwidth usage for shared reservations.
- Scalability of router-state with regard to number of senders and receivers.
- Scalable and simple approach to One Pass With Advertising (OPWA).
- Minimal receiver complexity.
- Minimal number of messages to set up reservations for new sender data flow blocks in large-scale multicast sessions.
- Support for heterogeneity of reservation QoS classes among receivers of a multicast session.
- No need to synchronize reservation styles among hosts of a multicast session.

DRP allows reservations to be set up ‘on-the-fly’ by sending Reservation packets, RES in-line with the data flow. In this respect, DRP is similar to ATM’s ABT/IT outlined in section 4. However, unlike ABT/IT there is no requirement for a connection-establishment phase with DRP. Instead, an application wishing to receive end-to-end QoS simply sends a RES packet straight away and is free to send the first data packet as soon as it wishes thereafter. As soon as the RES packet is processed by each node it reaches it will set up the desired reservation assuming that it is accepted by admission control within the node. Also, unlike ABT/IT, a sustainable cell rate for a data flow is not specified with DRP. This leads to two notable differences in the operation of ABT/IT and DRP. First, in ABT/IT a reservation request(BCR) will be accepted with a specified probability assuming the connection is conforming and the requested BCR does not exceed the value of the maximum BCR negotiated at connection-establishment[13]. By contrast, with DRP, the probability of acceptance of a reservation request for a new reservation or an increase in an existing one is not quantified. Of course any reservation request that decreases an existing reservation will always be accepted. A second notable difference concerns the QoS commitments delivered to the data flow once a reservation has been accepted. With DRP, the reservation will stay in place provided the source sends periodic refresh reservations to prevent soft-state timeout and no route changes or network errors occur. Moreover this will be true even if the data flow violates the traffic specification agreed at reservation setup. By contrast with ABT/IT if a data flow is detected as non-conforming at the block-level then the network is free to initiate renegotiation of the reserved BCR.

In addition to RES packets sent in-line with data, DRP uses Return (RTN) packets that are reverse-routed up the tree to provide the intermediate routers/switches and sender with certain feedback and end2end path information. DRP is applicable to both unicast and multicast scenarios but in the following sections we concentrate on the more complicated multicast case. As with RSVP, all flow-specific router node state setup using DRP is so-called ‘soft-state’ which means that it will be deleted in the absence of any refeshes within a certain timeout period.

7 Design Principles

7.1 Sender initiated reservations

The use of in-line reservation packets allows the sender to setup new reservations, or alter existing reservations, on demand at any point in the data transfer. This makes it possible to achieve a very close match between the instantaneous service provided by the network and the instantaneous requirements of the data flow. As a result, network resource usage can be minimised. These benefits are particularly prominent for stop/start data flows since the resources can be freed during the quiet periods and re-installed on a just-in-time basis at the start of each activity burst. By contrast with RSVP or traditional ATM signalling[14] the low QoS control dynamics would usually preclude such action. The dynamic setup/teardown approach that can be used by DRP on a per-activity burst basis for stop/start flows facilitates a new kind of reservation paradigm in which at the user level, a reservation request is never denied, but at the network level a reservation request might be denied on a block by block basis depending upon available resources at that moment in time. We now present an example of where such an approach might be useful.

Suppose a company has a private IP network which integrates voice and data and uses a

[13] However the probability that the initial connection request will be accepted at the requested SCR is not quantified.

[14] Traditional ATM signalling(e,g, Q.2931 and UNI) requires end-to-end handshaking.

1Mbps wide-area link to connect between its European and American offices. In order to support the QoS requirements of the voice traffic a resource reservation protocol is used within the IP network and the network manager configures the routers on the 1Mbps link to limit the reserved bandwidth at the IP-layer to 320kbps. The company's voice-access devices use the G.729 compression scheme in addition to voice activity detection and silence suppression. During a talk-spurt this produces a frame containing 10 octets once every 10ms to give a bit rate of 8kbps. In order to minimise % IP header overhead, 4 of these frames(40 bytes) are placed inside each IP packet. The IP header overhead is equal to 40 bytes in total(IP, UDP and RTP headers) to give a total IP layer bandwidth of 16kbps during a talk spurt. No packets are transmitted during a silent period. As described in [13] a reasonable model for a voice source can be achieved by assuming an exponential distribution for active and quiet periods with mean lengths of 350ms and 650ms respectively. This represents a voice activity rate of approximately 35%. Let us assume that limitations of the voice activity and silence suppression increase this to 40% in terms of packets produced. Suppose the following token bucket is used at the source, p=r=2Kbytes/s, b=80bytes(1 packet). This token bucket will not delay packets at all at the source and the low value of b ensures that if WFQ mechanisms are used in the network to realise controlled-load service then the WFQ delay of b/r will be small(40ms). Let us assume that the routers on the 1Mbps link use the simple sum admission control. In this case a reservation request will fail when the sum of the token bucket rates of all existing reservations plus the desired reservation exceeds 320Kbps(40Kbytes/s). Here we could use DRP in one of two ways. First, DRP could be used like a conventional reservation protocol, that is a reservation is requested at the start of a call and if accepted it isn't removed until the end of the call. In this case, once the number of calls receiving reservations is equal to 20 any further reservation requests will be rejected. This is an 'all or nothing approach'. Depending on company policy, the dynamic setup/teardown approach of DRP may be preferable in which case a reservation at the user level would never be denied although the quality of each user-level reservation would degrade approximately equally once the number of calls exceeded 20. When the number of user-level calls, N exceeded 20 the probability, Pfail(N) of a given on-the-fly reservation request at the beginning of an activity burst being rejected is given, to a first approximation, by equation (3)

$$Pfail(N) = \sum_{n=20}^{N-1} p^n (1-p)^{N-1-n} \frac{(N-1)!}{n!(N-1-n)!} \qquad (3)$$

where p= 0.4=probability that a given call is engaged in an activity burst

Whenever a reservation for an activity burst fails, the packets of the activity burst will receive best-effort service. Figure 5 illustrates equation (3) graphically.

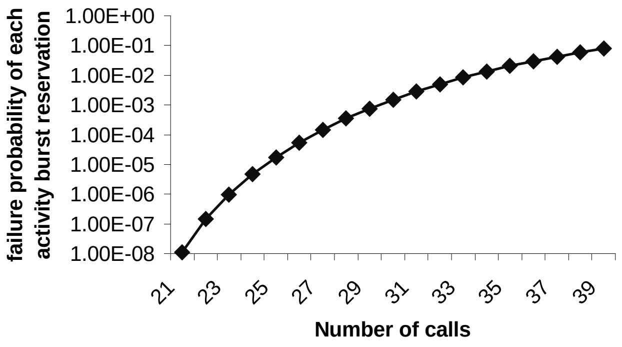
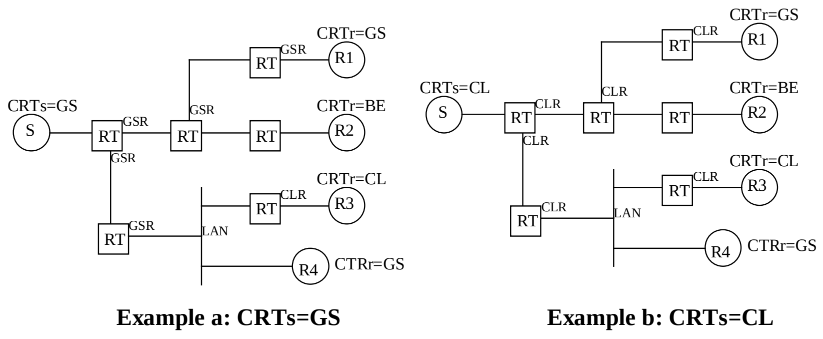


Figure 5: Failure probability of each activity burst reservation

7.2 Heterogeneity of QoS reservation classes between receivers of the same session

DRP allows the sender to request, ‘on-the fly’, intserv’s Controlled-Load Service[18] and Guaranteed Service[14] by sending a RES packet in-line with data. The sender designates a ‘ceiling’ reservation class (or Type), CRTs to each data flow block as well as an associated end-to-end QoS level. In addition each receiver specifies a ‘ceiling’ reservation class, CRTr which represents the highest quality reservation class it is willing to receive. The Guaranteed Service reservation type is taken to be the highest quality reservation class, followed by Controlled-Load service with ‘best-effort’(no reservation) being the lowest. Assuming that sufficient end-to-end resources exist, the reservation type received by a receiver will then be given by MIN(CRTs, CRTr). Each receiver is free to change its value of CRTr at any time by sending a RTN packet upstream containing the new value of CRTr. Merging of RTN packets as they travel up the tree ensures that the type of reservation, RToi installed at each outgoing[15] interface is given by MIN(CRTs, maxCRTr) where maxCRTr is the maximum value of CRTr in all RTN packets received on that outgoing interface. In addition the value of CRTr in the RTN packet sent upstream by a node is given by the maximum of the RToi values associated with each of the node’s outgoing interfaces. It is worth mentioning here that although we accommodate both guaranteed service and controlled-load service classes within the same multicast session the only type of ‘class merging’ at intermediate nodes simply consists of converting a guaranteed service traffic stream into a controlled-load service stream. This is trivial and is done by installing a controlled load reservation with a reservation Tspec equal to the Tspec of the incoming guaranteed service stream. Figure 6 shows some examples of reservation class heterogeneity within the same multicast session.



CLR	Controlled-load reservation
GSR	Guaranteed service reservation
CL	Controlled-load
GS	Guaranteed Service
BE	Best-effort
CRTs	Ceiling reservation type(sender)
CRTr	Ceiling reservation type(receiver)
LAN	Local Area Network
S	Sender
Rn	Receiver
RT	Router

Figure 6: Heterogeneity of reservation classes between receivers of same session.

Table 1 compares the DRP approach to providing end-to-end delay bounds with those of RSVP and ABT/IT. DRP is similar to ABT/IT in the sense that for a given sender to a multicast session the target end-to-end delay bounds will be identical for each receiver[16]. The difference is that with DRP the sender can control what this delay bound will be whereas with ABT/IT the delay bound is a feature of the QoS class provided by the network and cannot be

[15] The terms incoming and outgoing with regard to interfaces refer to the direction of data transfer. RES packets which follow the same path as the data packets always arrive on one of the node’s incoming interfaces whereas RTN packets which follow the reverse path to the data packets always arrrive on one of the node’s outgoing interfaces. In the case of a source-based tree the terms incoming and outgoing interface are unambigous. In the case of a shared tree an outgoing interface is also an incoming interface for the same multicast group. However for a given packet arrival at a shared tree node the incoming and outgoing interfaces are clearly defined as follows. The interface on which the packet arrived is the incoming interface for that packet. The rest of the on-tree interfaces are the outgoing interfaces for that packet.

[16] In the case of DRP we are only referring to those receivers that have actually requested an end-to-end delay bound, i.e. those with CRTr=GS.

controlled by end nodes. Like DRP, RSVP facilitates end node control of end-to-end delay albeit by receivers rather that senders. RSVP allows receivers finely grained control within a reservation class at the expense of added receiver complexity together with lack of support for reservation class heterogeneity among receivers of a session. By contrast, reservation class heterogeneity is supported in the DRP approach whereby the sender specifies a ‘ceiling’ QoS class and associated level for all receivers who each then have the option of downgrading QoS class. Although DRP does not allow receivers any finely grained control within a QoS class, we do not believe such a feature is necessary anyway and certainly not with regard to end-to-end delay bound. We argue that any such end-to-end delay bound is determined by the nature of the sender’s traffic stream and as such the sending application is the node most qualified to specify what it should be. The receivers simply need to be told what this end-to-end delay bound is so that they can set their playout buffers accordingly. Furthermore, removing receiver-control of end-to-end delay in DRP enables merging of RTN messages and RTN state in routers to ensure scalability to large multicast sessions as described in section 7.3.

	RSVP	ABT/IT	DRP
Sender control of delay bound	**No**	**No**	**Yes**
Receiver control of delay bound	**Yes**	**No**	**No**

Table 1: Comparison of different schemes with regard to provision of end-to-end delay bounds

Another advantage to using sender-based reservations rather than receiver-based reservations is a reduction in the volume of processing required at intermediate nodes in order to set up the reservations for each data flow block sent onto a multicast tree. As described in section 3.6, with receiver-initiated reservations as in RSVP each reservation at an intermediate node may be modified several times at the start of a new data flow block before it settles down to its final value for the data flow block. By contrast with DRP, typically as the RES packet passes down the multicast tree it will setup reservations only once on each on-tree outgoing interface it passes through.

7.3 Merging of RTN messages

DRP uses RTN messages which are reverse-routed up the distribution tree from receiver(s) to sender(s) for the following purposes:

1) To accumulate certain path characteristics information which is used by a node when calculating the level of resources to reserve.
2) To allow a receiver to downgrade its received reservation class below that suggested by the sender.
3) Optional feedback information that may be used to convey information to intermediate routers in cases where the end-to-end delay bound was not satisfied in the first pass of a RES message.

With respect to 1) above, RTN messages fulfill a similar role to Path messages in RSVP.

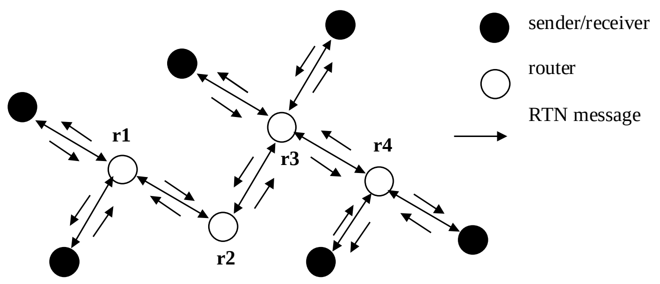


Figure 7:DRP RTN messages on a shared tree

For large-scale multipoint-to-multipoint applications the use of a single shared tree for all senders to a multicast group will consume far less resources[3] than a separate source-based tree for each sender to the group. Because of this, a single shared tree is likely to be preferable to a mesh of source-based trees in such scenarios. In such cases DRP displays much more favourable scalability characteristics than RSVP. With DRP, full merging of RTN messages is possible and ensures that the number of RTN messages on each link of a shared tree in the steady state is never more than two(one in each direction) every refresh interval as shown in Figure 7. By contrast, with RSVP the total number of Path messages on each link of a shared tree per refresh period in the steady state is equal to the number of senders as illustrated in Figure 1.

However perhaps a more important benefit of the DRP approach is the fact that the number of RTN state entries in each on-tree router is determined entirely by the number of outgoing interfaces and the number of next hops per outgoing interface. Hence in DRP the number of RTN state entries in each router of a multicast shared-tree never becomes an issue no matter how many hosts are sending to the group. By contrast, with RSVP the number of Path state entries in each router of a multicast shared-tree is equal to the number of senders to the group and consequently may become significant for large-scale multipoint-multipoint applications.

There are some circumstances in which the amount of flow-specific router-state associated with path characteristics and OPWA produced by DRP is likely to be higher[17] than with RSVP. This will be the case for either a shared-tree with a small number of senders or for a source-based tree. Of these two cases the only one that may cause scalability concerns is that of a mesh of source-based trees used to provide communication within a multicast group. However, given that such an approach is unlikely to be used for large-scale multicast applications then the flow-specific state for either RSVP or DRP should still remain at a manageable level.

7.4 Shared-Session Reservations

As mentioned in section 3.3, shared-style reservations that are setup using RSVP can be sub-optimal for 2 reasons. First, the reservation stays in place during quiet periods. Second, during active periods the reservation may sometimes or always be larger than necessary to meet the agreed QoS for the data flow currently using it. Both of these inefficiencies are obviated by the high control dynamics of DRP's sender-based 'on-the-fly' signalling since it allows the shared-session[18] to be handled using sender-specific reservations that are installed and torn down on-the-fly at the start and end of each activity burst respectively as exemplified in Figure 8.

---

[17] But higher by much less than an order of magnitude. By contrast, in the case of a shared tree used in a large-scale multicast application, the amount of flow-specific state in each router with DRP will be several orders of magnitude less than the amount of flow-specific state with RSVP.

[18] where only one sender to the session transmits at once.

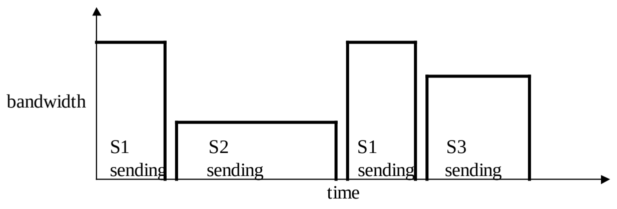


Figure 8: Bandwidth of just-in-time sender-specific reservations using DRP in the case of a shared session with no sender transmission overlap.

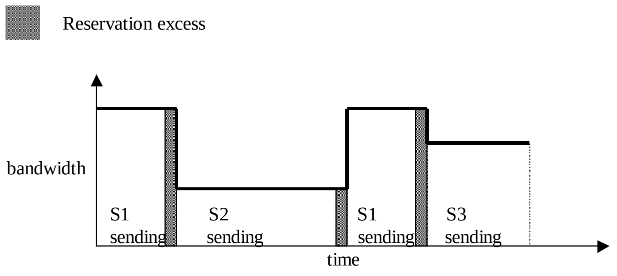


Figure 9: Bandwidth of a simple shared reservation for a shared session with no sender transmission overlap.

However, in such cases it may be possible for end users to detect a degradation in QoS at the start of each activity burst due to the finite time required to install the 'just-in-time' sender-specific reservation. As a result of this, DRP provides an additional reservation mode known as Sender-Specific with Residue(SSR) mode. To understand the reasoning behind the SSR mode of operation, which we will explain shortly, consider Figure 9 which shows a simple way in which we could attempt to minimise the QoS disruption at the start of each activity burst in a shared-session. Figure 9 shows a simple shared reservation which is altered at the start of each new activity burst to reflect changes in the sender characteristics. This approach minimises QoS disruption at the start of an activity burst in a shared session by attempting to ensure the presence of a free reservation that the 'just-in-time' reservation request can use as a starting point.

While the simple shared reservation mechanism just described works well in the example of Figure 9, it will suffer from under or over-reservations in cases where it is possible for multiple senders to be simultaneously active which might occur in the absence of appropriate conference control mechanisms. This deficiency of a simple shared reservation approach is highlighted in Figure 11 for the example traffic pattern of Figure 10. Bearing these potential hazards in mind, DRP provides an alternative reservation mode to the standard sender-specific(SS) mode known as Sender-Specific with Residue(SSR). In SSR mode each sender makes a sender-specific reservation at the start of each activity burst and sends a teardown request at the end of the activity burst. When a sender's teardown request[19] reaches an outgoing interface of a router the SSR reservation of the sender will only be removed if at least one other SSR reservation for the session is in place in the router at that outgoing interface. Otherwise the sender's SSR reservation is left in place, but a status flag associated with the reservation is set from 'active' to 'passive' state. A subsequent arriving SSR reservation request can then claim the 'passive' reservation and use it as a starting point in establishing the new reservation.

[19] A teardown request is simply a RES packet with the token bucket rate of the Tspec set to 0. It indicates to the intermediate routers that the sender no longer requires a reservation for its data packets.

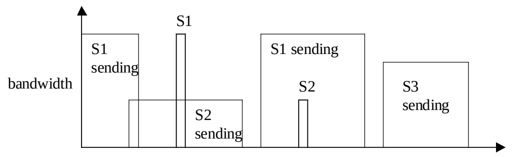


Figure 10: Traffic pattern for a shared session with some sender transmission overlap

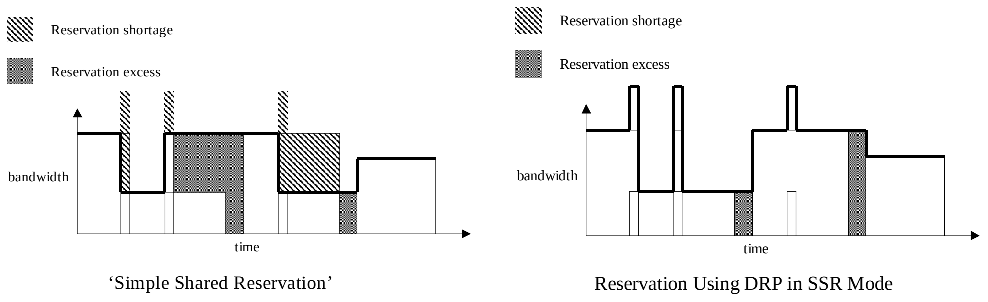


Figure 11: Bandwidth reserved for a shared session with some sender transmission overlap.

Figure 11 illustrates the operation of SSR mode for the traffic pattern of Figure 10. When only one sender is active at once, the operation of SSR mode is essentially the same as the simple shared reservation of Figure 9 and so will suffer from resource wasteage when all of the senders go simultaneously quiet[20]. However should the senders go simultaneously quiet for extended periods of time the soft-state nature of the reservation will cause it to eventually timeout and be removed. In cases where more than one sender is simultaneously active, the operation of SSR mode is essentially the same as SS mode and so cannot suffer from the under-reservation problem that exists with the simple shared reservation in Figure 11.

A notable advantage of DRP compared to RSVP is that DRP allows co-existence of both modes of reservations within the same multicast session while with RSVP each receiver within a given multicast session must choose the same reservation style. The way in which co-existence of reservation modes within the same multicast session is accommodated in DRP is summarised as follows.

When a reservation request arrives at an on-tree incoming router interface
it is copied to each on-tree outgoing interface where the following steps are applied:

```
If reservation for that sender already exists
        Set reservation's mode flag(0=SS, 1=SSR) according to mode field in RES packet.
        Adjust reservation level according to RES packet.
        Set reservation's status flag to 1 (active).
Else if mode field in RES packet indicates SS
        Create a new reservation according to filter spec and information  in RES
         packet.
        Set reservation's mode flag to indicate SS.
        Set reservation's status flag to 1 (active).
Else if a SSR reservation exists with state = 'passive'
        Set filter spec of that reservation to the sender of the RES packet.
        Adjust reservation level according to RES packet..
```

[20] For example in a multimedia conference if the audio channel used a different multicast group to the other multimedia traffic components there might be significant periods of time where the audio channel was quiet. By contrast in an audio-only conference the channel is unlikely to be quiet for any lengthy period of time.

```
        Set reservation's status flag to 1 (active).
Else
        Create a new reservation according to filter spec and information in RES packet.
        Set reservation's mode flag to indicate SSR.
        Set reservation's status flag to 1 (active).
```

When a reservation teardown arrives at an on-tree incoming router interface
it is copied to each on-tree outgoing interface where the following steps are applied:

```
If reservation mode is SS
        Remove reservation
Else if total number of installed SSR reservations including this one is greater than one
        Remove reservation.
Else set reservation's status flag to 0(passive)
```

7.5 Reservation request admission control.

Apart from the initiator of QoS requests(sender vs receiver) there are two other notable differences between the reservation mechanisms used by RSVP and DRP.

1) **Explicit vs implicit reservation requests**
With RSVP, for both Controlled-Load and Guaranteed Service reservations, the request explicitly informs the node of the level of resources to reserve. The same can also be said of a Controlled-Load Service reservation request using DRP. However with a DRP Guaranteed Service request the RES packet requests the level of resources to reserve implicitly by informing the router of the accumulated delay bound thus far, together with the target delay bound and the sender traffic characteristics. Using this information along with path information obtained from RTN packets each router is able to estimate the local reservation required and update the accumulated delay bound in the RES packet accordingly. Each router calculates a local reservation bandwidth which, if also reserved in each subsequent router, will lead to an overall delay bound equal to the target delay bound. However, should any router have insufficient resources to install the calculated local reservation bandwidth then it reserves the most that it can and the attempt is only referred to as a 'reservation failure' if the resultant accumulated delay thus far exceeds the target delay bound. If the attempt is not a so-called 'reservation failure' then the RES message is treated the same regardless of whether the level of local reservation initially calculated could be reserved or it couldn't. This is because even in the latter case the target end-to-end delay bound may still be met since each subsequent router will automatically attempt to reserve more in order to compensate. The action taken in the event of a so-called 'reservation-failure' is discussed next.
2) Action in event of reservation failure:
With RSVP, any request that fails admission control at a router is not propagated any further along its path towards the sender(s) and a ResvErr message is sent to affected receiver(s). By contrast, whenever a DRP node cannot satisfy the calculated local reservation, and the maximum level of resources that it can reserve is so low that it prevents the target end-to-end QoS from being satisfied, the request is not rejected. Instead, the node reserves as much resources as possible and sets specific QoS violation bits in the RES header while updating the other header fields in the usual manner before propagating the RES message down the distribution tree.

In the event of a DRP Controlled-Load Service 'reservation-failure', the node sets a bit, known as the QoSvoid bit, to 1. The RES packet is handled in the usual way by all subsequent routers encountered, although the presence of the non-zero QoSvoid bit will be an indication to receivers that the end-to-end QoS could not be achieved.

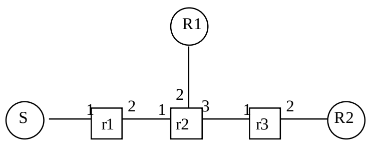
In the event of a Guaranteed Service request where the node could not reserve more than the mean rate of the sender's traffic, it becomes impossible to guarantee either lossless transmission or conformance to the target delay bound, or even the Controlled-Load Service. Consequently three flags should be set in the RES packet, namely the delayvoid, lossvoid and QoSvoid bits. When downstream routers see a RES packet with (CRTs=GS, delayvoid=lossvoid=1) then they take the 'effective CRTs' to be Controlled-Load Service(CL) and attempt to install a Controlled Load Service Reservation.

In the event of a Guaranteed Service request where lossless transmission has not yet been precluded [21] but the accumulated bound at a node exceeds the target delay bound for the first time, the action taken is as described in section 7.6.

In the case of Guaranteed Service reservations in a large multicast tree, there are some interesting differences between DRP's sender-based reservations and RSVP's receiver based reservations. To illustrate these differences we refer to the example topology of Figure 12. This shows the logical connectivity of a multicast session between a sender, S and two receivers, R1 and R2. These end nodes are interconnected via routers, r1-r3 and all links are 10Mbps Ethernet. The exported C and D error terms[14] from the routers are shown together with the token bucket parameters of the Sender Tspec. We will assume that both receivers, R1 and R2 require a queuing delay bound of 300ms to sender S. With RSVP , each receiver calculates an Rspec to be reserved in each router along the end-to-end path in order to achieve its delay bound. In this example, R1 calculates an Rspec of 327.6Kbytes/s while R2 calculates an Rspec of 490.89 Kbytes/s. At router r2 these two requests are merged so that the Rspec propagated to router r1 is 490.89Kbytes/s. Packets from S to receiver R1 will now experience a reservation bandwidth of 490.89kbyte/s in router r1, interface 2 rather than the requested 327.6 Kbytes/s. This will cause a reduction in R1's end-to-end delay meaning that theoretically the bandwidth reserved for R1 in r2, interface 2 could be decreased from the initially calculated value of 327.6 Kbytes/s while still achieving R1's end-to-end delay bound. However R1 does not facilitate such a mechanism and in this example R1's end-to-end delay bound will be less than, rather than equal to, that requested. By contrast, DRP keeps a running total of end-to-end delay bound which it updates at each hop and uses to calculate the local reservation required to stay on course for the desired end-to-end delay bound. As a result, in this example DRP automatically readjusts the reservation level in r2, interface 2 where 247.7 Kbytes/s is then reserved rather than 327.6 Kbyte/s as in RSVP.

For multicast examples such as this where the routers and links are homogeneous(same values of C, D error terms and link propagation delay) RSVP will never use fewer resources than DRP. However in environments with heterogeneous routers and links the matter is not as straightforward. With DRP, in a multicast environment the bandwidth to be reserved at each node is calculated based on, among other things, worst-case merging(see section 9) of path characteristics received from RTN messages. The effect of this worst-case merging can be for DRP to make an over-estimation in the local reservation. Any such over-estimation will cause a reduction in the local node queuing delay. In turn this will mean that DRP allows an increase in the local queuing delay at nodes further downstream whose reservations do then not need to be as high. This 'skewing' of the bandwidth reservation pattern in multicast sessions whereby nodes closer to the sender are more likely to over-estimate their local reservations can theoretically cause an increase in the overall reservation bandwidth required in the multicast tree. This is an area for further study.

[21] That is, every node so far has been able to reserve in excess of the mean rate of the sender's traffic



interface	Rspec reserved(Kbytes/s)	
	RSVP	DRP
r1 interface 2	490.89	490.89
r2 interface 2	327.6	247.7
r2 interface 3	490.89	490.89
r3 interface 2	490.89	490.89

Token bucket parameters	
Peak rate(p) Bytes/s	200K
Token bucket rate(r) Bytes/s	50K
Token bucket depth(b) bytes	1K
Router error terms	
C(bytes/s)	48,000
D(μs)	1200
Link propagation delay	0

Figure 12: Guaranteed Service Reservations using RSVP and DRP

7.6 Using feedback to increase the probability of achieving end-to-end delay bound

In the case of Guaranteed Service reservations the adoption of a strategy whereby an end-to-end reservation is only permissible by installing an equal reservation in each router reduces the chances of meeting a target end-to-end delay bound. This characteristic has been noted by the designers of Guaranteed Service and exploited to a reasonable degree through the introduction of a slack term[17] into the reservation flow specification. Use of the slack term enables higher reservations to be made between the receiver and the bottleneck router to compensate for the increase in delay incurred by the lower reservation in all routers between, and including, the bottleneck router and the sender. However it does not permit the reservation to be increased once it has passed through the bottleneck router on its way towards the sender. Such a restriction can sometimes prevent RSVP from achieving the target delay bound even on a path that actually contains enough resources to meet the target end-to-end delay bound. Unlike RSVP, DRP employs cooperation and feedback between routers to ensure that if a given end-to-end path is capable of supporting a specific target delay bound then DRP will always meet the target delay bound. In the DRP example of

Figure 13, each router is able to reserve a bandwidth in excess of the mean rate of the sender's traffic but at router r3, accumulated delay for the first time is in excess of the target delay bound, Dt. Consequently, router r3 sets a bottleneck flag associated with its local reservation as well as setting a delayvoid flag in the forwarded RES message. When r4 receives the RES message it notices that the delayvoid flag has been set to 1 and so as a result reserves the maximum reservation that it can(subject to any installed policy decisions) in order to minimise its contribution to the accumulated delay. When R receives the RES message it notices that the target delay bound has been exceeded and immediately issues a RTN packet containing the amount by which the target delay has been exceeded in a field in the packet known as the excess delay field. In addition, a bit in the packet known as the bottleneck flag, is set to 0. This RTN packet is reverse-routed up the tree but is ignored at each router until its bottleneck flag has been set to 1 which will occur when it reaches the interface of a router, r3 in this example, in which the bottleneck flag has been set to 1 for the installed reservation. The RTN packet will then travel hop by hop towards S with an attempt being made at each hop to eliminate the excess delay or at least reduce it as much as possible by increasing the level of the local reservation on the appropriate outgoing interface. If a router succeeds in reducing the excess delay to zero then the RTN packet will cause no further alterations in local reservations on the rest of its journey towards S. In this example, r2 is able to increase its local reservation and cause a reduction in its local queueuing delay of de1 which is then subtracted from the excess delay field before sending the RTN packet to r1 which manages to increase its local reservation sufficiently to completely eliminate the excess delay. The target end-to-end delay bound has now been achieved.

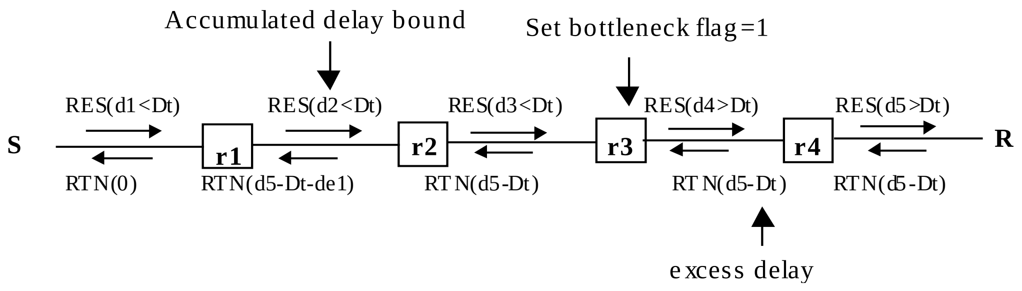


Figure 13: Use of DRP feedback Guaranteed Service reservation mechanism to maximise chances of meeting target delay bound.

In the next two sections we present details of the main fields required in RES and RTN packets together with the processing rules in order to provide the basic functionality of DRP described in the previous sections.

8 Reservation(RES) message

The IP destination address of the IP datagram encapsulating a RES message is equal to the session destination address while the IP source address is equal to the initial sender of the RES packet. The IP router alert option is used to ensure that the RES message is intercepted and processed by intermediate nodes.

8.1 RES message Common Part

- **Session** – (object defined in the RSVP protocol) – it contains the destination address, transport layer protocol identifier and transport layer destination port.

- **Phop -** ('previous hop' object defined in the RSVP protocol) – it is the identity of the last DRP-capable logical outgoing interface to forward this message. The Phop object consists of the pair (IP address, logical interface handle) and is required to install Phop state in the router to ensure correct reverse routing of RTN messages.

- **Sender Template -** (object defined in the RSVP protocol) – it is a filter specification identifying the sender. It contains the IP address of the sender and optionally the sender source port(in the case of Ipv6 a flow label may be used in place of the sender port)

- **timestamp** field - this is stamped with the time of the local node clock just before forwarding the RES packet to the next hop(s) down the distribution tree. It is used to calculate dnext as described in section 13.

- **CRTs** field(2 bits) - this identifies the ceiling reservation class of the sender. 11 indicates Guaranteed Service, 10 indicates Controlled-Load Service, and 00 indicates best-effort. 01 is currently unspecified although may at some time be used for a new service with quality in between best-effort and Controlled-Load Service.

- **Tspec** describing sender's traffic characteristics using the following token bucket representation as given in [14].
  - p = peak rate of flow (bytes/second)
  - b = bucket depth (bytes)
  - r = token bucket rate (bytes/second)

- m = minimum policed unit (bytes)[22]
- M = maximum datagram size (bytes)

- **end2end delay** field - this gives the current delay from when a packet was transmitted by the initial sender until it is due to arrive at the incoming interface of the current next hop.

- **Mode** field(1 bit) – this identifies the reservation mode. A value of 0 indicates SS mode while a value of 1 indicates SSR mode.

- **QoSvoid** bit – if set to 1 this indicates that no QoS guarantees can be offered.

8.2 RES message Guaranteed Service object

If CRTs = 11(Guaranteed Service) the RES packet will also contain a Guaranteed Service object comprising the following:

- **CSum -** accumulation of C values since last upstream reshaping point (see [14]).

- **DSum -** accumulation of D values since last upstream reshaping point (see [14]).

- **target-bound** field which indicates the target end-to-end delay bound of the sending application.

- **accumulated-bound** field which indicates the installed delay bound between sender and the incoming interface of the current next hop.

- **Flags** field containing
  - ⇒ **delayvoid bit** (If set, this bit is an indication to the receiver that the target delay bound cannot be guaranteed)
  - ⇒ **lossvoid bit** (If set, this bit is an indication to the receiver that a loss-free service cannot be guaranteed)

8.3 Node Processing of RES messages

When a node receives a RES packet for an end-to-end reservation attempt at which QoS violation has already occurred or which occurs following processing of the RES packet, the behaviour of the node is as described in section 7.5. Otherwise the processing of the packet is as described in the remainder of this section.

Upon receipt of the RES message the node passes it to admission control which then determines the reservation that needs to be made at each of the outgoing interfaces. The reservation class is given by MIN(CRTs, CRTr) where CRTr is the 'merged' value of CRTr for the appropriate outgoing logical interface as obtained from MRTNSE which is described in the next section.

If the reservation request is for the Controlled-Load Service then the reservation is governed entirely by the sender Tspec contained within the RES message. By contrast if the reservation request is for Guaranteed Service then the reservation is described by the combination of the sender Tspec and a reservation bandwidth, R that the admission control mechanism needs to determine using the following equations as given in [14]

$$Qdelay_{end2end} = \frac{(b-M)(p-R)}{R(p-r)} + \frac{(M+Ctot)}{R} + Dtot \quad \text{(case p > R} \geq \text{r)} \qquad (4)$$

[22] Policing will treat any IP datagram less than size m as being of size m.

$$Qdelay_{end2end} = \frac{(M + Ctot)}{R} + Dtot \qquad \text{(case } R \geq p \geq r\text{)} \qquad (5)$$

Admission control obtains the parameters M, p, b and r from the sender Tspec contained in the RES message. The value of Ctot is given by the sum of the router's local C value and the merged Ctot value as obtained from the MRTNSE(see next section) for the relevant outgoing interface. Likewise the value of Dtot in the above equations is given by the sum of the router's local D value and the Dtot value in the MRTNSE for the relevant outgoing interface. To obtain the value of Qdelay to insert into the above equations, admission control uses the relationship given in equation (6)

Qdelay = target-bound - accumulated-bound – dnext – propdelay (6)

where the target-bound and accumulated-bound are obtained from the corresponding fields in the RES packet, and propdelay and dnext are obtained from the MRTNSE for the outgoing interface. The value (propdelay+dnext) represents downstream propagation delay and the constituent parameters are explained in more detail in the next section.

Once the resultant value of Qdelay has been substituted into equations (4) and (5) along with the other mentioned parameters, a value of R to be installed at the outgoing interface is obtained.

Regardless of the reservation class, that is Controlled-Load or Guaranteed Service, if processing of the RES does not result in either the installation of a new reservation or a modification of an existing reservation(i.e. the RES packet was simply a refresh) then the soft-state timer for the reservation is simply reset. Otherwise, the reservation request is propagated immediately down the distribution tree after updating the appropriate fields in the packets header as follows. The end2end delay field in the RES packet is increased by adding to it the following:

- The propagation delay, dnext for the next hop
- An estimate of the current local queuing delay(for the relevant outgoing interface) for data packets of the flow to which the RES packet refers.

In addition, if CRTs=11 the following updates must be made to the RES packet:

- Add the following to the accumulated-bound field of the copied packet.
  - ⇒ The propagation delay, dnext for the next hop
  - ⇒ The installed local queueing delay bound(for the relevant outgoing interface) for data packets of the flow to which the RES packet refers. This local queuing delay bound is obtained by inserting the reserved value of R into equation (7) along with the local values of C and D.

$$Qlocal = \frac{C}{R} + D \qquad (7)$$

- If reshaping to the sender Tspec is being performed at the outgoing interface
  then set Csum=Dsum=0.

  Else

  Add the local value of C to the CSum field
  Add the local value of D to the DSum field

Once updating of the fields is complete the timestamp field is now set equal to the local clock before forwarding the RES packet to each next hop down the routing tree.

9 ReTurN(RTN) message

The IP destination address of the IP datagram encapsulating an RTN message is equal to the IP address of a previous hop node, the identity of which is obtained from installed Phop state obtained from RES messages, while the IP source address is equal to the IP address of the node out of which the RTN message was sent.

9.1 Common part

- **Session** – as for RES message.

- **Nhop** - (object defined in the RSVP protocol) – it is the identity of the DRP-capable logical outgoing interface that sent this message. The Nhop object consists of the pair (IP address, logical interface handle)

- **Sender address** – the combination of this field and the session object identify a source-based tree. In the case of a shared tree this field is ignored and should be set to all 0's.

- **timestamp** field which is stamped with the time of the local node clock just before sending the RTN packet to the previous hop up the distribution tree. This is used in calculation of dnext as described in section 13.

- **CRTr** field(2 bits). This field indicates the receiver's ceiling reservation class.

- **timedelta** field **-** this is used in calculation of dnext as described in section 13.

- **propdelay** field. This equals the data packet propagation delay along the 'worst-case rate-independent delay path'[24] between the node incoming[23] interface out of which the RTN packet was sent and each receiver downstream of that incoming interface.

- **pathMTU** field. This equals the minimum pathMTU value between the incoming interface out of which the RTN packet was sent and each receiver downstream of that incoming interface.

- **Ctot** field This equals the maximum accumulated Ctot value along the paths between the incoming interface out of which the RTN packet was sent and each receiver downstream of that incoming interface. The C error term is defined in the Guaranteed Service specification[14].

- **Dtot** This equals the maximum accumulated Dtot value along the 'worst-case rate-independent delay path'[24] between the incoming interface out of which the RTN packet was sent and each receiver downstream of that incoming interface. The D error term is defined in the Guaranteed Service specification[14].

[23] The term 'incoming' refers to the direction of data flow. RTN packets are reverse-routed up the distribution tree in the opposite direction to the data flow and so are always sent out of so-called incoming interfaces.

[24] The 'worst-case rate-independent delay path' is that path with the maximum value of (Dtot+propdelay)

- **path bandwidth** This equals the maximum path bandwidth value along the paths between the incoming interface out of which the RTN packet was sent and each receiver downstream of that incoming interface.

9.2 RTN Guaranteed Service feedback object

The RTN packet may optionally contain a Guaranteed Service feedback object comprising:

- **excess delay field** – the amount by which the installed end-to-end delay bound currently exceeds the target end-to-end delay bound.

- **bottleneck flag -** if set to 1 this indicates that the RTN message has travelled at least as far as the router where the accumulated delay-bound first exceeded the target delay-bound on the first pass of the RES message.

- **Sender Template** – same as that in RES packet whose end-to-end delay bound was exceeded.

At an outgoing interface, i of a router on the distribution tree, reception of an RTN packet from a next hop, j results in the updating of any matching router state, known as an RTN state entry or RTNSE_{ij}, or the setup of new state if no match exists. There will be a separate RTNSE_{ij} for each 4-tuple (Session, sender address, next hop, outgoing logical interface). The first three parameters of this 4-tuple are contained within the received RTN message while the outgoing logical interface(oif) is determined by the interface on which the RTN message arrived. In the case of a shared tree the sender address field will be omitted for the RTNSE_{ij}. The format of an RTNSE_{ij} is as shown in Table 2. In addition, for each outgoing logical interface, i a single Merged RTN State Entry (MRTNSE_i) is created from the set of entries $\{\text{RTNSE}_{ij}\}$ for that logical outgoing interface. There will be multiple RTNSE_{ij}s for a given logical outgoing interface if the logical outgoing interface has multiple next hops on the distribution tree which can occur if the logical outgoing interface connects to a shared medium LAN(e.g. Ethernet). The parameters of the MRTNSE_i and how they are formed from $\{\text{RTNSE}_{ij}\}$ are also shown in Table 2. The ‘merged values’ of various parameters in each RTN message sent out of an incoming interface to a previous hop upstream are obtained from $\{\text{MRTNSE}_i\}$, the set of MRTNSE for the outgoing interfaces as shown in Table 2.

$RTNSE_{ij}$	*$MRTNSE_i$*	*Merged RTN packet sent upstream out of interface k ($k \neq i$)*
CRT_{ij}	CRT_i=MAX{CRT_{ij}}	CRT_k=MAX{CRT_i}
$Ctot_{ij}$	$Ctot_i$=MAX{$Ctot_{ij}$}	$Ctot_k$=MAX{$Ctot_i$+$Clocal_{ki}$}(footnote 25)
$Dtot_{ij}$	$Dtot_i$=$Dtot_{ij}$ Where j is such that $TRID_i$=$TRID_{ij}$	$Dtot_k$=$Dtot_i$+$Dlocal_{ki}$ such that i gives MAX{$Dlocal_{ki}$+$TRID_i$} for that interface k (footnote 25)
$Propdelay_{ij}$	$Propdelay_i$=$propdelay_{ij}$ Where j is such that $TRID_i$=$TRID_{ij}$	$Propdelay_k$=$propdelay_i$+$dnext_i$ such that i gives MAX{$Dlocal_{ki}$+$TRID_i$} for that interface k (footnote 25)
$PathBandwidth_{ij}$	$PathBandwidth_i$= MAX{$PathBandwidth_{ij}$}	$PathBandwidth_k$= MAX{MIN($pathbandwidth_i$, link $rate_i$)}
$PathMTU_{ij}$	$PathMTU_i$= MIN{$pathMTU_{ij}$}	$PathMTU_k$= MIN{MIN(path MTU_i, $linkMTU_i$)}
$dnext_{ij}$	$dnext_i$=$dnext_{ij}$ where j is such that $TRID_i$=$TRID_{ij}$	
$TRID_{ij}$= $Dtot_{ij}$ +$propdelay_{ij}$ +$dnext_{ij}$	$TRID_i$=MAX{$TRID_{ij}$}	
sender $template_s$	*sender $template_s$*	*sender $template_s$*
$excessDelay_{sij}$	*$excessDelay_{si}$= MAX{$excessDelay_{sij}$}*	*$excessDelay_s$= MAX{$excessDelay_{si}$ –$delayReduction_{si}$}*
$bottleneckFlag_{sij}$	*$bottleneckFlag_{si}$= MAX{$bottleneckFlag_{sij}$}*	*bottleneck $flag_s$=MAX{bottleneck $flag_{si}$}*

Table 2: relationship between RTN state entries, MRTN state entries and merged RTN packets.

The last three rows of Table 2 represent optional GS-feedback objects and are written in italics to differentiate them from the core entries shown in the table. Merging between GS-feedback object state only occurs if the objects relate to the same sender template, s. A merged GS-feedback object for sender template, s is only included in the merged RTN packet sent upstream if the RTN packet is addressed to Phop for sender template s as obtained from installed RES state. With regard to the excess delay entries shown in the table, $delayReduction_{si}$ refers to the local reservation queuing delay reduction achieved since the RES for sender s at interface i was installed.

If CRTr is not equal to GS in the propagated RTN message, the rules in Table 2 are overridden by setting Ctot=Dtot=propdelay=0 in order to ensure that only those links

[25] $Clocal_{ki}$, $Dlocal_{ki}$ =router's value of C and D error terms between incoming interface k and outgoing interface i

receiving Guaranteed Service are taken into account when conducting worst-case merging of GS-specific parameters.

Whenever the contents of a RTN message to be sent upstream differ from the preceding one, the RTN message is sent immediately. Otherwise, i.e. in the steady state, an RTN message is sent to a previous hop once per some refresh period.

10 Summary

In this paper we have discussed the need for resource reservation in the Internet and examined the use of RSVP for this purpose. We have highlighted certain favourable characteristics of RSVP such as its use of 'soft-state' reservations. Consequently we acknowledge RSVP as a useful starting point in the design of alternative reservation protocols but we do not accept that it represents the ultimate solution because of certain deficiencies and restrictions that we demonstrated in the text. We also discussed ATM Block Transfer with Immediate Transmission(ABT/IT) drawing particular attention to its use of 'in-line' sender-initiated reservation requests which gives rise to very efficient bandwidth usage, even for connections where the short-term traffic characteristics of the connection are constantly varying. In this respect, ABT/IT represents a significant improvement over the low reservation control dynamics of RSVP.

Based on the strengths and weaknesses of RSVP and ABT/IT which we analysed in detail in the text, we proposed an alternative QoS signalling protocol called Dynamic Reservation Protocol which achieves the following main goals:

1) High reservation control dynamics to achieve efficient bandwidth usage.
2) Scalability of router-state with regard to number of senders and receivers. The protocol is especially suited to large-scale-multicast applications where it can expect to achieve a router state saving of several orders of magnitude compared to RSVP.
3) Heterogeneity of QoS classes and reservation styles for nodes within a given multicast session.

Details of control messages were presented along with associated processing rules. Although in principle DRP offers considerable benefits over existing reservation protocols certain aspects of it are not well understood and further work is required especially in the following areas:

1) Reservation setup time for each of the different service classes.
2) Impact of SSR mode on reservation set up time compared to SS mode.
3) Effect of worst-case merging of OPWA data – For a large multicast tree this will tend to cause the nodes closest to the sender to over-estimate the bandwidth required for their local Guaranteed Service reservation. However as the RES packet moves down the tree towards the receivers the potential for over-estimation at each node diminishes until at the last node the bandwidth yielded by (4) and (5) will be the exact amount required to make the accumulated delay equal to the target delay bound when the packet arrives at the receiver(s). Any implications of this phenomenom need to be clarified.
4) Investigation into alternative Guaranteed Service feedback techniques for the purpose of reducing the end-to-end delay bound when it is in excess of the target-delay bound after one pass of the RES packet. For example one alternative worth investigating is the generation of the RTN packet containing the Guaranteed Service feedback object as soon as the bottleneck node is encountered rather than waiting until the RES packet arrives at the receiver.

11 Acknowledgements

We would like to thank British Telecom Labs, Ipswich, England for supporting this work. In particular we are grateful to Alan O'Neill and Terry Hodgkinson of BT labs for many

valuable discussions regarding the material presented in this paper.

12 References

[1] W. Almesberger, J. Boudec and T. Ferrari. Scalable Resource Reservation for the Internet, EPFL DI-LRC, CH-105 Lausanne, Switzerland.

[2] ATM Forum (1996). ATM User Network Interface (UNI) Specification Version 4.0. AF-UNI-4.0.

[3] T. Billhartz, J. Cain, E. Farrey-Goudreau, D. Fieg and S. Batsell. Performance and Resource Cost Comparisons for the CBT and PIM Multicast Routing Protocols in DIS Environments, IEEE Journal of Selected Areas in Communications, April 1997. http://www.epm.ornl.gov/~sgb/pubs.html.

[4] R. Braden, L. Zhang, S. Berson, S. Herzog and S. Jamin. Resource Reservation Protocol (RSVP) - Version 1 Functional Specification, August 12, 1996.

[5]K. Carlberg, J. Crowcroft. Building Shared Trees Using a One-to-Many Joining Mechanism.

[6] L. Delgrossi and L. Berger. Internet Stream Protocol Version 2 (ST2) Protocol Specification - Version ST2+, August 1995, RFC1819.

[7] Forgie, J., "ST - A Proposed Internet Stream Protocol", IEN 119, M.I.T. Lincoln Laboratory, 7 September 1979.

[8]IETF, Integrated Services (intserv) Charter, http://www.ietf.org/html.charters/intserv-charter.html

[9] IETF, Integrated Services over Specific Link Layers (issl) Charter, http://www.ietf.org/html.charters/issll-charter.html.

[10] ITU-T (1995). Q.2931: Broadband Integrated Services Digital Network (B-ISDN); Digital Subscriber Signalling System No. 2 (DSS2); User-Network Interface (UNI) Layer 3 Specification for Basic Call/Connection Control.

[11] ITU Recommendation I.371. Traffic Control and Congestion Control in B-ISDN, (08/96).

[12]P. Newman et al. Ipsilon Flow Management Protocol specification for IPv4 Version 1.0, Internet Draft, February 1996, draft-rfced-info-flowman-00.txt.

[13]H. Saito and M. Kawarasaki. An Analyis of Statistical Multiplexing in an ATM Transport Network. IEEE Journal on Selected Areas in Communications, vol 9, no 3, April 1991.

[14] S. Schenker, C.Partridge, R.Guerin. Specification of Guaranteed Quality of Service, Request for Comments, September 1997, RFC2212.

[15]S. Seidensticker, W. Smith, M. Myjack. Scenarios and Appropriate Protocols for Distributed Interactive Simulation, Internet Draft, March 1997, draft-ietf-lsma-scenarios-01.txt.

[16] C. Topolcic. Experimental Internet Stream Protocol, Version 2 (ST-II), October 1990, RFC1190.

[17] P. White. RSVP and Integrated Services in the Internet: a tutorial, IEEE Communications magazine, May 1997, ftp://cs.ucl.ac.uk/darpa/rsvp2.ps

[18] J. Wroclawski. Specification of the Controlled-LoadNetwork Element Service,

Request for Comments, September 1997, RFC2211.

13 Appendix A: Determining link delay

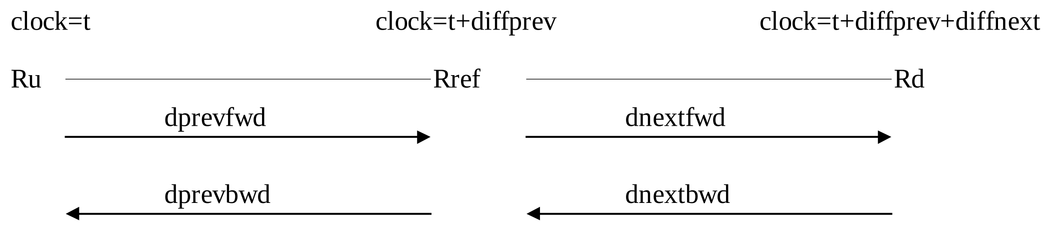


Figure 14: Reference model for determining link delay

Variable name	Description
Ru	**upstream router**
Rd	**downstream router**
Rref	**reference router**
Dprevfwd	**propagation delay of previous hop link(upstream) in forward direction**
Dprevbwd	**propagation delay of previous hop link(upstream) in backward direction**
Dnextfwd	**propagation delay of next hop link(downstream) in forward direction**
Dnextbwd	**propagation delay of next hop link(downstream) in backward direction**
Diffprev	**time by which clock of reference router is in advance of clock of upstream router**
Diffnext	**time by which clock of downstream router is in advance of clock of reference router**

Table 3: Description of variables

Using the following procedure a node is able to determine the propagation delay , dnext of a next hop link.

13.1 Upon receiving a RES packet from upstream

The node sets tarrival to its local clock. The timestamp value, tstamp is then

extracted from the RES packet. We then have the relationship

tarrival - tstamp = dprevfwd + diffprev = timedeltaprev. (8)

This value of timedeltaprev is now stored in the node.

Before forwarding the RES packet to the next hop(s) downstream the timestamp field is set to the node’s local clock.

13.2 Upon receiving a RTN packet from downstream

The node sets tarrival to its local clock. The timestamp value, tstamp is then extracted from the RES packet along with the value of the timedelta field. We then have the following relationships:

$$\text{timedelta} = \text{dnextfwd} + \text{diffnext} \qquad (9)$$

tarrival - tstamp = dnextbwd - diffnext (10)

Adding (9) and (10) gives

dnextfwd + dnextbwd = timedelta + tarrival - tstamp (11)

If it is known that the link delay is symmetrical then equation (11) simplifies to

dnextfwd = (timedelta + tarrival - tstamp)/2 (12)

Otherwise we have to assume the worst case for dnextfwd which gives
dnextfwd = timedelta + tarrival - tstamp (13)

Before sending the RTN packet to the next hop upstream the timestamp field is set to the node's local clock and the timedelta field is set to the stored value of timedeltaprev.