A Framework for Quality of Service Support of On-Demand Multicast Services in the Internet

Paul Patrick White

Submitted For Award of Doctor of Philosophy

Department of Computer Science

University College London

December 2000

Abstract

When the same information needs to be sent to many receivers, doing so using a single multicast transmission rather than a number of unicast transmissions offers several potential benefits, namely the consumption of less network bandwidth and server processing power while achieving lower transmission latency.

In this thesis we focus on the use of multicast for an 'on-demand' multicast delivery system whereby the time at which the sender transmits data to the group is influenced by the current demand, that is requests for the data made by receivers. Such an ‘on-demand’ multicast delivery system might be used for both real-time streaming(e.g. audio and video) and delivery of non-streamed information(e.g. web objects).

Our aim is to develop a framework that exploits the benefits of multicast as much as possible while satisfying the QoS requirements of each sender-receiver pair. In order to achieve these goals we present 3 complimentary components in the QoS framework of an on-demand multicast delivery system. Firstly we present 2 new methods of on-demand request scheduling, that is the server process of scheduling individual transmissions of requested resources as well as determining the delivery type of each transmission, either unicast or multicast. Secondly we present a resource reservation protocol that can be used to provide data flows with QoS commitments with regard to packet delay, loss rate and bandwidth by ensuring the availability of sufficient resources along the end-to-end path. Thirdly we present a new scheme for packet loss recovery in a multicast environment. For each of our three component contributions we analyse the relative merits compared to existing approaches and support our claims with simulation results.

Table of Contents

List of Figures

Acknowledgments

I would like to thank the whole of UCL CS department for providing an excellent environment in which to do my PhD work. In particular I would like to thank my supervisor Jon Crowcroft for his guidance and encouragement. Also a special thanks to Vicky Hardman and Graeme Knight for checking over my thesis. The input of John Andrews and Ken Carlberg also warrants a mention for being of great help to me during my time at UCL.

I am also very grateful to BT Labs, Ipswich for funding my work and allowing me to access some of their finest Internet gurus, in particular Alan O’Neill, Martin Tatham and Terry Hodgkinson.

1 Introduction

1.1 Background

The Internet is a rapidly expanding global communications network that promises to become, if not already, the ubiquitous communications medium of the Information age. In January of 2000 the Internet was estimated as connecting over 70 million end-hosts, an increase of 67% compared to the same time the previous year[64]. The infrastructure of this communications colossus comprises numerous physical and link-layer technologies that are logically interconnected using the Internet Protocol(IP)[1], the network layer protocol of the Internet.

IP, as originally conceived in the 1970s is a connectionless, packet-switching protocol which uses neither call admission control nor per-call state within the network. Each packet is simply routed to its destination hop-by-hop according to the globally unique IP destination address contained within the IP header. No deliberate service differentiation is made between different data flows at routers and available bandwidth is shared among all traffic. When the aggregate packet flow sent to a router output port is greater than the output link rate, packets must be queued in the output buffer which causes delay. Furthermore, in periods of congestion, the output buffer may overflow and packets will be lost. In addition, techniques employed in the network such as load-balancing may result in different packets of the same data flow traversing a different set of routers which can result in re-ordering of the packet stream as witnessed at the receiver. Even in the absence of load-balancing, packet re-ordering can still occur if the underlying network node routing tables and consequently the end-to-end path change midway through data transfer. Because of these indeterminate delivery characteristics, the service offered in the classical IP paradigm is termed 'best-effort'. On top of this core 'best-effort' service, end-to-end reliability and re-ordering can be achieved through appropriate transport layer protocols such as TCP(Transmission Control Protocol) which uses such techniques as positive acknowledgement and retransmissions which further adds to the overall information transfer delay.

The pooling of resources inherent in the traditional Internet philosophy is a key strength that ensures high utilisation of resources while overload results in graceful degradation of service rather than total denial. In addition, a connectionless model, whereby per-flow state is restricted to end-nodes, is very robust and scalable to networks of large sizes. Although the information transfer in the best-effort delivery model of IP is susceptible to variable queuing delays, and even packet loss

[1] The most popular version of IP in use in the Internet today is IPv4 although usage of IPv6 is now gaining momentum.

which can add to further delay if retransmission is required, this has not been a significant issue for traditional Internet traffic consisting of client-server transactions of non-real-time data. This is illustrated by the dramatic growth of the Internet since the introduction of the World-Wide-Web[2] in 1989 which unleashed the power of the Internet to the non-technical public.

However as we move further into the age of multimedia communications many real-time applications are being developed that are delay-sensitive to the point where the best-effort delivery model of IP can be inadequate even under modest network loads. Although the problem has been alleviated somewhat through making certain applications adaptive to network load where possible, there is still a firm need to provide many applications with additional service classes offering enhanced Quality of Service(QoS) with regard to one or more parameters such as bandwidth, packet queuing delay and loss. In addition, as the Internet is increasingly used by business, there is a growing number of users for whom the speed of overall information transfer is especially important. These changing requirements of applications and users have motivated research into the use of resource reservation in IP networks as a means of providing selected data flows with a degree of QoS commitment. With such an approach, resource reservation would be the exception rather the rule with the majority of traffic continuing to receive best-effort delivery. The use of appropriate service scheduling algorithms in routers which made any idle ‘reserved’ bandwidth instantaneously available to best effort traffic would then ensure that high utilisation within the network was maintained.

A major landmark in recent years in the evolution of the Internet is the introduction of multicast routing and forwarding techniques for group communication. Multicast techniques were motivated by the growing demand for group applications where the same information needs to be sent to multiple recipients. Although it is possible to accommodate this communication requirement using traditional unicast packet forwarding, such an approach is inefficient and does not scale with the number of receivers for the following three reasons[6]. Firstly, on any links which are common to the paths between more than one (sender, receiver) pair, the same information will be sent more than once. Secondly, the sender must keep state to handle the communication to each of the receivers. Thirdly, the time required for the sender to transmit the information to the full set of receivers will increase with the number of receivers due to the finite bandwidth available on the sender’s output link.

[2] WWW collectively describes the distributed information system that overlays the Internet and is realised using a combination of technologies including Hypertext Markup Language(HTML), Hypertext Transfer Protocol(HTTP), Domain Name Service(DNS) and hyperlinks.

Multicast communication overcomes these three deficiencies by setting up multicast forwarding state in the network to provide one or more multicast forwarding trees for delivery of information between group participants. Packets to be sent to receivers in the group are then addressed to a single multicast group address. In the absence of packet loss, each packet of information then only needs to be sent once and the multicast tree copies the packet at branching nodes in the tree to ensure that the packet eventually reaches all participants while traversing each link in the multicast tree only once[3]. The multicast forwarding state for a particular group is set up on demand in accordance with the underlying multicast routing protocols. In addition the appropriate multicast trees will be modified on demand each time a host joins or leaves the group.

At present most of the routers in the Internet are not multicast-capable and can only route according to IPv4 unicast addresses. To address this issue and allow graceful migration of the Internet from a unicast-only environment to one that includes full multicast-capability a virtual overlay network known as the Multicast Backbone(Mbone) has been introduced. The Mbone consists of islands of multicast-capable regions that are connected across the non-multicast capable remainder of the Internet using what are known as tunnels. At the tunnel entrance, that is a multicast-capable region's egress router, the multicast packet is placed in the payload of an IPv4 packet whose IP destination address is set to the router at the tunnel exit, that is the ingress router of another multicast-capable region. When a packet reaches a tunnel exit, the multicast packet is extracted from the payload of the encapsulating IPv4 unicast packet and is then routed according to its multicast destination address. In Feb 1997 the Mbone was estimated as containing 11,000 end nodes[3] while at the end of 1995 its rate of growth was a doubling of connected end hosts every 8 months.

1.2 Problem Statement

The use of multicast for many applications has attracted a lot of interest, in particular for real-time conferencing and the delivery of real-time data such as stock market quotes. With applications such as these the sender chooses when it will send the information to the group while receivers 'tuning' in to the delivery vehicle by joining the multicast group have no bearing on the time at which the information is sent.

Another kind of multicast delivery system that might be used is one that could be described as 'on-demand' in which case the time at which the sender transmits data to the group is influenced

[3] Strictly speaking there may be times where some multicast forwarding techniques cause the same packet to traverse the same link more than once, e.g. the flood phase of DVMRP[50]. In addition

by the current demand, that is requests for the data made by receivers. With an ‘on-demand’ multicast distribution system a server can aggregate requests over some time period before commencing service of each set of duplicate requests at the end of the time period using a single multicast transmission. In this way the server is able to reap the benefits of multicast for servicing multiple receivers as highlighted in section 1.1.

The focus of this thesis concerns the ‘on-demand’ multicast delivery system which we envisage might be used for both real-time streaming(e.g. audio and video) and delivery of non-streamed information(e.g. web objects).

In this thesis we extend the basic concept of on-demand multicast delivery so that in addition to current demand the latency requirements of the individual receivers themselves also influence the sender transmissions. Our aim is to develop a framework that exploits the benefits of multicast as much as possible within the constraints imposed by the QoS demands of each sender-receiver pair. We identify five QoS parameters that a receiver may wish to be fulfilled as follows:

1) Packet Delay

   In the case of streamed audio/video, a client may have a limit on acceptable packet delay imposed by the finite amount of buffering space it has available.

2) Packet Loss Rate

   In the case of streamed audio/video, client applications can tolerate a non-zero packet loss rate provided it is within certain bounds.

   In the case of delivery of web objects, client applications will usually need to experience zero information loss.

3) Bandwidth

   In the case of streamed audio/video, the bandwidth required in the network will be at least as great as the stream playout rate in order to ensure uninterrupted playout at the client.

   In the case of delivery of web objects, a minimum bandwidth may be required in order to achieve an acceptable download time.

4) Time to Start of Transmission

   This parameter is only likely to be relevant for delivery of streamed audio/video such as ‘movies on demand’.

5) Time to Reception of all of Requested Information

during the flood phase of DVMRP packets may reach receivers that did not request it. This inefficiency is eliminated in an explicit join multicast protocol such as CBT[7].

This parameter is only likely to be relevant for delivery of web objects.

In the above list of QoS parameters, 1) and 2) are closely related to 3) while 5) is closely related to both 3) and 4). In order to accommodate these QoS requirements we identify 3 complimentary components in the QoS framework of an on-demand multicast delivery system as follows:

1) On-Demand Service Scheduling.

This describes the server process of scheduling individual transmissions of requested resources as well as determining the delivery type of each transmission, either unicast or multicast. Although the preferred method of transmission is multicast, unicast service of a request may be necessary or preferred under certain circumsances; one example being whenever the waiting time to commencement of the next multicast transmission of the requested resource would result in violation of the requesting user's latency requirements (QoS parameters 4) and 5) above).

2) Resource Reservation and Allocation.

In order to provide QoS commitments with regard to packet delay, loss rate and bandwidth some means of ensuring the availability of sufficient resources along the end-to-end path is required.

3) Packet Loss Recovery.

As far as communicating applications are concerned, the three causes of packet loss can be listed as corruption during transmission, network errors and congestion causing packet buffer overflow at network nodes. Of these three causes, congestion usually dominates in an IP network.

Resource reservation for a specific data flow at a specific buffer can be used to eliminate congestion losses of the data flow at that buffer. However, a resource reservation that guarantees no packet queuing loss often consumes far more resources[4] than one that just provides an approximate minimum bandwidth while allowing occasional packet queuing loss. Furthermore it is unreasonable to assume that reservation-capable nodes will always be available along the full end-to-end path. Consequently resource reservation is neither an efficient nor guaranteed method of preventing congestion losses. Moreover packet loss due to data corruption or failures within the network are always possible regardless of whether or not resource reservation is being used.

Because network packet loss is unavoidable it is necessary to incorporate some method of packet loss recovery into our overall QoS framework.

[4] For example a Guaranteed Service[45] reservation which guarantees zero congestion packet loss will typically be allocated a significantly higher amount of service bandwidth and buffer space than a Controlled-Load[59] reservation which only offers an approximate minimum bandwidth while allowing occasional congestion losses.

1.3 Usage of Caching with On-Demand Multicast Services.

Since a common goal of both caching and multicast is to minimise network bandwidth consumption, it is appropriate to consider caching and how it fits in with an on-demand multicast delivery scheme. Caching involves storing popular web objects at various points between the origin server and the requesting client(s) in order to minimise the number of hops that a copy of the requested object must traverse in order to reach the client. In a basic implementation of caching, a cache might be present in the client itself and a proxy server for the client's subnetwork. In more advanced schemes such as [13] a hierarchy of caches may be used within the network. Even if caching is used the use of multicast in the delivery of on-demand services will still be beneficial in terms of reducing the bandwidth consumed in satisfying those requests that do result in transmission of the requested resource from the origin server. In such a scenario the use of on-demand multicast would not just be limited to the origin server but could also be implemented at cache proxy servers situated between the origin server and the clients.

1.4 Scope of Thesis

In summary, the scope of our thesis is a framework encompassing server scheduling mechanisms and network protocols that can be used to provide QoS for on-demand multicast services in a bandwidth-efficient manner.

Although we recognised in section 1.3 that caching may be a useful element of an on-demand multicast delivery system its investigation is outside the scope of our work.

The billing details of on-demand multicast services is also considered outside the scope of our work although we do make brief comments and observations on this topic as and when appropriate throughout the thesis.

1.5 Structure of Thesis

Chapter 2 provides a review of the state of the art and associated limitations with regard to the 3 components of our proposed QoS framework for support of an 'on-demand' multicast delivery system as identified in section 1.2. Chapters 3-5 present new approaches to the 3 components along with analysis and validation of our work. Finally in Chapter 6 we summarise our work, discuss further work, including billing, and present our conclusions.

1.6 Contributions of Thesis

Our thesis contributes new methods for each of the three components identified in section 1.2 as being required in a QoS framework to support on-demand multicast services. We now summarise our contributions.

1.6.1 New Methods of Service Scheduling for On-Demand Multicast Services.

In chapter 3 we present two new methods of service scheduling for on-demand multicast services.

Our first method, known as Asynchronous Multicast Push(AMP) with Classes of Service(CoS) is applicable to resources which, due to their short transmission duration, are unlikely to experience more than one multicast transmission at any one time. An example of this kind of resource would be a web object.

In AMP with CoS, each request from a client includes a latency class which reflects the requesting client's latency requirements. The server will consider the latency class of a request when determining the maximum holding time of each request, that is the maximum delay that may occur between the server receiving the request and passing the requested resource to the output buffer queue for transmission. If within a maximum waiting time determined by the class of a request, one or more duplicate requests arrive then the full set of duplicate requests is eligible for multicast delivery although such a potential multicast transmission will only actually take place provided a free multicast address is available considering the relative importance of all other potential multicast transmissions. Any request that has not undergone multicast transmission within the maximum waiting time determined by its class must undergo unicast transmission immediately in order to comply with the requesting client's latency requirements. Compared to a scheme that applies the same maximum holding time to all requests, AMP with CoS yields a superior resource saving in an environment with heterogeneous client latency requirements. A working prototype of AMP with CoS has been built which we evaluate through emulation using a request access pattern obtained from a WWW server.

Our second method, known as Optimised Batch Patching(OBP) with CoS is applicable to resources which, due to their long transmission duration, are likely to experience more than one multicast transmission at any one time. An example of this kind of resource is a movie. OBP with CoS facilitates two kinds of transmissions. Firstly, so-called regular transmissions which constitute a multicast of the entire movie and secondly so-called patches which may be multicast or unicast. Clients join a multicast group for either a new or existing regular transmission in order to receive all or part of the movie respectively. In cases where the joined regular channel is not new, the client will be missing the early part of the movie which can be recovered using a patch transmission.

OBP with CoS is novel in two ways.

Firstly requests are artificially delayed at the server in order to increase the probability of aggregating duplicate requests prior to their commencement of service. This facilitates delivery of a patch to multiple clients via a single multicast, an approach which is less-bandwidth hungry than allowing only unicast to be used to deliver patches. As for AMP with CoS the aggregation process considers the latency requirements of the client to determine the maximum holding time of a specific request.

Secondly, for the aggregation process just described, OBP with CoS determines the optimal point, in terms of bandwidth consumption, at which each new regular multicast should commence. We illustrate the bandwidth savings of OBP with CoS compared to a conventional patching approach using simulation.

1.6.2 A New Resource Reservation Protocol

In chapter 4 we present a dynamic sender-based reservation protocol that combines selective features of key state of the art reservation protocols together with several novel concepts to produce a new QoS signaling protocol known as Dynamic Reservation Protocol(DRP) which can be used to set up the IETF's Integrated Services models[29] 'on-the-fly' in a very bandwidth-efficient manner. We present details of packet formats and processing rules for DRP and discuss implementation issues. In addition, we compare the reservation bandwidth efficiency of DRP against selected state of the art alternatives for both single sender and shared multicast sessions using the ns simulator.

1.6.3 A New Method of Reliable Multicast

The bandwidth efficiency of multicast schemes that achieve reliability through retransmissions can be severely compromised if each retransmission is simply transmitted to the whole group as for the initial transmission. In a more optimal scheme, the retransmission of each lost packet is constrained to the minimum set of links necessary to deliver the lost packet to all receivers that are missing it. One way to achieve this is to build, on a per-lost packet basis, a retransmission tree that is a subset of the multicast tree. One approach for doing this is Pragmatic General Multicast, PGM[47] where the default retransmission tree is empty and any receiver requiring reception of a particular lost packet must explicitly join the retransmission tree for that lost packet. We call such an approach a 'retransmission join' approach.

Motivated by the fact that a 'retransmission join' approach is not the most efficient for small groups, in chapter 5 we present an alternative which we call a 'retransmission prune' approach. In a 'retransmission prune' approach the default retransmission tree is identical to the multicast tree and any receiver that does not wish to receive the retransmission of an advertised lost packet must explicitly prune itself from the retransmission tree for the lost packet.

We analyse the bandwidth efficiency of both retransmission tree building methods across different network topologies and group sizes. In the case of on-demand multicast services where the server has knowledge of the number of receivers prior to commencement of transmission the server can make use of our findings to determine the most efficient method, either a 'join' or 'prune' approach, for managing the retransmission trees for the transmission. The server can then convey its decision to the group receivers who can then register for any retransmissions in accordance with the favoured approach.

1.7 Publications

The work contained in this thesis encompasses the following publications which are referenced in section 7.

Paul White. RSVP and Integrated Services in the Internet: a Tutorial, IEEE Communications magazine, May 1997.

Paul White and Jon Crowcroft. Integrated Services in the Internet: State of the Art(extended version of publication #4), Proceedings of IEEE, December 1997, Volume 85, No. 12, pp 1934-1946.

Paul White. ATM Switching and IP Routing Integration: The next Stage in Internet Evolution? 2nd IEEE International Workshop on Broadband Switching Systems in Taipei, Taiwan, December 3 - 4, 1997. Also appears in April 1998 issue of the IEEE Communications magazine.

Paul White and Jon Crowcroft. WWW Multicast Delivery with Classes of Service. Fourth International Workshop on High Performance Protocol Architectures HIPPARCH '98, University College London, June 15-16 '98.

Paul White and Jon Crowcroft. A Dynamic Sender-Initiated Reservation Protocol for the Internet. 8th IFIP Conference on High Performance Networking (HPN'98) The Millennium Push of Internet, Vienna University of Technology, Vienna, Austria, September 21-25, 1998

Paul White and Jon Crowcroft. A Case for Dynamic Sender-Initiated Reservations in the Internet. Journal of High Speed Networks. Special issue on QoS Routing and Signaling. Vol 7 No 2, 1998. (Extended version of HPN '98 paper)

Paul White and Jon Crowcroft. A New Technique to Improve the Retransmission Efficiency of Reliable IP Multicast. First International Workshop on Networked Group Communication, Pisa 17-20 November 1999 (poster paper)

Paul White and Jon Crowcroft. Optimised Batch Patching with Classes of Service. ACM/CCR, October 2000.

2 State of the Art in QoS Support of On-Demand Multicast Services

2.1 Service Scheduling of Requests

The use of multicast for on-demand services has been proposed elsewhere in the literature. The extent to which such schemes are able to leverage the benefits of multicast depend upon the characteristics of the resource with regard to the probability of what we refer to as ‘transmission overlap’. Transmission overlap is where multiple transmissions of the same resource are active at a given moment in time and its probability of occurrence is determined by the relative values of mean transmission duration and time between consecutive transmissions. The higher the mean transmission duration compared to the mean time between consecutive transmissions the higher the probability of transmission overlap.

An example of a resource with a high probability of transmission overlap is video-on-demand since here the mean transmission duration of say 90 minutes will be significantly greater than the mean time between consecutive transmissions of the same video.

A resource with a low probability of transmission overlap is that of a web server hosting relatively small web documents since here each transmission is likely to have finished before the next one of the same document commences.

Broadly speaking service scheduling mechanisms can be categorised into those that exploit transmission overlap and those that don’t. The characteristics of the resource with regard to probability of transmission overlap will determine which category is appropriate. We now summarise some basic service scheduling mechanisms of both categories that appear in the literature.

2.1.1 Basic Schemes – No Exploitation of Transmission Overlap

[14] presents a technique whereby outstanding requests at the server are queued in a central queue prior to commencement of service. Associated with each such request is a list of clients awaiting delivery of the requested file. Files at the server are labeled as either hot(popular) or cold(not popular) according to their access history. When a request for a so-called hot-file arrives a check is made to see if it matches on any of the requests in the central queue. If no match occurs then the new request is added to the end of the central queue. If a match does occur then the client list associated with the matching request is updated to include the new requesting client, and no new entry in the central queue results. When each request reaches the head of the central queue the

request is scheduled for multicast delivery if it has more than one client in its associated client list. Otherwise the request is scheduled for unicast delivery.

[14] suggests that a portion of the multicast address space could be reserved for WWW multicast and each server could be allocated a contiguous block of addresses within this WWW multicast address space. [14] presents a many-to-one mapping of a server's files to the individual addresses within the server's allocated multicast address space. This is done using a hash function and has the advantage that clients with knowledge of the hash function know in advance which address multicast delivery of a given file would occur on. Upon sending a request to the server a client could then join the multicast group in readiness to receive the requested file via multicast should the server choose that mode of delivery. The disadvantage of the many-to-one file/address mapping function is that some clients may receive packets from files that they did not request which is obviously wasteful in terms of bandwidth consumption.

With the technique mentioned in [14], multicast delivery will only occur if the time between duplicate requests is less than the queuing time in the central queue. This queuing time and consequently the probability of multicast delivery will increase with overall server load. Consequently significant bandwidth savings will only occur when the server is significantly loaded.

The scheme in [39] which is referred to as Asynchronous Multicast Push(AMP) facilitates significant bandwidth savings over a wider range of server loads by explicitly specifying the request accumulation period rather than allowing it to be directly controlled by the volume of data queued for transmission. The request accumulation period is equivalent to what we call the maximum holding time(MHT) which is the maximum delay that may occur between the server receiving a request and passing the requested file to the output buffer queue for transmission. As far as the client is concerned the MHT is analogous to 'post-dial delay' in that the higher the MHT the higher the potential waiting time for the client before it receives the requested information.

As far as resource consumption is concerned, the higher the MHT the server is permitted to use, the greater the probability that a request matches on a duplicate before it is serviced. Since accumulated duplicate requests can be served via multicast to achieve a bandwidth saving compared to a unicast-only server, increasing the MHT will increase the bandwidth saving.

Conventional AMP[39] applies the same MHT to each request for a given file. Consequently in an environment with heterogeneity among the service latency requirements of clients, the MHT for a given file is limited by the most stringent of the clients' delay constraints. This means that the MHT applied to many of the requests would be lower than necessary to satisfy the latency requirements of the requesting client. Clearly this is sub-optimal in terms of potential bandwidth saving. A more efficient approach would support heterogeneity among the MHTs of different requests for the same file. In this case each individual request MHT would reflect the requesting client's latency requirements. For example, the MHT of a business user's request might be set lower than the MHT of a casual user's request.

In addition to request-initiated multicast of web pages, other techniques have been proposed that continuously multicast popular web pages at specific time intervals. This approach which [39] refers to as Continuous Multicast Push(CMP) enables best-effort UDP multicast to be used without any additional loss recovery mechanisms since any lost packets can be recovered automatically when the file is transmitted in the next cycle. AMP schemes on the other hand typically use some kind of reliable multicast protocol to handle lost packets. In addition, with CMP, clients that have advance knowledge of the multicast address that is being used can join the group without sending a request to the server. [2] and [39] present integrated server architectures that combine versions of AMP and CMP together with standard unicast delivery in order to provide each request with the most appropriate mode of delivery.

2.1.2 Basic Schemes – Exploitation of Transmission Overlap

2.1.2.1 Periodic Broadcast

In the schemes of [16][1][23][62] intended for video-on-demand, each video accommodated by the server receives dedicated usage of a number, K of equal bandwidth channels on the server output link. Each video file is subdivided into K increasingly bigger fragments each of which is periodically broadcast over a separate dedicated channel. To access a video a client simply tunes into channel 1 by joining the associated multicast group. Playout of the video at the client then commences upon the start of delivery of the next fragment on channel 1. While playing out the fragment on channel 1 the client tunes into channel 2 and begins storing the next full fragment of channel 2 so that upon completion of playout of the fragment from channel 1, playout can switch immediately to the stored channel 2 fragment without interruption. The client then tunes out of channel 1 and tunes into channel 3 to begin storing the next full channel 3 fragment. This procedure of playing out one fragment while downloading the next continues until the full video has been played out. It should be noted that in order to provide uninterrupted playout at the receiver, the playout duration of fragment x must be greater than the transmission duration of fragment x+1 for x in the range (1, K-1). Also since the size of fragment x is not greater[5] than that of fragment x+1 this means that the playout rate is less than the transmission rate. With periodic broadcast it is possible

[5] In fact it is smaller.

to achieve almost true video on demand by making the transmission period of fragment 1 reasonably small.

The technique of periodic broadcast is only suited to popular files. For less popular files periodic broadcast is unable to meet the dual goals of high bandwidth efficiency and low request latency. Also, even for popular files, periodic broadcast has one shortcoming in that it imposes a high client buffering requirement, typically equivalent to 70% of the video file size. This is a significant requirement and may preclude some clients from using the service.

2.1.2.2 Patching

This is a technique introduced by [24] which allows the benefits of multicast to be reaped for ‘near’ video-on-demand(VOD) and even ‘true’ VOD. With ‘true’ VOD, service of each request is commenced as soon as it is received by the server. By contrast with ‘near’ VOD there is a noticeable but user-acceptable delay between each reception of a request and commencement of its service. In [26] this delay is only ever caused by bandwidth limitations on the server output link although as we will explain later an alternative is for delay to be artificially imposed by the server scheduling algorithm in order to minimise output bandwidth consumption.

By the time service of a delayed request is about to commence the server may have accumulated additional requests for the same video. Each such group of identical requests is known as a batch. Following the service point of each request batch as determined by the server scheduling algorithm, the batch must then be served over 1 or 2 channels, that is multicast groups, either a so-called regular channel alone, or the combination of a regular channel and a so-called patching channel. A regular channel delivers a full film from start to finish while a patching channel delivers only the missing part of the film from the start until the point at which the clients of a batch joined the regular channel. In [26] the multicast groups were realised at the application layer in order to minimise congestion on the internal server bus. However the technique would also work using network layer multicast which would result in network bandwidth savings in a multicast-capable IP network.

The optimal method of patching known as grace patching works as follows. Upon receipt of a batch of identical requests the server identifies the newest regular channel that is serving the requested resource and equates the channel’s age to a corresponding buffer space, e.g. to store 10 minutes of Mpeg video requires approximately 100 Mbytes of buffer space. If the corresponding buffer space exceeds the buffer capabilities of the clients in the batch then patching cannot be used for the batch in which case a new regular channel must be opened instead and its identity conveyed to the clients in the batch.

Assuming the buffer space of the clients is sufficient then patching can be used as follows. The server opens a new channel known as a ‘patching’ channel which will deliver only the early part of the requested resource that the clients will miss upon joining the newest regular channel. The server then conveys the identity of both regular and patching channel to the batch of clients which then proceed to join both channels. Each client then buffers the regular channel and plays out the patching channel. When the patching channel has been exhausted each client then switches to playout of the buffered regular channel.

[26] compared the performance of a VOD multicast system using patching with one that always started a new transmission of the full video for each batch of requests, and found significant performance improvements in the patching case. For example, [26] found that compared to a no-patching approach, patching was able to support ‘true’ VOD at 4 times the request rate for a typical server configuration.

Compared to periodic broadcast we consider the patching technique to hold more promise in terms of service accessibility and generality. Service accessibility of patching will be higher since it imposes less of a buffering requirement on the client and is therefore less likely to preclude a client on the grounds of insufficient buffer space. In addition patching could be said to have higher generality than periodic broadcast since its usefulness is not restricted to very popular files as is the case with periodic broadcast.

Apart from the number of requests in each batch, with a patching scheme there is an additional factor that determines bandwidth efficiency, namely whether or not a batch with multiple requests is served via a new regular multicast or the combination of a patch and an existing regular multicast. The scheme presented in [24] bases this decision entirely on the buffering capabilities of the clients. In other words, to serve a batch with multiple requests [24] always uses a multicast patch provided each client has sufficient buffering capabilities. This strategy is not optimal with regard to bandwidth consumption. Intuitively we can see why by considering two cases, firstly where the most recent regular multicast has just begun and secondly where it is nearing completion. In the first case each batch would result in a short duration patch while in the second case each batch would result in a long duration patch which would be almost as large as the duration of the regular multicast. In other words the amount of traffic contained in a patch increases as the age of the most recent regular multicast increases. Once the regular multicast reaches a certain age it becomes more efficient to start a new regular multicast rather than to continue patching to the existing regular multicast.

[11] presents a proof of this for true video-on-demand patching and refers to the scheme as Optimized Patching. The time interval following a regular multicast during which it is more

efficient to patch to the existing multicast rather than begin a new regular multicast is known as the Patching Window. The Patching Window therefore defines the minimum time interval between successive regular multicasts of the same video.

In [11] service of any pending batch commenced as soon as a free channel became available on the server output link. Such a scheme is biased towards minimising latency to commencement of service which is obviously desirable from a user's point of view.

An alternative approach would be to minimise bandwidth consumption on the server output link which in turn would minimise network load and as such would be desirable from a network provider's point of view. Network bandwidth consumption can be minimised if the server delays commencement of service of a batch beyond the point at which a free channel first becomes available. In other words the server artificially delays service of each request in order to increase the number of requests contained in each batch and as a result the number of requests satisfied by any resultant patch. Such an approach would always be near VOD rather than true VOD but would be acceptable providing the user experienced a cost reduction compared to the true VOD case. In a 'simple batching process' applied to such a near VOD scheme the server might aggregate duplicate requests over a fixed batching interval before commencing service of the batch at the end of the interval. In a more optimal scheme the server might optimise batching size and hence bandwidth efficiency by considering the latency requirement of the client when determining how long it may artificially delay a request for. This is a similar argument to that presented in section 2.1.1 for the case of AMP.

2.2 Resource Reservation and Allocation.

2.2.1 Overview

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[6] in the nodes along the path will usually be required in order to identify the flows to receive special QoS as well as the QoS to be received.

[6] 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.

In order to allow end-hosts to invoke special QoS delivery on demand for a data flow in an IP network several protocols have been developed. In the next few sections we examine the most promising of these protocols. In addition we consider ABT/IT, which although designed for ATM networks, has features that could easily migrate to an IP network with appropriate protocol modifications.

2.2.2 ST group of protocols

The Stream Protocol, ST[21] is a connection-oriented network layer protocol introduced in 1979 in order to allow end-applications to signal their QoS requirements and trigger the setup of per-flow reservation and packet-handling state for their data flows. ST was intended to complement rather than replace IP, the connectionless network layer protocol of the Internet. It was envisaged that both of these network layer protocols might operate simultaneously in a multimedia environment in which case non-real-time applications would use IP to transport their data while real-time applications would use ST to leverage its QoS signaling capabilities. Another specification, ST-II[48] which was finalised in 1990 maintains much of the architecture and philosophy of ST but clarifies many areas left unaddressed in the original specification while making the protocol more generic and easier to implement.

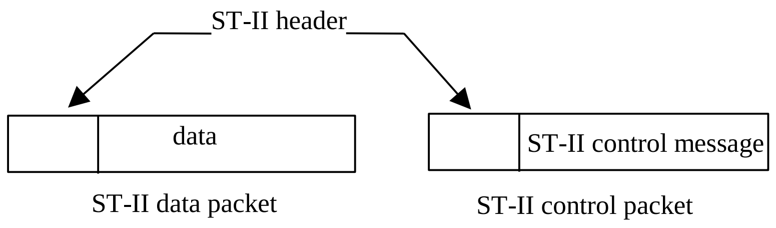


Figure 2.1: Types of ST-II packets.

As shown in Figure 2.1 each ST-II packet contains a common ST-II header which encapsulates either a higher layer PDU in the case of ST-II data packets or an ST-II control message in the case of ST-II control packets. ST-II control packets are used to set up and manage ‘streams’ for the transmission of ST-II data packets. The term stream collectively describes both the end-to-end path followed by transmitted ST-II data packets and the network state information and resources allocated for transmission of these data packets along the path. Each stream is identified by a globally unique name and has a single sender known as the ‘origin’ and one or more receivers known as ‘targets’.

Setup of a stream first requires the origin to determine the list of targets by some out-of-band means. The origin then transmits a ST-II CONNECT message to the set of next hops for this list of targets. The main components of a CONNECT message sent to a particular next hop are firstly, the subset of the initial target list reachable via that hop, secondly, a proposed ‘Hop Identifier’ for that particular hop and thirdly, a Flow Specification(flowspec) describing the desired QoS commitments of the stream. A Hop Identifier is simply a 16-bit value used to uniquely identify each hop of the stream’s tree. This allows forwarding of ST-II data packets down the stream’s tree to be done according to hop identifiers rather than IP addresses in order to achieve faster and more efficient data handling albeit at the expense of greater overhead at the stream setup phase. The flowspec includes desired and minimum acceptable packet size and rate in addition to accumulated delay and variance while the desired end-to-end delay and variance are implicitly taken as zero.

Upon reception of an ST-II CONNECT message, the main processing tasks carried out by an ST-capable node are firstly, the acknowledgement of the Hop Identifier chosen by the previous hop, secondly the attempt to install the reservation in accordance with the flowspec and thirdly for the reservation attempt, generation of either a REFUSE message sent to the previous hop or a CONNECT message with an updated flowspec sent to the next hop (see Figure 2.2) . A REFUSE message will be generated if the node has insufficient resources available to meet the minimum requirements of a CONNECT message. Reception of a REFUSE message is an indication to a node that the minimum acceptable QoS cannot be achieved via the attempted next-hop in which case the CONNECT should be resent to an alternative next-hop. Assuming a node’s admission control accepts the CONNECT message, the node updates the flowspec by adding its own estimated contribution to the end-to-end delay and delay variance fields before sending the CONNECT to the set of next hops downstream. In addition the node may reduce the desired rate and packet size provided they still remain above the minimum acceptable values contained in the appropriate fields of the flowspec.

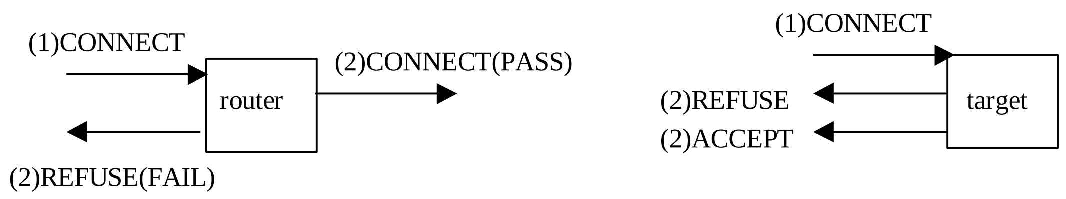


Figure 2.2: Node processing of ST-II CONNECT message.

When CONNECT packets finally reach the targets the flowspecs contained therein are inspected and each target then has the option of either reducing its end-to-end reservation or removing itself from the stream if it deems the accumulated end-to-end QoS unacceptable. Assuming the end-to-end QoS is acceptable, the target confirms its acceptance by including the final flowspec in an ACCEPT message which is then sent hop-by-hop towards the origin. Based on returned ACCEPT and REFUSE messages the origin can then decide whether or not it still wishes to proceed with the stream with the end-to-end QoS characteristics that were actually obtained. If it does wish the stream to stay in place then the origin may then send a CHANGE message with desired rate and packet size values below those in the original CONNECT message in order to reflect the actual end-to-end reserved values indicated in the flowspecs of received ACCEPT messages.

Once the stream has been set up, the origin can add or delete targets from its tree as well as send CHANGE messages with modified flowspecs whenever it wishes. Finally, once data transfer is complete, the origin uses ST-CONTROL messages to tear down the stream, that is remove from the network all forwarding state and reservations associated with the stream.

The ST2+ protocol specification published in 1995 revised the ST-II specification to improve interoperability between implementations and reflect experiences gained in their deployment while reducing protocol complexity where possible. In particular ST2+ does not use Hop Identifiers as these were considered to be unnecessarily complicated. Instead, ST2+ uses a globally unique stream identifier to identify each stream in the Internet.

In addition ST2+ clarifies the three different ways in which a stream can be built. Firstly, sender-initiated whereby the origin sets up a stream to a list of targets and may add additional targets to the stream afterwards. Secondly, mixed whereby the origin sets up a stream to a list of targets and additional targets may add themselves later. Thirdly, receiver-initiated whereby the origin sets up a stream without any targets although targets may add themselves to the stream afterwards.

ST2+ also mandates that the full specification is adopted by implementations unlike ST which allowed the option of subset implementations. In addition ST2+ removes several ST options whose added complexity was felt to outweigh any benefit. Those options removed include source-routing, reverse charge, point-to-point and full-duplex, the latter two having been intended to improve the efficiency of stream setup for two common communication scenarios.

ST2+ also contains some subtle changes and clarifications with regard to the router processing of CHANGE and ACCEPT messages. In both ST-II and ST2+ an intermediate node requires end-to-end confirmation of each CHANGE operation via a corresponding ACCEPT message from downstream. ST2+ specifies that any CHANGE message waiting to be implemented at a node outgoing interface must be queued and not implemented until ACCEPT messages for all previous CHANGE operations at the node interface have been received from downstream. ST-II on the other hand allows a CHANGE message to be implemented at a node interface even if a previous

CHANGE operation at the node interface has yet to receive a corresponding ACCEPT from downstream. The manner in which ST-II and ST2+ deal with a REFUSE response to a CHANGE is also different. In ST-II this condition is treated as a network failure and results in the downstream targets being dropped and a REFUSE message being sent to the previous hop upstream which might then try an alternate route to the targets. With ST2+ on the other hand reception of a REFUSE in response to a CHANGE simply causes the reservation to roll back to its pre-change request value.

2.2.2.1 Limitations of ST group of protocols

2.2.2.1.1 Protocol Overhead

With the ST protocols there is a substantial amount of overhead associated with both the setup of the stream and any subsequent sender-initiated reservation changes that may occur. The high overhead of these operations is due to the two levels of handshaking that occur for each CONNECT or CHANGE message sent. Firstly, the protocol includes hop-by-hop acknowledgement with retransmission of each message for reliability purposes. Secondly, upon sending a CONNECT or CHANGE message a node expects in return an ACCEPT or REFUSE message for each downstream receiver. An ACCEPT or REFUSE message is generated in the first instance by the receiver and is processed hop-by-hop on its journey towards the sender.

We argue that hop-by-hop acknowledgments are unnecessary in a QoS-aware domain, that is a domain supporting the reservation protocol, ST or otherwise, since reservation control packets can be assigned high priority by the individual routers and consequently should never experience congestion loss. At worst, the rare loss of a reservation packet due to data corruption or a network failure should only manifest itself to the end-user as a temporary service disruption since the next reservation packet will most likely restore full service[7]. The duration of any such service disruption can be minimised if the sender ensures transmission of a reservation packet at least once per some configurable time period regardless of whether or not the application's QoS requirements have changed since transmission of the previous reservation packet.

We argue that hop-by-hop acknowledgements are only necessary when the data flow transits a non-QoS aware domain. In this case the domain itself is regarded as a single hop as far as the acknowledgement is concerned and it should be a matter between the ingress and egress routers of the non-QoS-aware domain to implement the positive acknowledgement with retransmission to ensure reliable delivery of the reservation message across the QoS-less domain. Hop-by-hop

[7] If the reservation packet is for a reduction in an already established reservation then the loss of such a packet will not produce any user-perceived degradation in QoS.

positive acknowledgment with retransmission should therefore be treated as an extension to the core protocol to be used across QoS-less domains rather than a core feature of the protocol to be used in a ubiquitous manner as is the case with ST.

The protocol overhead of ST is even more apparent in the case of receiver-initiated joins in which case the above CONNECT-ACCEPT/REFUSE handshaking process is preceded by hop-by-hop processing of a JOIN message on the path between the new target and the existing stream.

2.2.2.1.2 All or Nothing Approach of ST

With ST2+ if a reservation change fails at a particular router then the CHANGE message is not propagated any further and each of the individual reservations between that router and the closest upstream branch point are modified back to their pre-change request values, i.e. we have an all-or-nothing approach. From the QoS viewpoint of the end-user this is not the best approach if the reservation change was for an increase in QoS. In this case the QoS shortfall of the end-user would be minimised if failure of an individual reservation change to meet the new higher value was not allowed to jeopardise the reservation changes of other individual nodes, both upstream and downstream of the failed reservation point.

2.2.2.1.3 Limited QoS Dynamics

In ST2+ whenever a reservation is altered due to processing of a CHANGE message the reservation is classed as being in CHGING state until an ACCEPT arrives from downstream. Any additional CHANGE received while in state CHGING is simply queued until the ACCEPT arrives. Upon arrival of the ACCEPT the state changes to ESTDL and the node is then allowed to process the next CHANGE message in the queue. Ignoring any processing delays, in the worst case a node will have to wait for one round trip time between entering a CHGING state and leaving that CHGING state. Consequently if the sender sends a number of CHANGES spaced in time by less than the round trip time then the CHANGES will get queued and the worst-case waiting time for a CHANGE at the back of the queue will be equal to the round-trip time multiplied by the number of CHANGES in front of it in the queue. Because of the limited QoS dynamics of ST2+, if the sender’s data stream characteristics are changing over time intervals less than the round trip time and a sender attempts to faithfully track the changing data stream characteristics with matching CHANGE messages, then when a queued reservation CHANGE is actually implemented in a node it will no longer be in synchronisation with the actual data flow.

Unlike ST2+, the ST-II specification permits nodes to process arriving CHANGE messages even if the node is still awaiting arrival of an ACCEPT message in response to an earlier CHANGE message. In principle this means that ST-II, unlike ST2+, is able to implement QoS changes over time intervals less than the round-trip time although doing so would represent a high degree of overhead due to the large number of handshaking messages(ACKs, ACCEPTs) that would be generated in response to the closely spaced CHANGE messages.

2.2.2.1.4 No Support for Heterogeneity of QoS Classes Among Receivers of a Session

Although the ST2+ specification allows flexibility with regard to the type of flowspec and consequently QoS class chosen by the sender it does not permit receivers to request a lower QoS class if desired. The end-to-end delivery service to all receivers will always be at the QoS class chosen by the sender.

2.2.2.1.5 No Intra-Session Control of Reservations by Receivers

With ST, upon receiving a CONNECT message a target has the option of reserving a lower amount of resources than the suggested amount in the flowspec. However once the receiver has been added to the stream it has no way of altering its reservation at a later time.

2.2.2.1.6 Receivers are Forced to Join a Stream in Order to Receive ST Packets

Any receiver wishing to obtain packets from a sender that is transmitting using ST can only do so if it joins the associated stream. Joining a stream involves a certain amount of overhead in terms of control messages transferred, network state installed and latency of the joining process. For some receivers the overhead of the stream setup might not be justified by the QoS guarantees offered by the ST stream compared to best-effort delivery. For example it may be the case that the paths to certain receivers are relatively congestion-free in which case best-effort delivery alone may be able to provide those receivers with sufficient QoS. In addition some receivers may just be ‘browsing’ the content delivered by various content-sources before actually deciding upon one to settle on. In both of these cases the initial overhead in subscribing to an information channel would be greatly reduced if the receiver was able to do so in a lightweight manner for best-effort delivery with the option of upgrading to higher QoS as and when appropriate and reverting to best-effort delivery as and when appropriate. ST unlike RSVP does not facilitate this. ST is a network layer replacement for IP on a per-communications session basis and in a session-wide manner. In other words with the ST model, the sender chooses to transmit using either ST or IP and once it has made this choice the delivery mechanism to all receivers is defined.

2.2.3 RSVP

In parallel with the work on ST2+, the IETF began working on another reservation protocol known as the Reservation Setup Protocol(RSVP)[10]. This has now reached full protocol status within the IETF and of all flow-specific reservation protocols developed to date for the Internet currently receives the most industry support.

Unlike ST, RSVP is designed to enhance rather than replace the underlying IP delivery service even on a per-session basis. RSVP is simply a reservation protocol that runs on top of IP without any network layer functionality of its own. Consequently in the RSVP model the joining of a receiver to a multicast communications session is a two stage process. Firstly the receiver joins the underlying multicast group and secondly the receiver may optionally request a resource reservation over the end-to-end path determined by the underlying multicast routing protocol. Moreover the receiver may initiate both teardown and re-installation of its reservation whenever it wishes.

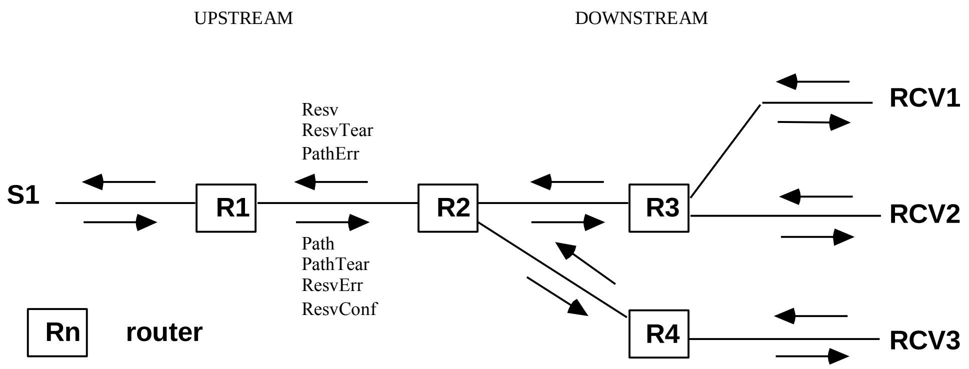


Figure 2.3: RSVP Messages in an Example Multicast Scenario.

Unlike the ST protocols, RSVP reservations are implemented as so-called soft-state which means they will time-out and be removed in the absence of any refresh reservation requests within a certain time period. This 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 significant reduction in complexity compared to a hard-state approach where such responsibility is assigned to the network instead. RSVP has other notable architectural differences compared to the ST protocols such as receiver-initiated rather than sender-initiated reservations[8]. 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.

As illustrated in Figure 2.3 the primary messages of RSVP are the Path message and the Resv message. The Path message travels downstream from sender(s) to receiver(s) of the communications session[9] but must have the IP router alert option set in the IP header in order to ensure it is processed by each RSVP-capable node that it encounters[33]. 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 even in asymmetrical routing environments.

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 on-tree routers to clip the requested reservation Tspec in certain circumstances[10].

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 assumption that only one of them is likely to be transmitting at once, a good example being an audio conference.

In the steady-state[11,] 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

---

[8] ST-II+ permits both sender and receiver-initiated reservations, ST-II and ST permit sender-initiated reservations only.
[9] The communications session is defined by the 3-tuple (IP destination address, transport layer protocol, transport layer destination port).
[10] 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 2.2.3.1.3.

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. Merging is performed according to the rules specified in the RSVP specification and 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.

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)[29] of the IETF has standardised two special QoS delivery models, both of which offer applications an end-to-end minimum bandwidth albeit with different assurances. Firstly, Guaranteed Service which offers applications an end-to-end delay bound and zero queuing loss. Secondly, 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[28].

2.2.3.1 Limitations of RSVP

2.2.3.1.1 Limited QoS Dynamics

Currently RSVP only allows receiver-initiated reservations and as a result is precluded from achieving optimal reservations in circumstances where the characteristics of the traffic stream change over short time intervals. With RSVP the sender could indicate any change in the traffic characteristics by sending a modified Path message which would trigger updated reservations to be generated upon arrival at the receiver. The total delay between the sender becoming aware of the QoS characteristics of the stream and the reservation change being implemented would be equal to one round trip time.

A good example of where the limited QoS dynamics of a receiver-initiated reservation mechanism such as RSVP might be exposed is that of a shared-reservation in an audio session for

[11] 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 state values 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, the new merged Resv message will be propagated upstream immediately.

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 2.2.3.1.3.

Another example is that of streamed audio/video where perhaps client buffering limitations impose such a limit on packet delay that packet loss recovery via retransmissions is not possible since any such retransmission would probably arrive at the receiver later than its playout point. In such a scenario the only means of ensuring a sufficiently low packet loss rate and consequently acceptable quality of the played out video at the clients would be to use resource reservation. Consider a video consisting of alternating periods of high and low on-screen action equating to high and low bit rate of the encoded video stream respectively[12]. Furthermore suppose that the duration of the low bit rate periods is substantially greater than that of the high bit rate periods which may typically be of a few seconds duration. In such a scenario, RSVP would be unable to respond fast enough to accurately track the instantaneous bit rate of the video. 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 the video stream would be over-specified.

2.2.3.1.2 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 on-tree routers and receivers. 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[13] associated with Path messages on each link of a multicast shared-tree[14] in the steady state will grow in proportion to the number of senders as shown in Figure 2.4a. If RSVP aggregation is used as shown in Figure 2.4b, then multiple Path messages within the core can be aggregated into a single Path message.

[12] One such example is a program that shows action video clips interspersed with discussion between a studio presenter and guests.

[13] 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.

[14] 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.

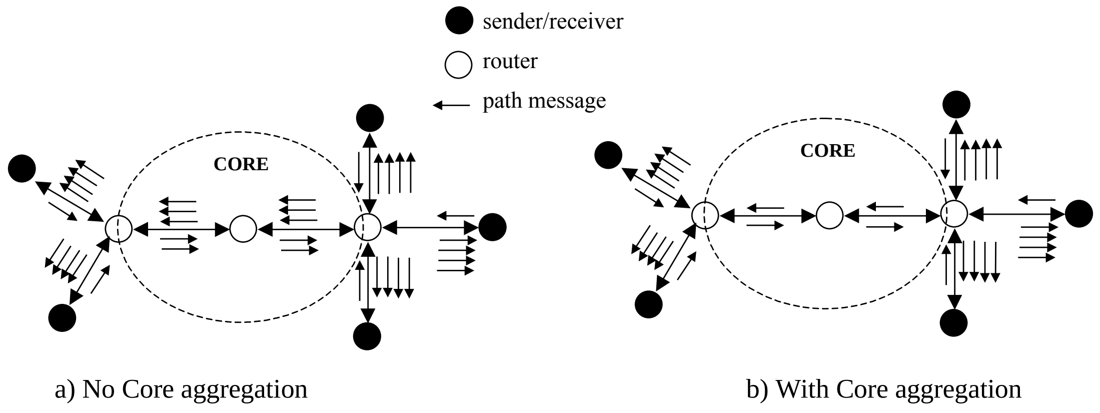


Figure 2.4: Number of RSVP Path Messages per Refresh Period in the Steady State on a Shared Multicast Tree With and Without Core-Aggregation

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.

2.2.3.1.3 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.

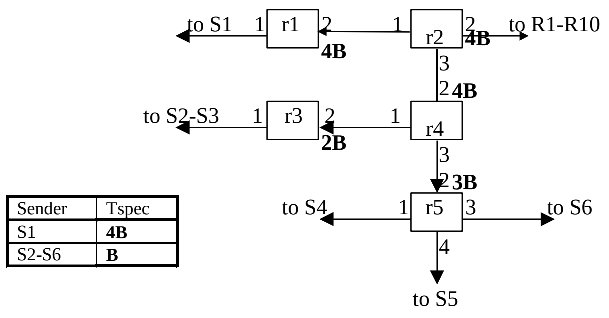
The Tspec contained in the Resv message arriving at an interface may exhibit static suboptimality for 2 reasons. Firstly, one or more receivers may have requested more than the maximum Tspec of all sources contained in the request's filter spec, in order to allow for a certain amount of overspeaking. 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 (2-1) where the sender Tspecs are obtained from Path state installed in the router.

$$Tspec = MIN(MAX\{\text{Sender Tspecs}\}, \text{Tspec of reservation request}) \qquad (2\text{-}1)$$

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

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

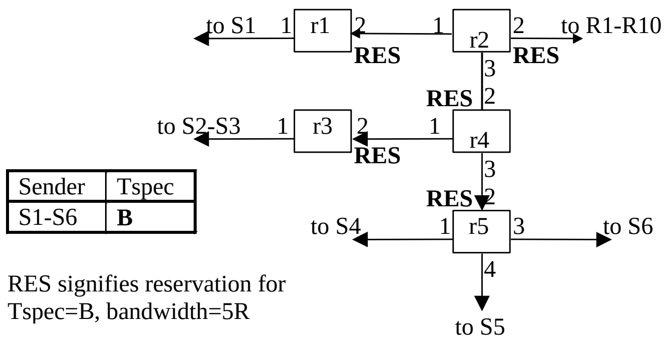
We now present an example in Figure 2.5 to illustrate static and dynamic suboptimality of reserved Tspecs. R1-R10 represent receivers of a multicast session that have requested shared-explicit[15] style reservations such that the union of the associated filter specs of these reservations gives the set 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.



Sender	Tspec
S1	**4B**
S2-S6	**B**

Figure 2.5: Example of Shared-Explicit, Controlled-Load Service Reservations at RSVP-Capable Routers

In the example of Figure 2.5, the Tspec of the installed reservation at r2 interface 2 will be 4B. This is an example of dynamic reservation suboptimality and when any 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 entry Tspecs will be equal to 5B(S2-S6) and the installed reservation Tspec will be equal to 4B in accordance with equation (2-2). 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 entry Tspecs 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.



Sender	Tspec
S1-S6	**B**

Figure 2.6: Example of Shared-Explicit, Guaranteed Service Reservations at RSVP-Capable Routers

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 the receivers' reservation requests. This is because Guaranteed Service reservation requests contain a service bandwidth to be reserved, Rspec in addition to a Tspec. The minimum required Rspec for each (sender-receiver) logical path that is to

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

receive a reservation is a function not only of the sender Tspec, but also the desired end-to-end delay bound and the path characteristics between the sender and receiver. As a result the required Rspec can vary among (sender, receiver) pairs of the sender. 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’ – reserved Rspec > minimum target Rspec of all sender-receiver pairs captured by the reservation. The target Rspec of a given sender-receiver pair is the minimum reserved Rspec required to meet the target end-to-end delay bound for that sender-receiver pair.

Dynamic ‘Rspec suboptimality’ – reserved Rspec > target Rspec of at least one of the sender-receiver pairs captured by the reservation.

While RSVP provides a mechanism, equation (2-2), for reducing static Tspec suboptimality, no such mechanism exists to reduce static Rspec suboptimality. This situation is illustrated in the example of Figure 2.6 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 packets from 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 next hop 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. The *Rspec, Rspecout* to be included in the reservation request would be obtained by taking the maximum of all bandwidth values contained in *listout*.

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.

2.2.3.1.4 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, perhaps because it might affect the service experienced by some of the receivers. For example assume a fixed-filter style reservation 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.

2.2.3.1.5 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. Perhaps 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.

2.2.3.1.6 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 session-wide changes in guaranteed service QoS 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. Firstly when the reservation reaches the sender.

Secondly, when the reservation fails admission control at any node. Thirdly, 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.

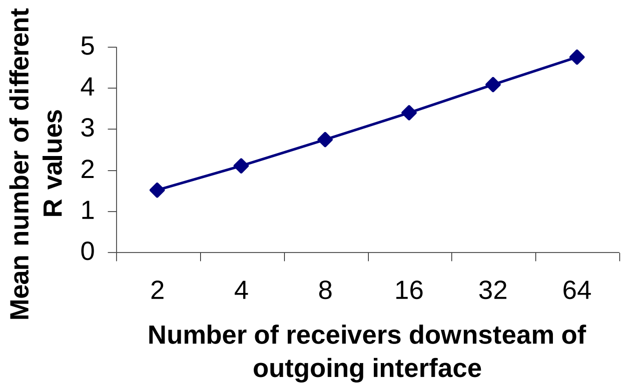


Figure 2.7: The mean number of different values of service bandwidth, R that a GS reservation at a node outgoing interface assumes before steady state is achieved.

In a multicast tree if we assume that the paths to each receiver have different characteristics then we may also assume that each receiver will calculate a different Rspec for its reservation flowspec in order to achieve its target end-to-end delay. In such a scenario Figure 2.7 shows the mean number of different values of service bandwidth a GS reservation at a node outgoing interface assumes before steady state is achieved. 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[16].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.

[16] 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.

2.2.4 ABT/IT

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 protocols such as ITU's[17] Q.2931 standard for public networks[31] or ATM Forum's UNI standards[6] for private networks use end-to-end handshaking to set up an end-to-end reservation before data transfer can take place. Moreover these traditional ATM signalling protocols do not permit in-call renegotiation of the end-to-end QoS. Consequently the negotiated reservation at connection setup would be determined by the most demanding anticipated requirements of the call and as such for a large proportion of the call the reservation may be instantaneously over-specified. A more dynamic and flexible approach is that provided by the ATM Block Transfer (ABT) signalling protocols which permit in-call QoS negotiation of a sender's traffic on a block-by-block basis. In addition, one variant of ABT actually allows renegotiation without end-to-end handshaking and as such is able to achieve an even greater match between the instantaneous service provided by the network and that required by the data flow.

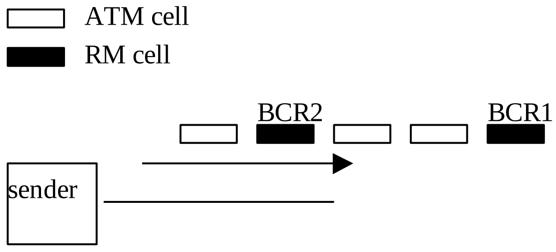


Figure 2.8: In-line reservations of ABT/IT using RM cells.

In the ABT 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 (Figure 2.8) 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 BCR and Sustainable Cell Rate(SCR). The maximum BCR represents the largest value of BCR that a resource management cell should contain. The SCR 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 connection’s 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 for 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.

2.2.4.1 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.

2.2.4.1.1 Connection Establishment Phase

[17] The International Telecommunications Union is an international organisation which among other things is responsible for the co-ordination of telecommunication standards.

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.

2.2.4.1.2 Limited QoS Control

With ABT, the QoS control is restrictive in 2 ways. Firstly the only class of service offered is an emulated DBR service. Secondly, 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

2.2.4.1.3 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[18] resources in accordance with the SCR traffic descriptor negotiated at connection-establishment even if the current BCR is zero

2.2.4.1.4 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.

[18] That is, consume resources as far as admission control of other connection requests is concerned.

2.2.4.1.5 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.

2.2.5 YESSIR

“Yet another Sender Session Internet Reservation” Protocol, (YESSIR)[40] was developed as a lightweight alternative to RSVP, for the benefit of real-time applications using the Real-time Transport Protocol(RTP). RTP is now the de-facto standard for providing end-to-end delivery services for real-time traffic. RTP operates on top of the User Datagram Protocol(UDP), a lightweight transport layer protocol whose only functionality is the provision of source/destination ports and an optional data checksum. RTP complements UDP to provide transmitted datagrams with additional transport layer services including sequence numbers, payload type identification and timestamping. A companion protocol, the Real-Time Control Protocol(RTCP) monitors QoS at receivers and delivers this information to all participants in a session. This is done via the periodic transmission by each host of either Reception Reports(RR) or Sender Reports(SR), the latter type of report being transmitted by those hosts that are also capable of transmitting RTP data packets.

As the name suggests, YESSIR reservations are sender-initiated. Furthermore they can be specified either explicitly in RTCP SRs or implicitly in either RTCP SRs or RTP data packets. The implicit reservation approaches require no extensions to end-host software beyond the capability to set the router alert option[33] in IP packets.

Two methods for the sender to implicitly specify a reservation exist with YESSIR. In the first, the sender simply sets the router alert in RTCP packets which are then intercepted by each on-tree router encountered on their journey down the distribution tree. Each YESSIR-capable intermediate router can then calculate the mean rate of the data flow and hence reservation bandwidth required by examining the RTCP header fields containing the timestamp and total number of bytes sent. The other implicit reservation approach that YESSIR suggests is intended to be used for payload types that have fairly predictable traffic characteristics. In this case the sender can periodically set the router alert option in RTP data packets in order to allow their interception by intermediate routers in the distribution tree which might then be able to calculate the reservation required by inspection of payload type field in the RTP header. For example, a payload type denoting G.711-encoded voice would be an indication to the intermediate node that a reservation bandwidth of 64kbps was required since this is the bit rate of this type of fixed rate data stream. One thing that is unclear about this second implicit approach suggested by YESSIR is how the routers determine which RTP profile to use for mapping payload type codes to payload types. RTP requires such profiles because of the limited size of the payload type field in the RTP header.

Explicit YESSIR reservation requests are realised by setting the router alert option in RTCP packets and extending the RTCP header to include a YESSIR reservation message comprising a generic fragment, a flowspec fragment, an optional reservation error fragment and optionally an Adspec as used in RSVP. YESSIR specifies 3 permissible formats for the flowspec. Firstly an IntServ flowspec specifying a Controlled-Load or Guaranteed Service reservation. Secondly, specification of payload type. Thirdly, a TOS(type of service) flowspec comprising the TOS[19] value contained in the IP header together with a bandwidth to be reserved for the flow. Payload type and TOS flowspecs permit aggregation of packet-handling state. For example each router could set up an output buffer per-media type or per-TOS value rather than per-flow. The router would of course still need to keep track of the contributions of each flow to the bandwidth requirement of each aggregated output buffer although data packet handling itself would be per media type or per TOS value.

Like RSVP, YESSIR permits shared reservations and uses soft-state to implement all reservations. Unlike RSVP, which does not forward reservation requests if they fail, YESSIR does if operating in partial reservation mode. In this case the reservation message after the failure point is still capable of installing reservations. Then over time it would be likely that the number of links without reservations along the end-to-end path would reduce and eventually vanish as future periodic YESSIR reservations possibly succeeded at points where previously they had failed.

2.2.5.1 Limitations of YESSIR

2.2.5.1.1 Transport Layer Dependency

In the YESSIR specification, as it stands, Explicit YESSIR reservations can only be carried within RTCP SR's which are transmitted periodically in accordance with the RTP specification.
In other words, explicit YESSIR reservatons mandate the usage of RTP while the end applications have no direct asynchronous control over the transmission of the reservations. Reservations will simply be transmitted once every RTCP SR transmission period which is determined by RTCP according to the number of session participants. While these restrictions are fine for real-time

[19] The TOS field in the IP header of a packet allows a preference to be expressed with regards to one of the following parameters: delay, throughput, loss. In addition it allows a precedence(priority) value to be assigned to each packet.

transmissions of moderate to long duration with non-varying session QoS requirements they impose restrictions in the following 3 cases:

1) Real-time transmissions of short duration.

The lack of end-application control of the timing of the transmission of RTCP SR's may result in the transmission of a large part, if not all, of the sender's data before the reservation is sent.

2) Real-time transmissions with variable session QoS requirements.

The data streams of some real-time applications may have QoS requirements that vary on a block-by-block basis. In these cases, for optimal bandwidth efficiency, the QoS delivered by the network should be changed on a block-by-block basis to match the QoS required by each block. Accurate block-by-block control of network-delivered QoS is not possible in the absence of asynchronous control of explicit reservation transmissions by end-applications.

In cases where the maximum QoS requirement of all blocks over the next inter-SR transmission period is known immediately prior to the sending of an SR the reservation could be set to represent this maximum requirement. However this is likely to represent suboptimal bandwidth usage since for many of the blocks in the inter-SR period the reservation may represent an over-specification.

In cases where the maximum QoS requirement over the next inter-SR transmission period is not known immediately prior to the sending of an SR the sending application could resort to an estimation which for some of the blocks in the inter-SR transmission period might represent an under or over-reservation.

3) Non-real time transmissions of any duration.

Very few, if any, non-real time applications are likely to derive any benefit from RTP and so are unlikely to use it as a matter of course. Without using RTP such applications would be unable to initiate reservations according to the YESSIR protocol. Although it is possible that sender applications could send dummy RTCP SR's whose only useful purpose was as a carrier of YESSIR reservation requests such an approach would be inefficient since a large proportion of the RTCP SR would contain information that was unused.

2.2.5.1.2 Over-Reservation of Guaranteed Service in Multicast Sessions

For an end-to-end Guaranteed Service reservation with a uniform reservation/hop profile(that is the same bandwidth reserved in each router along the end to end path), the reservation bandwidth required for a specific (sender, receiver) pair of the multicast session is determined by the following 3 parameters:

1) Sender Tspec

2) End to end delay requirements
3) End to end path characteristics

For a given sender 1) and usually 2) above will be the same for each receiver while 3) will vary among receivers.

YESSIR gathers the end-to-end path characteristics in an Adspec carried in RTCP SR’s. Upon delivery of an RTCP SR to a specific receiver, the Adspec contained therein represents the end to end path characteristics between the sender and that specific receiver. The receiver makes this information available to the sender by including it in a RTCP RR(Receiver Report) that it later sends. Upon reception of a RTCP RR from a specific receiver, i, the sender can then calculate the minimum value of reservation bandwidth, Ri necessary to satisfy the end-to-end delay bound to receiver, i. The actual value of reservation bandwidth, Rres specified in the next SR sent by the sender is then given by maximum value in {Ri}. Because the same maximum value of Ri will be installed in all on-tree outgoing interfaces, at many of them it will represent an over-reservation. In large multicast sessions or those with a high level of heterogeneity among the end-to-end path characteristics for each (sender, receiver) pair the level of over-reservation can be expected to be significant.

2.2.6 SRP

Unlike the reservation protocols described above, the reservation protocol known as Scalable Reservation Protocol(SRP)[4]does not trigger the set up of any per-flow reservation state in routers. Instead any reservation in a router that is set up using this protocol applies to the aggregate of all reserved traffic passing through a specific output port. Any per-flow state is restricted to end-nodes.
With SRP, any data packet will be one of three types, namely ‘reserved’, ‘request’ and best-effort, the latter receiving no bandwidth reservation at all. For each packet that is part of a flow requiring a QoS commitment, the sender will mark its type as reserved if sending it will not exceed the current end-to-end reserved bandwidth according to certain calculations which we describe shortly. Otherwise the packet’s type is marked as ‘request’.
The scheduler at each node’s output port uses two queues, reserved and best-effort with a proportion of the available bandwidth assigned to each. Upon arrival of a packet of type ‘reserved’ at a node output port the packet is simply placed into the reserved queue for the output port. By contrast, when a packet of type ‘request’ arrives at a node output, the proportion of available bandwidth assigned to the reserved queue is increased to reflect arrival of the request packet before placing it in the reserved queue.

Central to the scheme is the use of bandwidth estimators which examine the number of reserved and request packets over a specific observation interval and feed the information into an algorithm in order to estimate the amount of reservation bandwidth currently needed for the data flow. Bandwidth estimators are used in 3 places, namely at senders, routers and receivers. At the sender, bandwidth estimators provide an optimistic estimate of the end-to-end reserved bandwidth enjoyed by a specific data flow. At routers bandwidth estimators predict the aggregate amount of reserved bandwidth needed at each output port. The use of such estimators in routers allows the reserved bandwidth to be automatically reduced should the sending sources go idle or reduce their sending rate. At receivers, for each specific data flow a bandwidth estimator provides a conservative estimate of the bandwidth reserved at the last hop router. This information is then fed back to the sender which then takes the minimum of this and its own optimistic estimate to determine the threshold between labeling packets of that flow as either request or reserved. Because the reservations are per output port rather than per-flow this mechanism is scalable to large networks carrying a large number of flows. In such a scenario per-flow policing would still be required at network access points to ensure that end hosts did not exceed their indicated traffic levels.

2.2.6.1 Limitations of SRP

Although SRP has excellent scalability characteristics it is only able to offer end applications an approximate minimum bandwidth without any quantitative guarantees on loss or delay. As such it may be inadequate for scenarios with stringent QoS requirements such as commercial-grade video conferencing.

2.3 Packet Loss Recovery

The Transmission Control Protocol(TCP) is a very effective protocol for reliable transmission in the Internet which it implements through the use of positive acknowledgement and retransmission. Each time the sender transmits a packet it starts a retransmission timer for the packet. If the packet remains unacknowledged upon expiry of the retransmission timer then the sender deems the packet to have been lost and consequently retransmits it. As part of this mechanism, TCP continually updates an estimate of round-trip time(RTT) in order to set the retransmission timer to a suitable value, B*RTT where B>1 (typically 2). TCP also implements flow control through the notion of a sliding window which effectively determines the maximum transmission rate. The sliding window size and consequently maximum transmission rate can be controlled both by the receiver(end-to-end flow control) and the sender(congestion control). A key strength of TCP is its robustness in that it makes no assumptions about the error recovery capabilities of the underlying link-layers, operating instead purely on an end-to-end basis and requiring no cooperation from intermediate nodes.

While these mechanisms of TCP work fine in the case of unicast they have several shortcomings if applied directly to multicast[20]. Firstly they do not scale to large multicast sessions due to acknowledgement implosion at the sender. Secondly, they require the sender to have knowledge of, and track changes in, the receiver set. Although such information may be available for on-demand multicast servers where the ‘demand’ is indicated by recording the number of requests received at the server over some time period, it may not be available for those cases where the demand is indicated by some out-of-band means. For example, in the case of a live sporting event, the demand may be indicated by past experience as being very large in which case the server is aware in advance that the transmission is suitable for multicast and would therefore not bother monitoring the number of individual requests from end-users. Thirdly, parameters maintained by the sender such as congestion window and round-trip time are related to the path to a specific receiver and so cannot easily be mapped to a multicast scenario where there may be many different (sender, receiver) pairs, the path characteristics of which may vary by several orders of magnitude. These issues suggest that sender-based control of reliability is not the most appropriate method for a multicast scenario and that it is better to assign the responsibility for ensuring reliability to receivers instead[20].

2.3.1 SRM

One reliable multicast protocol which assigns the responsibility of reliability to the receivers rather than the senders is known as Scalable Reliable Multicast(SRM). Here any end host detecting packet loss sets a repair request timer to a random value influenced by its distance from the sender so that the smaller the distance the smaller the timeout value. If a duplicate request from another end-host is detected before expiration of the timer then the request is canceled in order to minimise the risk of duplicate requests which would obviously be wasteful in terms of bandwidth. If no such duplicates are detected then the request will be multicast upon expiry of the request timer. Any end-host is able to reply to the request subject to similar suppression algorithms applied to the reply.

2.3.1.1 Limitations of SRM

A deficiency of the basic SRM mechanism together with many other reliable multicast techniques that achieve reliability through retransmissions is that any retransmission is distributed to the full multicast group including those receivers that do not require the retransmission. This is wasteful in terms of bandwidth and as such compromises one of the major potential benefits of multicast transmission as highlighted in section 1.1.

2.3.2 PGM

In a bandwidth optimal reliable multicast retransmission scheme, the retransmission of each lost packet should be constrained to the minimum set of links necessary to deliver the lost packet only to the subset of receivers that are missing it. One reliable multicast protocol that achieves such optimal retransmissions is Pragmatic General Multicast, PGM[47] which builds, on a per-lost packet basis, a retransmission tree that is a subset of the multicast tree and is realised by the combination of the underlying multicast tree forwarding state and additional network state(retransmission state) installed by the reliable multicast protocol. With PGM, the default retransmission tree is empty and any receiver requiring reception of a particular lost packet must explicitly join the retransmission tree for that lost packet. Joining the retransmission tree results in the installation of retransmission state indicating the subset of on-tree outgoing interfaces that are to forward the retransmitted packet. In conventional multicast, packet forwarding is done entirely according to underlying multicast tree forwarding state. Consequently in order to constrain retransmissions to the set of links defining the retransmission tree some method is required to instruct routers to also consider any installed matching[20] retransmission state during the forwarding of each retransmitted packet. PGM facilitates this requirement by setting the router alert option in retransmitted packets in order to indicate to routers that the packet requires additional higher layer processing. Inspection of higher layer fields in the packet would then reveal to the router that the packet was a retransmission and so packet forwarding should be constrained to the subset of on-tree outgoing interfaces that form the retransmission tree. After this packet forwarding process the router would then remove any matching[20] retransmission state.

2.3.2.1 Limitations of PGM

PGM's join approach to retransmission tree building is the only one it accommodates. An alternative approach to retransmission tree building is what we refer to as a 'retransmission prune' approach, whereby the default retransmission tree is identical to the multicast tree and any receiver that does not wish to receive the retransmission of an advertised lost packet must explicitly prune itself from the retransmission tree for the lost packet. With a 'retransmission prune' approach retransmission state indicates the subset of outgoing interfaces that the retransmitted packet is NOT to be forwarded out of.

[20] Matching in terms of (source, group, packet sequence number).

When the number of receiver LANs is small a ‘retransmission prune’ approach produces less retransmission state than a ‘retransmission join’ approach.

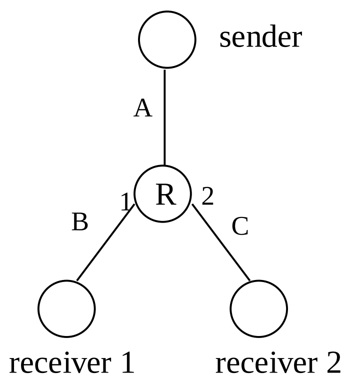


Figure 2.9: Example of a Distribution Tree for a Small Multicast Group

As a very simple example of this consider Figure 2.9 which shows a small multicast group involving a single sender and two receivers with a single network router R contained with the multicast tree.

In the case of a prune retransmission approach a packet lost on link A will not generate any prune state since in this case the retransmission needs to be delivered to both receivers Assuming a packet is lost on link B then a single instance of prune state is required at R interface 2 in order to ensure that the retransmission is delivered only to receiver 1. Similarly if a packet is lost on link C then a single instance of prune state is required at R interface 1. Assuming that packet loss is equiprobable on each link this means that on average a single packet loss will generate 2/3 instances of prune state in network routers.

In the case of a join approach a packet lost on link A will generate 2 instances of join state in router R, one at interface 1 and one at interface 2. A packet lost on link C will generate a single instance of join state in R at interface 2. Similarly a packet lost on link B will generate a single instance of join state in R at interface 1. Assuming that packet loss is equiprobable on each link this means that on average a single packet loss will generate 4/3 instances of join state in network routers, i.e. twice the amount of retransmission state that would be generated using a prune approach.

As we shall see in chapter 5, control message overhead will also typically be less for a retransmission prune approach than a join approach if the number of receiver LANs is small.

If knowledge of the receiver set is unavailable or if the receiver set may vary during the course of transmission then a ‘retransmission join’ approach is preferable as it has better scalability characteristics.

On the other hand if the receiver set is known and fixed throughout the course of the session then a more optimal approach would be to choose between a ‘retransmission prune’ and a ‘retransmission join’ approach on a per session basis according to the size of the receiver set. However PGM does not do this and is therefore suboptimal in this respect.

2.3.3 Forward Error Correction(FEC)

Forward Error Correction at the packet level allows the information in packets lost through congestion to be recovered at the receiver without requiring retransmission by the sender. In such schemes redundancy between packets transmitted is used such that the receiver is able to recover the information in all k packets transmitted from only j packets received where $0 < j_{min} < j < k$.

2.3.3.1 Limitations of FEC

We view the use of any packet level FEC scheme as an application layer decision and even if such a scheme is used it cannot guarantee full reliable delivery for non-cyclic transmissions. Consequently within our overall framework we deem there to be a requirement for the support of packet loss recovery through retransmissions.

3 New Methods of Service Scheduling for On-Demand Multicast Services

In this section we present two new methods for service scheduling of on-demand multicast servers. The suitability of each method to a particular scenario depends on the probability of 'transmission overlap', that is time overlap between different transmissions of the same resource as discussed in section 2.1.

Our first method is applicable to on-demand multicast servers where the probability of transmission overlap is small, e.g. distribution of web documents. Here we address the suboptimality of AMP discussed in section 2.1.1 by extending the basic principles of AMP so that the maximum holding time of a request at the server is determined by a request class which reflects the requesting client's latency requirements. We refer to our new scheme as AMP with Classes of Service(AMP with CoS) and present emulation results of the bandwidth saving achievable in an environment with heterogeneous client latency requirements compared to conventional classless AMP.

Our second method is applicable to on-demand multicast servers where the probability of transmission overlap is high, e.g. video on demand. Here our scheme is based on the principles of Optimised Patching, which were discussed in section 2.1.2.2, but includes AMP with CoS batching of requests in order to exploit the benefits of multicast as much as possible and achieve a very bandwidth efficient scheduling scheme for near video-on-demand. We call our scheme Optimised Batch Patching with Classes of Service(OBP with CoS). Compared to Optimised Patching, OBP with CoS consumes less network bandwidth at the expense of higher latency until commencement of service as perceived by the end user. We examine the behaviour of our scheme using analysis and simulation. In addition we discuss various implementation issues including charging models and system dimensioning.

3.1 Asynchronous Multicast Push(AMP) with Classes of Service

In Asynchronous Multicast Push with Classes of Service, AMP with CoS, the server divides time into epochs of duration *T*. Each client request for a file has an associated Client Designated Class(*CDC*) which is an integer indicating the maximum 'decision time' measured in epochs that the client is willing to let the server apply to the request. We define decision time as the delay between the server receiving a request and making the 'final service decision', that is determining its method of service, unicast or multicast. In the case of a unicast 'final service decision' the requested file is sent to the output buffer for transmission immediately after making the decision. In the case of a multicast final service decision the requested file is sent to the output buffer for transmission k epochs(k<1) after making the decision. Each file at the server has an associated Server Designated Class(*SDC*) which is configured a priori to impose an upper limit on the maximum decision time for any request for the file. The SDC provides for files of a real-time nature that may need to be sent to the requesting client within certain time constraints in order to remain useful. Files with no specific real-time characteristics would simply have their SDC set to the default of ∞. When a request arrives at the server, the *SDC* of the requested file is combined with the *CDC* of the request in order to produce an "effective class"(*c*) for the request according to equation (3-1).

$$c = MIN(SDC, CDC) \tag{3-1}$$

The effective class of a request reflects the client's general latency requirements together with any specific latency requirements of the requested file, and equals the maximum delay in epochs between the server receiving a request and making the final service decision for the request. For the remainder of this chapter we use the term 'class' to mean 'effective class'. Although in principle the scheme is extendable to any number of classes, in our analysis we assume just 5 classes, 0 to 4. Files with *SDC*>0 are known as multicast pool files and each request for such a file is eligible for multicast delivery provided both *CDC*>0 and the condition in equation (3-2) is satisfied.

$$cR_i \geq M \tag{3-2}$$

where R_i is the weighted average number of requests per epoch for each file, i, and M is the multicast eligibility threshold. Any requested file not eligible for multicast is unicast immediately over the existing TCP connection that was used to carry the request. The set $\{R_i\}$ is updated once every analysis period of *Ea* epochs(Ea > 1) according to equation (3-3).

$$R_i = \alpha R_i + \frac{(1-\alpha)\, R_{sample}^{i}}{E_a} \tag{3-3}$$

where R_{sample}^{i} is the number of requests received for file *i* over the previous *Ea* epochs, and $\propto$ is a value between 0 and 1 which controls how fast the weighted average responds to changes in the request arrival rate(a lower value of alpha causes it to respond faster). Implementation of equation (3-2) ensures that those requests for files that are not popular and which therefore have little chance of matching on duplicate requests were they to be held at the server are served immediately via unicast. We refer to those requests whose mode of delivery has yet to be decided as outstanding requests. At each epoch boundary the server applies a service-allocation procedure to the group of outstanding requests to decide how each one is to be handled. During this procedure, the server determines which files with outstanding requests are to undergo multicast transmission commencing in the next epoch. We call these files ‘multicast-active’ files and in the next section explain the full procedure for determining them. Once the set of ‘multicast-active files’ has been selected the service-allocation procedure determines the handling of each outstanding request as shown in Figure 3.1.

```
if request is for a multicast-active file
        ⇒ service via multicast transmission commencing in the next epoch.
else if class of the requested file = 1
        ⇒ service via unicast over existing TCP connection ASAP
else
        ⇒ request class, c = c – 1
           hold request until next epoch boundary
```

Figure 3.1: Summary of Service Allocation Procedure for AMP with CoS

3.1.1 Detailed Service Allocation Procedure for AMP with CoS

As in [14] we assume that a portion of the multicast address space is reserved for WWW multicast and that each server using our scheme is allocated a contiguous block of addresses within this WWW multicast address space. However, the mapping between a server’s files and its available multicast addresses differs significantly between our scheme and that in [14] where a static many-to-one mapping is used. By contrast, in our scheme the mapping which will be explained later in this section is one-to-one and dynamic. We assume that only N multicast addresses are available for use by the server while the number of multicast pool files is equal to F where $F>N$. Consequently even if duplicate requests are outstanding at an epoch boundary at which their final service decisions must be made it may not always be possible to assign them to multicast delivery since all of the available multicast addresses may be in use for the delivery of other files. We assume a limit on the number of multicast addresses used by each server to ensure that the technique will scale to a large number of servers without depletion of the available multicast address space[21].

We now describe in detail the service allocation procedure that AMP with COS applies at each epoch boundary to the group of outstanding requests which are recorded in the set of hashtables $\{hashCount_c\}$ where c = 1,2,3,4 denotes request class. Before the server begins its

[21] This is much less of an issue in the case of the recently proposed source-specific multicast approach[30] whereby forwarding is done according to the combination of source address and multicast destination address. In such a scheme each source has the full 232.0.0.0/8 multicast address range at its disposal.

service-allocation procedure it first places a lock on the hashtables and any associated parameters in order to prevent their alteration by any new connections that it may accept during the service-allocation process. The necessary steps in the service allocation procedure are now as follows. First the server divides the set of *N* multicast addresses into 2 sets, {*freeAddress*} and {*busyAddress*} containing *Nfree* and (*N-Nfree*) addresses respectively. {*freeAddress*} comprises all addresses from the server's *N* multicast addresses that are currently free, that is no longer in use by any transmissions that were started in previous epoch(s). The service-allocation procedure can determine the current status of each address, that is busy or free, by inspecting a so-called *busy flag* that is associated with each address.

Next the server must re-allocate each of the *Nfree* free addresses to a separate file. The *Nfree* files to be used for this purpose are obtained as follows. Firstly, an initial set of candidate files, {*fcandidate*} is chosen according to selection criteria that vary according to which mode of AMP with COS is being used. We define two modes of operation, Holdback and No-Holdback mode. With No-Holdback mode, {*fcandidate*} contains all files whose total number of outstanding requests is greater than one. With Holdback mode {*fcandidate*} is the same as with No-Holdback mode except all files whose number of outstanding class 1 requests is equal to 0 are excluded. Once the set of files {*fcandidate*} has been obtained in accordance with the particular mode of operation in use, the next step is to add or remove files from {fcandidate} in order to obtain a set of files {*felected*} containing *Nfree* members. If the number of files in {*fcandidate*} is denoted as *Ncandidate* then the set {*felected*} is obtained as follows.

If Ncandidate = Nfree:

In this case {felected}={fcandidate}.

If Ncandidate < Nfree:

An additional (*Nfree-Ncandidate*) files must be added to {*fcandidate*}. These are obtained from those files with the highest values of *Ri* not currently in {*fcandidate*} but which had an allocation to a multicast addresses at the previous epoch boundary.

If Ncandidate > Nfree:

For each file in {*fcandidate*} a measure of its suitability for multicast, known as its multicast suitability factor, *Sm* is calculated. This is defined as the reduction in the number of bytes that will be transmitted if the file's multiple outstanding requests are served by a single multicast rather than a separate unicast transmission for each outstanding request. This definition of *Sm* can be written more concisely in the form of equation (3-4) where count is the number of outstanding requests for the file.

$$Sm = (count - 1)\, filesize \tag{3-4}$$

{felected} is now formed from the *Nfree* most desirable files, that is those with the highest values of *Sm*, from the set *{fcandidate}*.

It is now necessary to assign each of the files contained in {*felected*} to a separate multicast address from the *Nfree* addresses contained in *{freeAddress}*. In doing this, any file in {felected} which, at the previous epoch boundary, was assigned a multicast address which is now contained in {freeAddress} is assigned the same address at this epoch boundary. This rule prevents unnecessary address allocation flapping. The assignment of the remaining multicast addresses in {freeAddress} to the remaining files in {felected} is arbitrary.

Each file in {felected} that was also contained in {fcandidate} will undergo multicast transmission commencing in the next epoch and is said to be multicast-active. For each outstanding request for a so-called multicast-active file the server sends a HTTP 302 response (redirect) over the existing HTTP/TCP connection. Included in this response will be the file's allocated multicast address together with the UDP destination port number to be used for the multicast. The server then removes the request from the appropriate hashtable in the set {*hashCountc*}.

Any remaining outstanding class 1 requests are now serviced over the existing HTTP/TCP connections and removed from $hashCount_1$, the hashtable for outstanding class 1 requests. Next each outstanding request of class n(n>1) is changed to class n-1 and the hashtables {$hashCount_c$} updated accordingly. The server then removes its lock on the hashtables and any requests that were received during the service allocation process will now be added to the hashtables which will subsequently be updated every time a new request arrives. The actual multicasting of the multicast-active files is not done immediately after the service-allocation process. Instead the server first waits for k epochs(k > 1) in order to allow sufficient time for all affected clients to receive the necessary information and join the appropriate multicast group.

3.1.2 Implementation of AMP with CoS for a Web Browser

Figure 3.2 shows how AMP with CoS can be implemented for a browser accessing HTML pages.

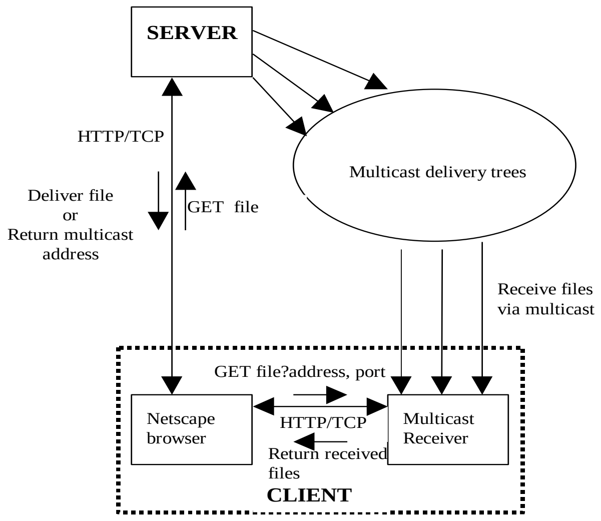


Figure 3.2: Implementation of AMP with CoS for a Web Browser

In order to fully leverage the benefits of AMP with CoS for delivery of HTML web pages 'intra-page pipelining' should be used, i.e. all components of a web page should be treated as a single entity. This means that the size of a *.html or *.htm page as far as AMP with CoS is concerned is equal to the sum of the size of the web page plus the size of all of its components, i.e. gifs, jpegs, frames. In addition should any web page be selected for multicast, the whole of the web page's components should be sent over a single multicast address using a multipart/related Multipurpose Internet Mail Extension (MIME) to encapsulate the page's components[36][41].

For each request that the server chooses to serve via multicast, a user-defined MIME type, x-mcast-start contained within the HTTP 302 redirect response serves as the trigger to start a multicast receiver application, to which are passed details of the multicast group to be used for transmission of the requested file. The response also contains the HTTP entity-header relating to the requested file as well as fields regarding the encoding and length of the file.

The subsequently multicasted packets include a lightweight application layer protocol above the UDP layer and the MIME type information used by the receiver to separate out the components of the web page. The lightweight application layer protocol is used to indicate the name of the file and the sequence number of the packet. The sequence number is set to 1 for the first packet sent and is incremented for each subsequent packet sent. The sequence numbers are used at the receiver to correctly order the received packets and to detect packet loss which is indicated by gaps in the sequence number space. As soon as a receiver has obtained all of the requested file it immediately leaves the multicast group. Reliability is achieved using integrated RMRJ/RMRP architecture which we present in chapter 5.

As proof of concept we have implemented the architecture of Figure 3.2 but with omission of both ‘intra-page pipelining’ and packet loss recovery features.

3.1.3 Emulation of AMP with CoS

In the longer term we intend to analyse the performance of our server in a real network environment. However, in order to assess some of the characteristics of our server at a much earlier stage we have built an emulator written in the Java language and incorporating the same service allocation code used by our server but with the addition of intra-page pipelining.

The input to our emulations is a server log representing all accesses over the 12 hour period 10am-10pm, March 7th 1998 of a British Telecom(BT) server at Ipswich, England but ignoring all CGI queries and any requests that did not receive a 200 response code[22] by the server. The level of traffic reflected in this log is insufficient to notice any significant resource savings using our scheme with reasonable values for the epoch period. However, the log is still useful in terms of its access pattern.

[5] examined the access pattern of 6 WWW servers and noted many invariants among them. For example, each of these 6 servers displays non-uniform referencing behaviour, that is concentration of requests among a small percentage of the requested files, the so-called hot files. For each of the 6 servers, 10% of the distinct files were responsible for 85-95% of all requests while less than 3% of all requests were for distinct files. Other invariants that [5] noted among the set of 6 server logs examined include Pareto[42] file size distribution and for files accessed more than once, inter-reference times that were independent and exponentially distributed. Moreover [5] notes that the set of 6 servers examined cover different orders of magnitude of server activity.

Consequently, we believe that emulation results obtained from the BT server log will remain meaningful even if the traffic level is scaled up by an order of magnitude, which is necessary in order to illustrate any potential resource savings of our scheme. If we assume that a real implementation of our scheme may have an epoch duration of say 3 seconds, then multiplying the traffic level by 10 times for this epoch duration will result in the same rate of requests per epoch as keeping the traffic rate the same but multiplying the epoch duration by 10. This is the approach taken in our emulations, that is, we use the unaltered BT server log but assume an epoch duration of 30 seconds.

The emulation assumes lossless transmission for both unicast and multicast. We discuss the impact of packet loss on our scheme in section 3.1.4. In addition the emulation assumes that each multicast transmission has been completed by the first epoch boundary after commencement of the multicast.

During emulation, the file sizes associated with each request in the log file were added together in order to give *bytesTotalFileSize* which represents the benchmark against which we measure the bandwidth resource saving of our system on the server output link.

If we denote the total number of file bytes transmitted by the server using our scheme as $bytesTransmitted_{ourScheme}$ then the percentage fewer file bytes, *Bsave* that the server will transmit with our scheme compared to a traditional unicast-only server can be written as equation (3-5)

$$Bsave = 100\frac{(bytesTotalFileSize - bytesTransmitted_{ourScheme})}{bytesTotalFileSize} \qquad (3\text{-}5)$$

The results of our emulations are shown in Figure 3.3 to Figure 3.10 where we assume that each request has a non-zero class. We use the notation *mode(share1:share2:share3:share4)* to denote the mode used and the expected proportion of requests taken up by each request class according to their probabilities. For example holdback(1:1:1:1) indicates that a given request could be any class between 1 and 4 inclusive with an equal probability. Holdback(1:0:0:0) indicates that all requests are of class 1. We assume an epoch period of 30 seconds in all our simulations and a static value of 0.5 epochs for the parameter k. In each of Figure 3.3 to Figure 3.8 the multicast eligibility threshold, M is set to 0 which essentially means that all requests passed to the service allocation module will be multicast eligible in accordance with equation (3-2).

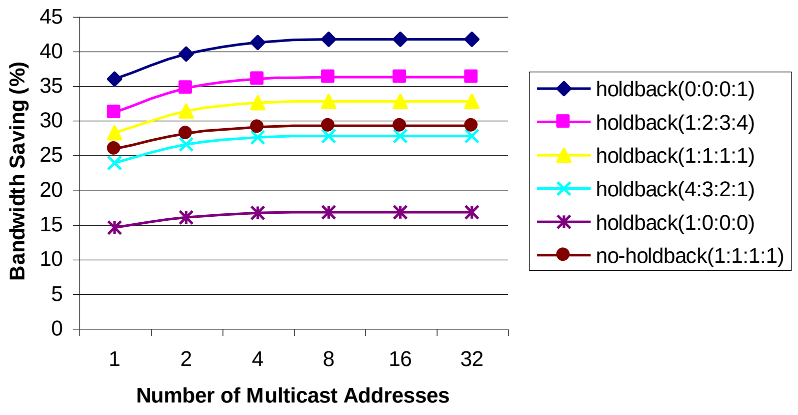


Figure 3.3: Bandwidth Saving of Holdback and No-Holdback Mode(M=0)

22 An HTTP response code of 200 signifies that the request was for a valid file and serviced OK.

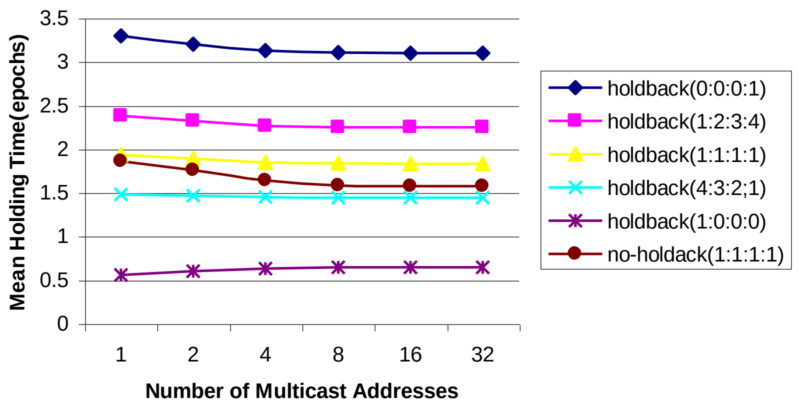


Figure 3.4: Mean Holding Time(all classes) of Holdback and No-Holdback Mode(M=0)

Figure 3.3 to Figure 3.8 include plots of holdback(1:1:1:1) and no-holdback(1:1:1:1) in order to allow a comparison between the two modes of operation to be made. As can be seen in Figure 3.4, holdback(1:1:1:1) gives a higher mean holding time than no-holdback(1:1:1:1) since it delays the multicast of files for as long as possible. A consequence of this is a better bandwidth saving than no-holdback since the holdback variant satisfies a higher number of requests per multicast. This improvement in bandwidth saving is illustrated in Figure 3.3. For example, when the number of multicast addresses equals 8 the % bandwidth savings of holdback(1:1:1:1) and no-holdback(1:1:1:1) compared to a unicast-only server are 32.7 and 29.2% respectively. From Figure 3.3 it can be seen that a near-optimal bandwidth saving is achieved with a relatively low number number of multicast addresses. For holdback(1:1:1:1) the bandwidth saving with 2 and 4 multicast addresses is 31.4 and 32.5% respectively compared to 32.7% which represents the ceiling which is reached at approximately 8 multicast addresses.

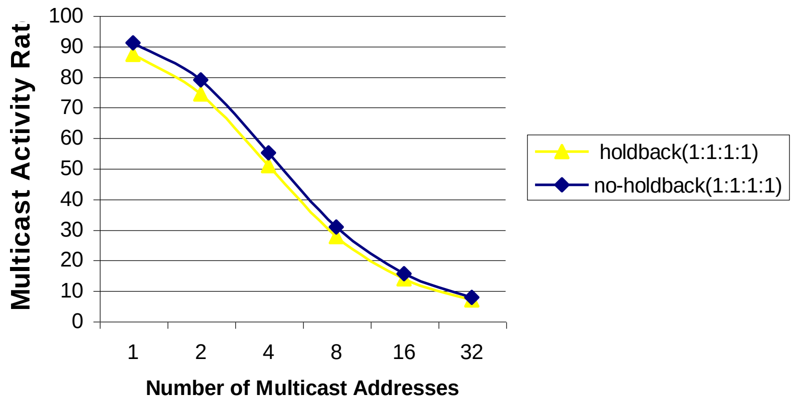


Figure 3.5: Multicast Activity Rate of Holdback and Non-Holdback Mode(M=0)

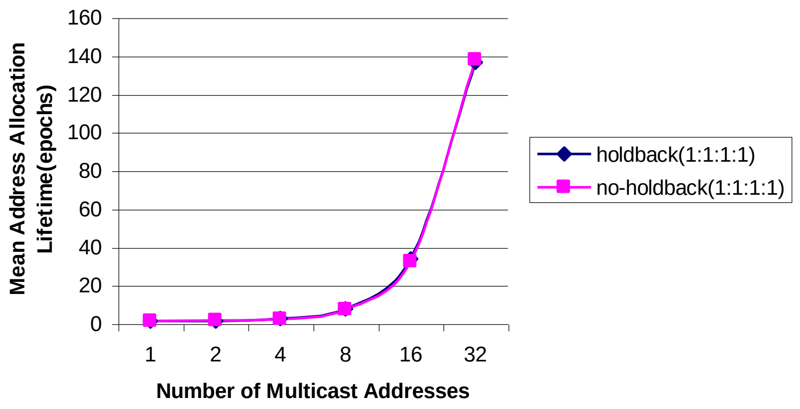


Figure 3.6: Mean (address, file) Allocation Lifetime of Holdback and No-Holdback Mode (M=0)

Figure 3.5 shows what we call the multicast activity rate which is the percentage of potential multicast opportunities at which a multicast transmission occurs. There are *N* potential multicast opportunities per epoch since up to one multicast may occur on each of the *N* multicast addresses every epoch. As expected, the multicast activity rate of the no-holdback scheme is greater than that of the holdback scheme although the difference is slight. The difference is due to the fact that with no-holdback multiple requests for the same file at an epoch boundary will always lead to a multicast in the next epoch providing a free address is available. By contrast with holdback such a situation will only lead to a multicast provided at least one of the requests is of class 1.

Figure 3.6 shows the address allocation lifetime of the two variants, that is the average number of epochs that each (file,multicast address) association lasts for. As can be seen the two variants differ little in this respect although the holdback variant does in fact have a slightly more stable address allocation due to a lower multicast activity rate which means a lower probability of a file with an assigned multicast address losing its address allocation at a given epoch boundary. A more stable address allocation lifetime might enable more efficient operation of certain packet loss recovery mechanisms.

Although there is little to choose between the holdback and no-holdback modes of our scheme, we prefer the holdback mode on the grounds that it offers better bandwidth saving while we do not regard the extra delay incurred as significant. For N=8 no-holdback(1:1:1:1) displayed a mean delay(all classes) of 1.58 epochs and a bandwidth saving of 29.2% while holdback(1:1:1:1) displayed a mean delay(all classes) of 1.83 epochs and a bandwidth saving of 32.7%.

In Figure 3.3 and Figure 3.4, in addition to holdback(1:1:1:1) and no-holdback(1:1:1:1) we also include results for the holdback scheme with different proportions of each class of request. As expected the greater the proportion of requests that are of a lower class(class 4 being the lowest) the greater the overall bandwidth saving and mean holding time.

Comparing holdback(1:1:1:1) to holdback(1:0:0:0) is equivalent to comparing the holdback scheme to classless AMP[23] in an environment where each class of request is equiprobable. Hence from Figure 3.3, examining the case of 8 multicast addresses it can be seen that the holdback variant of AMP with CoS produces a bandwidth saving of 32.7 % compared to 16.7 % in the case of classless AMP thereby illustrating the primary benefit of our scheme.

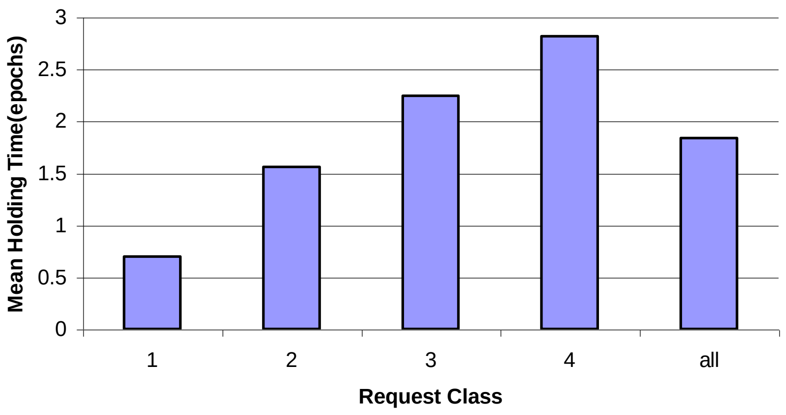


Figure 3.7: Mean Holding Time of each Class for Holdback(1:1:1:1) (M=0, N=8)

Figure 3.7 illustrates the service differentiation given to each request class for holdback(1:1:1:1) with 8 multicast addresses. There is a near-linear relationship between mean holding time and request class, that is the mean holding time of request class n is approximately n times that of request class 1.

[23] Holdback(1:0:0:0) is equivalent to classless AMP with a request accumulation period of 30 seconds which is the maximum value that classless AMP may use in order to satisfy the latency requirements of class 1 requests.

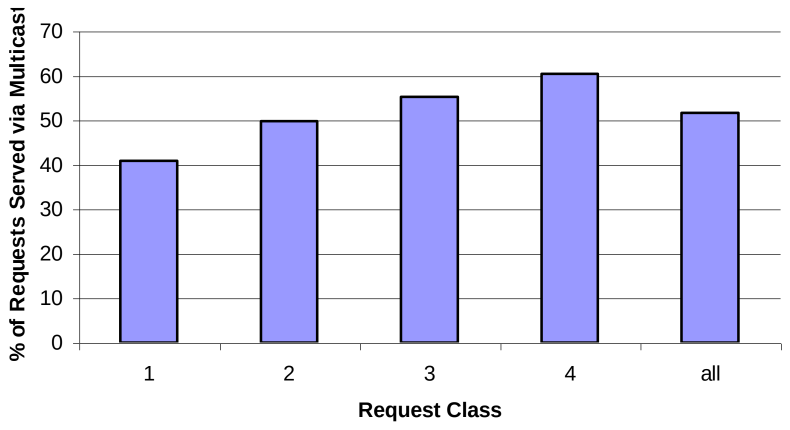


Figure 3.8: % of each Request Class Served via Multicast with Holdback(1:1:1:1) (M=0, N=8)

Figure 3.8 illustrates the % of requests that experience multicast delivery for each of the 4 request classes in the case of holdback(1:1:1:1) with 8 multicast addresses. As expected, higher latency request classes experience a higher % of requests undergoing multicast delivery.

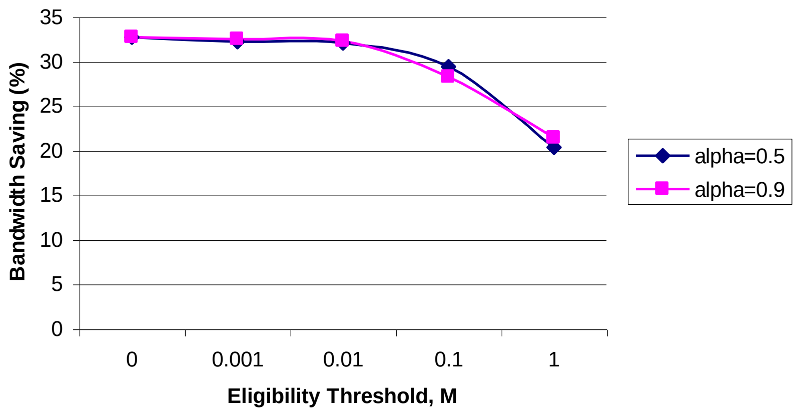


Figure 3.9: Effect of Multicast Eligibility Threshold(M) on Bandwidth Saving of Holdback(1:1:1:1) for N=8

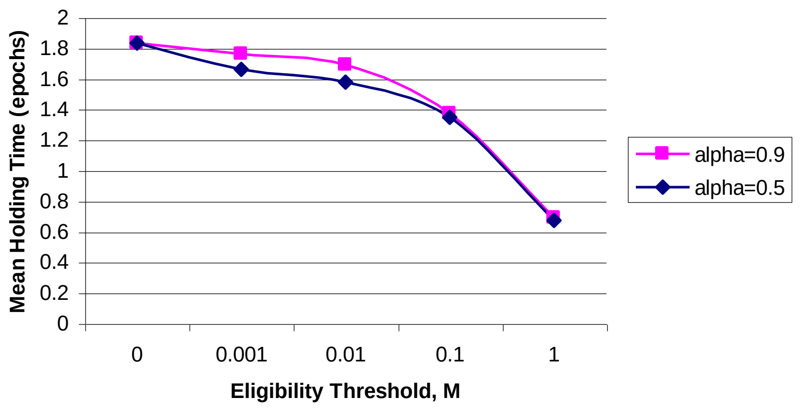


Figure 3.10: Effect of Multicast Eligibility Threshold(M) on Mean Holding Time of Holdback(1:1:1:1) for N=8

Figure 3.9 and Figure 3.10 analyse the effectiveness of the multicast eligibility algorithm of equation (3-2), the aim of which is minimise the mean holding time of requests without significantly reducing the bandwidth saving. Each request that arrives at the service allocation module is first subjected to the multicast eligibility test and if unsuccessful (i.e. deemed to be a request for a less popular file) is sent immediately to the output buffer queue for unicast transmission over the existing TCP connection.

If we examine the plots for alpha=0.5 then with M=0 the bandwidth saving for no-holdback(1:1:1:1) is 32.7 % and the mean holding time is 1.83 epochs. Increasing M to 0.0.1 reduces the mean holding time to 1.58 epochs and the bandwidth saving to 32.02 %. Relative to the figures for M=0, M=0.0.1 reduces the mean holding time and bandwidth saving by 14% and 1% respectively. This represents a significant reduction of mean holding time with only a negligible reduction of bandwidth saving thereby validating the effectiveness of the multicast eligibility algorithm.

3.1.4 Implementation Issues

3.1.4.1 Reliability

The emulation and discussions presented in the previous sections of this chapter assumed a loss-free environment. In reality however, some degree of loss is inevitable and since clients will usually require complete reception of the documents that they request, some means of providing a reliable service above the UDP multicast layer needs to be included. Our emulation assumed loss-free unicast as well as loss-free multicast and so we do not expect the bandwidth savings to differ significantly when packet loss is taken into account. Moreover the relative performance of AMP with classes of service compared to single class AMP should change very little when packet loss is taken into account if we assume that they both use the same recovery mechanisms for packets that were lost during multicast.

To achieve reliability we propose using the integrated RMRJ/RMRP architecture presented in chapter 5. AMP with CoS dynamically reallocates free addresses to files at epoch boundaries.

Although the address reallocation mechanism favours re-allocation of a given free address to the same file as the previous time it is impossible to guarantee no changes in (address, file) allocation if the number of addresses is less than the number of files. The procedure for address release, that is setting an address as being ‘free’ could be either server-initiated or client-initiated as will now be described. The client-initiated approach is more robust in the sense that it ensures a multicast address is never freed until each client has received all packets of the multicast transmission. The downside however is that it will require more TCP state to be handled at the server. The relative merits of each approach is an area for further study.

Server-initiated address release

Here each TCP connection between client and server would be closed as soon as the client had received the HTTP 302 redirect response indicating the multicast address to use. The server would class the address as being free at the first epoch boundary at which it had finished initial transmission of the file and had no outstanding retransmission requests for the file. If the address is declared free at time tfree then in the worst case a new initial transmission will commence on the address at time tfree+kb. If any clients are still missing lost packets at time tfree+kb then those clients might at some point thereafter begin to receive original data packets of a new transmission. If this should happen then those clients should leave the group and open a new TCP connection to the server to specifically request the still missing portions of the file pertaining to the lost packets.

Client-initiated address release

Here each TCP connection between client and server would be kept open until the client had received all of the file and informed the server accordingly. Only when each client had received all of the file and informed the server would the server deem the multicast address being used to be free once more.

3.1.4.2 Multicast Join-time and Setting k

Another issue that we need to consider is the delay between the server assigning multicast delivery to a requested file, and all intended recipients joining the multicast tree. This delay is given by the sum of two components. First, the delay incurred in sending the HTTP 302 response back to a receiver and second the delay incurred in the receiver joining the multicast tree in accordance with the multicast protocol in use. If the sum of these two components is greater than k epochs then the receiver may not have joined the multicast tree by the time the sender begins to multicast the requested file and so may not receive some (or all) of the file.

In theory the delay incurred by the HTTP 302 response could be minimised by setting the delay Type Of Service bit in the IPv4 packet header in order to convey the low delay requirement to intermediate nodes who might then take this into account when handling the packet. In practice however these bits are ignored by the vast majority of network nodes. Another alternative is to reserve resources in the network nodes between the server and clients along the path to be taken by the prospective HTTP 302 redirects. For this purpose the sender-based reservation protocol, DRP presented in chapter 4 could be used. Once the HTTP 302 packet passed through each node DRP could be used to remove the reservation. However, like the use of the Type of Service bits, the reservation approach is dependent upon support by the intermediate network nodes.

In our implementation and in the emulation, k is hardcoded to the value 0.5 epochs. In cases of network congestion this may provide insufficient time for all clients to join the multicast group prior to commencement of transmission of the requested file. In such cases, as far as the client is concerned the outcome is no different to that experienced by packet loss due to congestion in the network and can therefore be handled in the same way.

Further work needs to be done to look at the implications of the following alternative approaches. First, setting k dynamically in accordance with information the server is able to obtain regarding round-trip time to each receiver. Second, usage of an acknowledgement phase so that any multicast transmission does not commence until each of the intended recipients has verified its joining of the multicast group by sending a notification message over the existing TCP connection. Actually in the case of source-specific multicast[30] this acknowledgement phase may potentially be handled by the multicast routing protocol itself.

3.1.4.3 Multicast Suitability Factor, Sm

To illustrate the benefits of our scheme our definition of Sm in our emulations and implementation was concerned solely with the reduction in the number of transmitted bytes on the server output link. However, AMP with COS is also suited to more elaborate definitions of Sm that take into account other factors regarding the cost of multicast transmission[27]. This is an area for further work.

3.2 Optimised Batch Patching with Classes of Service

Optimised Batch Patching with Classes of Service(OBP with CoS) combines the principles of Optimised Patching together with AMP with CoS in order to produce a very bandwidth-efficient scheduling mechanisms for near video-on-demand(VOD). In order to present our work in the most logical fashion we build the scheme up from its core components and analyse it at each stage. We begin by applying AMP to the basic patching technique to give us Batch Patching. We then optimise the patching window in order to produce Optimised Batch Patching. Finally we introduce request latency classes in order to yield our complete scheme, OBP with CoS.

3.2.1 Batch Patching

Batch Patching consists of classless AMP applied to patching. The timing diagram for such a scheme is illustrated in Figure 3.11 which considers a video streaming server hosting a single video file. Any transmissions will be done at the same mean rate in terms of frames/s as the actual playout rate of the video itself.

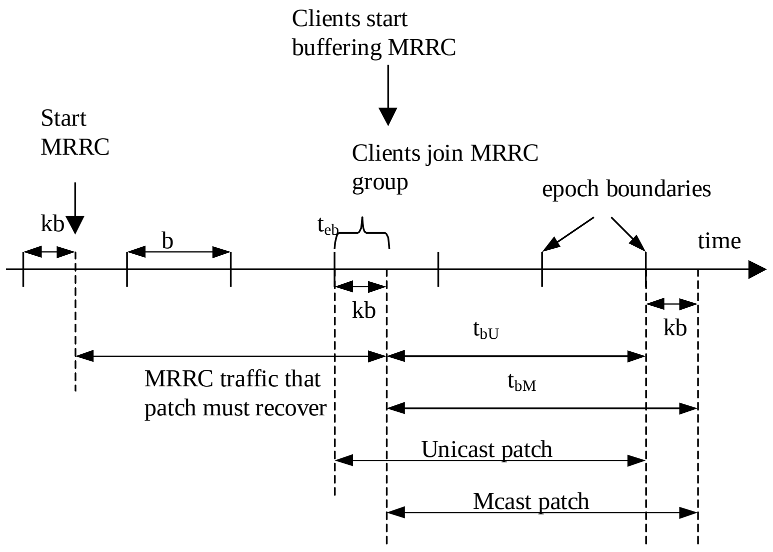


Figure 3.11: Timing Diagram for Batch Patching

We let b denote the batching interval and assume that each client has an identical amount of buffer space equal to B bytes. We let t_{eb} denote the time at an epoch boundary where a non-empty batch exists. Here the scheduler must decide whether service of the batch should commence using a new regular multicast or a patch to an existing regular multicast.

Should the server choose to start a new regular multicast, it will inform each client in the batch of the multicast address of the new regular channel.

Should the server choose to deliver a patch then it will inform each client in the batch of the multicast address of the Most Recent Regular Channel(MRRC) and use a patch to recover the early part of the file that has already been transmitted over the MRRC. If the number of clients in the batch exceeds 1 then the patch will be delivered by multicast in which case the clients will also be informed of the multicast address of the patching channel. In the case of a batch containing a single client the patch will simply be delivered via unicast.

Although the server begins informing clients of the multicast address of the MRRC at time=t_{eb}, in the worst case we allow an additional duration of kb(k<1) before each client has joined the channel and begun receiving traffic on it. This means that the patch is responsible for recovering all packets that were transmitted over the MRRC before teb+kb. Assuming that the patch is transmitted at the same[24] rate as the regular channel then the duration, dpatch of the patch will be given by

$$\text{dpatch} = (t_{eb} + \text{kb}) - (\text{start time of MRRC}) \qquad (3\text{-}6)$$

We assume that the server commences any unicast patch at time=t_{eb} but delays commencement of any multicast patch until time=t_{eb}+kb in order to allow each client in the batch sufficient time to join the multicast group for the patch. Transmission of the patch will then cease at t_{eb}+dpatch in the case of a unicast patch and t_{eb}+kb+dpatch in the case of a multicast patch.

Although a client will join the MRRC and begin receiving packets over it at any time between t_{eb} and t_{eb}+kb, any packets received over the MRRC before t_{eb}+kb are discardable since such packets will be recovered by the patch anyway. Once the patch finishes, playout of the buffered MRRC channel will begin at the client and will lag network transmission of the MRRC by dpatch seconds in the case of a multicast patch and dpatch-kb in the case of a unicast patch. The time-transformed buffer requirements at the client in the two cases are then as follows

$$t_{bM} = \text{dpatch} \qquad (3\text{-}7)$$

$$t_{bU} = \text{dpatch - kb} \qquad (3\text{-}8)$$

where t_{bM} and t_{bU} are the time transformed buffer requirements for the multicast patch and unicast patch cases respectively. The actual buffer requirements, B_M and B_U bytes, for the multicast and unicast patch cases respectively will then be as follows:

$$B_M = dpatch * rate(t_{bM}) \qquad (3\text{-}9)$$

$$B_U = (dpatch - kb) * rate(t_{bU}) \qquad (3\text{-}10)$$

[24] Same with regard to frames/s. The rate(frames/s) of a regular channel is the same as the playout rate of the video itself. Transmitting the patch at this same rate will require the least amount of client buffering.

where rate(t_b) bytes/s is the maximum of the mean rate of the video over any interval equal to t_b. t_b can be any value {b, 2b, 3b…} in the case of a multicast patch and any value {(1-k)b, (2-k)b, (3-k)b…} in the case of a unicast patch. The larger the value of t_b the smaller the value of rate(t_b). However it is unrealistic to assume that the value of rate would be made available to the server for every value of t_b. It is more likely that only the worst case values of rate, denoted $rate_1$ and $rate_{(1-k)}$ for t_b = 1 and (1-k) epochs respectively would be made available to the server. These values could be measured for each video and stored at the server. Rearranging equations (3-9) and (3-10) and inserting the worst case values of rate gives the following

$$dpatch_M < \frac{B}{rate_1} \tag{3-11}$$

$$dpatch_U < \frac{B}{rate_{(1-k)}} + kb \tag{3-12}$$

where $dpatch_M$ and $dpatch_U$ are the maximum duration multicast and unicast patches that can be accommodated without exhausting a client buffer of size B which is being used to buffer the regular channel. If a potential patch violates equation (3-11) in the case of multicast or equation (3-12) in the case of unicast then the server is precluded from sending it and must instead begin a new regular multicast in the next epoch.

3.2.2 Optimised Batch Patching(OBP)

Equations (3-11) and (3-12) impose an upper limit on the duration of a patch in accordance with the client buffer capabilities. If the value of B is not small, then allowing a patch duration as large as that determined by equations (3-11) and (3-12) will not yield the maximum bandwidth efficiency. This was discussed briefly in 2.1.2.2. Instead, higher bandwidth efficiency will be achieved by limiting the patch duration to some lower value which [11] refers to as the ‘patching window’, W. [11] optimises W for a true VOD Patching scheme and calls the technique Optimised Patching.

We refer to a Batch Patching scheme with W set to its optimal value as Optimised Batch Patching(OBP). With OBP, the duration, dpatch of any patch, whether unicast or multicast, must satisfy the condition of equation (3-13) as well as (3-11) and (3-12).

$$dpatch \le W \tag{3-13}$$

For each mature batch that existed at a given epoch boundary an OBP server would test whether a patch was possible by checking that the value of dpatch associated with the potential patch satisfied (3-13). If it didn’t then a patch could not be delivered and a new regular multicast would be scheduled instead. Assuming a potential patch was possible it would commence immediately via unicast if the number of requests in the batch was equal to 1, and kb seconds later via multicast if the number of requests in the batch was greater than 1.

We will now derive an equation that relates the bandwidth consumption of Optimised Batch Patching with the parameters b, λ, W and mean video rate, r. The derived equation will allow us to calculate the optimal value of W for specific values of b, λ and r. In the following analysis we consider a single video file and assume that clients have ample buffer space for all values of dpatch.

We refer to an RM-epoch as one in which a regular multicast was started and a non-RM-epoch as one in which a regular multicast did not begin. In order to calculate the mean transmission rate on the server output link we need to determine both the mean interval between RM epochs and the mean of the aggregate number of bytes contained in all patches commencing between two adjacent RM epochs chosen at random.

Following an RM epoch, the next RM epoch will be triggered by the next occurrence of a non-empty batch whose associated value of dpatch exceeds W and so violates equation (3-13). It is not possible for this to happen in the first W/b epochs and so these will definitely be non-RM epochs. Following this series of non-RM epochs the next non-empty batch to occur will definitely violate equation (3-13) and hence cause the next epoch thereafter to be a RM epoch.

These observations yield the following equation for the mean number of epochs between two adjacent RM epochs

$$n = \operatorname{int}\left(\frac{W}{b}\right) + \left(\sum_{i=1}^{\infty} i\, P(0)^i \left(1 - P(0)\right)\right) \tag{3-14}$$

where P(i) is the probability of the number of requests in a batch being equal to i. Assuming a Poisson arrival process, P(n) is given by equation (3-15)

$$P(i) = \frac{\lambda^i e^{-\lambda}}{i!} \tag{3-15}$$

where λ is the mean number of requests per batch.

For ease of representation in forthcoming analysis we denote P(0) as P. From equation (3-15) this gives

$$P = P(0) = e^{-\lambda} \tag{3-16}$$

Returning now to (3-14) let us examine the special case where W is an integer multiple of b. Equation (3-14) then becomes

$$n = \frac{W}{b} + \sum_{i=1}^{\infty} i\, P^i \left(1 - P\right) \tag{3-17}$$

Let us now evaluate the summation that represents the second term on the right hand side of equation (3-17)

$$\sum_{i=1}^{\infty} i\, P^i(1-P) = (1-P)(P + 2P^2 + 3P^3 + 4P^4 + 5P^5 + \ldots)$$

$\Rightarrow$

$$\begin{aligned}\sum_{i=1}^{\infty} i\, P^i(1-P) &= P + 2P^2 + 3P^3 + 4P^4 + 5P^5 + \ldots \\ &\quad - P^2 - 2P^3 - 3P^4 - 4P^5 - \ldots\end{aligned}$$

$\Rightarrow$

$$\sum_{i=1}^{\infty} i\, P^i(1-P) = P + P^2 + P^3 + P^4 + P^5 + \ldots \qquad (3\text{-}18)$$

Substituting for the geometric series on the right hand side of equation (3-18) we have

$$\sum_{i=1}^{\infty} i\, P^i(1-P) = \frac{P}{(1-P)} \qquad (3\text{-}19)$$

Substituting (3-19) into (3-17) we have

$$n = \frac{W}{b} + \frac{P}{(1-P)} = \frac{W(1-P) + bP}{b(1-P)} \qquad (3\text{-}20)$$

Now only the first W/b epochs following an RM epoch are capable of generating a patch. For each of these epochs the probability of generation of a patch will be given by 1-P. Hence the mean, μ of the aggregate duration of all patches commencing between two adjacent RM epochs chosen at random is given by

$$\mu = b\sum_{i=1}^{\frac{W}{b}} i(1-P) \qquad (3\text{-}21)$$

Solving the summation in equation (3-21) gives the following

$$\mu = b(1-P)\left(\frac{W}{b}+1\right)\frac{W}{2b} = \frac{(1-P)W^2}{2b} + \frac{(1-P)W}{2} \qquad (3\text{-}22)$$

The number of bytes, α corresponding to μ is then as follows

$$\alpha = \frac{(1-P)rW^2}{2b} + \frac{(1-P)rW}{2} \qquad (3\text{-}23)$$

where r is the average rate in bytes/s of a single transmission of the video. To obtain an estimate of the mean transmission rate on the server output link we need to consider an RM epoch and the subsequent run of non-RM epochs before the next RM epoch. The mean time period, t of such a sequence is given by

$$t = nb + b = b(1+n) \tag{3-24}$$

And the number of bytes, β in all transmissions commencing in this period is given by

$$\beta = \alpha + Tr \tag{3-25}$$

where T is the total run time of the video.

The average transmission rate, R on the server output link can now be written as follows

$$R = \frac{\alpha + Tr}{b(1+n)} \qquad \text{(b<W<T)} \tag{3-26}$$

Substituting equations (3-20) and (3-23) into (3-26) we have

$$R = \frac{\dfrac{(1-P)rW^2}{2b} + \dfrac{(1-P)rW}{2} + Tr}{b\left(\dfrac{W(1-P) + bP + b(1-P)}{b(1-P)}\right)} = \frac{(1-P)rW^2 + (1-P)brW + 2rbT}{2b^2\left(\dfrac{W(1-P) + b}{b(1-P)}\right)}$$

$\Rightarrow$

$$R = \frac{(1-P)rW^2 + (1-P)brW + 2rbT}{2bW + \dfrac{2b^2}{(1-P)}} \tag{3-27}$$

Equation (3-27) can also be written as equation (3-28) where R1 and R2 are given by equations (3-29) and (3-30) respectively.

$$R = R1R2 \tag{3-28}$$

$$R1 = (1-P)rW^2 + (1-P)brW + 2rbT \tag{3-29}$$

$$R2 = \frac{1}{2bW + \dfrac{2b^2}{1-P}} \tag{3-30}$$

Figure 3.12 sketches equations (3-29) and (3-30). At W=0 R1 and R2 are both positive and have negative gradients. Consequently the product R1R2 will have a negative gradient at W=0. Also from equation (3-27) it can be seen that R tends to infinity as W tends to infinity. The combination of these two observations indicate that a minimum exists in R for a value of W between 0 and infinity.

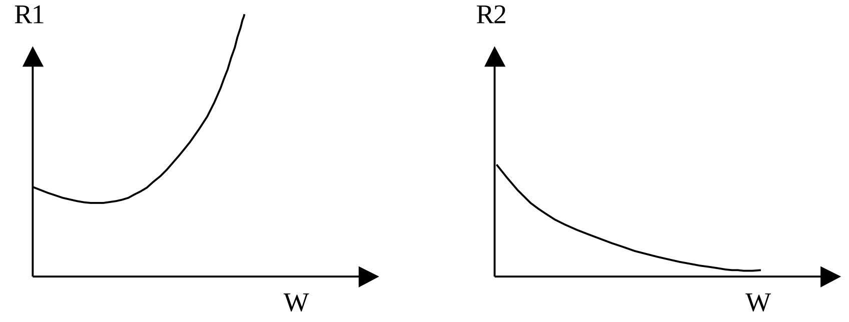


Figure 3.12: R1 and R2

The location of the minimum in R can be found by differentiating R and setting the result equal to 0. This yields the following equation

$$\frac{dR}{dW} = \frac{-2b(r(1-P)W^2 + (1-P)brw + 2rbT)}{\left(2bW + \frac{2b^2}{1-P}\right)^2} + \frac{2r(1-P)W + (1-P)br}{2bW + \frac{2b^2}{1-P}} = 0$$

(3-31)

Multiplying both terms in equation (3-31) by the denominator of the second term we have

$$\frac{-2b(r(1-P)W^2 + (1-P)brW + 2rbT)}{\left(\frac{2bW(1-P) + 2b^2}{1-P}\right)} + 2r(1-P)W + (1-P)br = 0$$

$\Rightarrow$

$$\frac{-(r(1-P)W^2 + (1-P)brW + 2rbT)}{W(1-P) + b} + 2rW + br = 0 \qquad (3\text{-}32)$$

Multiplying all terms of equation (3-32) by the denominator of the first term gives the following

$$-r(1-P)W^2 - (1-P)brW - 2rbT$$
$$+ 2r(1-P)W^2 + W(1-P)br + 2rbW + b^2 r = 0$$

$\Rightarrow$

$$r(1-P)W^2 + 2rbW + b^2 r - 2rbT = 0 \qquad (3\text{-}33)$$

Solving for W in equation (3-33) we have

$$W = \frac{-2rb \pm \sqrt{4r^2b^2 - 4r^2(1-P)(b^2 - 2bT)}}{2r(1-P)}$$

$$\Rightarrow$$

$$W = \frac{-2rb \pm 2r\sqrt{b^2 - (1-P)b^2 + 2(1-P)bT}}{2r(1-P)}$$

$$W = \frac{-b \pm \sqrt{Pb^2 + 2(1-P)bT}}{1-P} \tag{3-34}$$

The result in equation (3-34) gives the two roots to equation (3-33) which is continuous in time. However since W can never be negative the negative root must be invalid. In addition, in deriving (3-33) we assumed that W was an integer multiple of b. Hence for consistency the true solution, that is the value of W that is an integer multiple of b and which gives the minimum in R is given by equation (3-35).

$$W = b\,\mathrm{int}\left(\frac{-b + \sqrt{Pb^2 + 2(1-P)bT}}{b(1-P)} + \frac{1}{2}\right) \tag{3-35}$$

Equation (3-35) is an important result that can be exploited by the server to optimise bandwidth usage of Batch Patching. In addition it can be used to test buffering capabilities of clients and price them accordingly if limitations in their buffering capabilities prevent achievement of optimal bandwidth usage.

Figure 3.13 and Figure 3.14 plot equation (3-27) for videos of duration 90 and 180 minutes respectively with b set to 1 minute. In each case 4 plots are shown, each one for a different value of λ, the mean number of requests per batching interval. In Figure 3.13 the simulation results for the λ=1 case are also shown. The close match between simulation and analysis provides a strong degree of confidence in our analysis.

Figure 3.15 plots curves for the Optimised Patching scheme of [11] which produces true VOD. The curves in Figure 3.15 were obtained from the following equation[25] which was derived in [11] and gives the normalised transmission rate for Optimised Patching as a function of video run time, T, patching window size, W and arrival rate, λ.

$$\frac{R}{r} = \frac{2T\lambda + W(W+1)}{2W\lambda + 2} \tag{3-36}$$

[25] In equation (3-36), the units of W and T are seconds while those of λ are requests/second. These parameters were scaled to minutes and requests/min respectively in order to produce the plots in Figure 3.15.

It should be noted that equation (3-36) and indeed equation (3-35) is only applicable provided spare bandwidth is available on the output link.

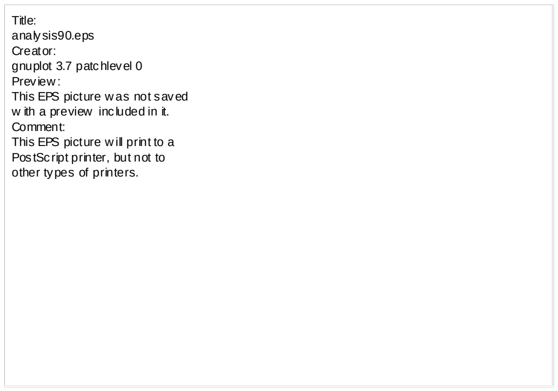

Title:
analysis90.eps
Creator:
gnuplot 3.7 patchlevel 0
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

Figure 3.13: Normalised Transmission Rate vs W at different values of λ(requests/min) for Batch Patching of a 90 minute video

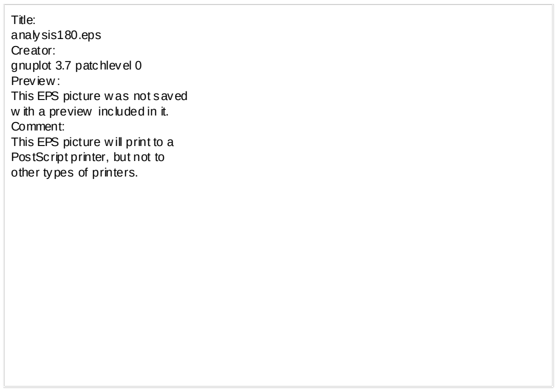

Title:
analysis180.eps
Creator:
gnuplot 3.7 patchlevel 0
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

Figure 3.14: Normalised Transmission Rate vs W at different values of λ(requests/min) for Batch Patching of a 180 minute video

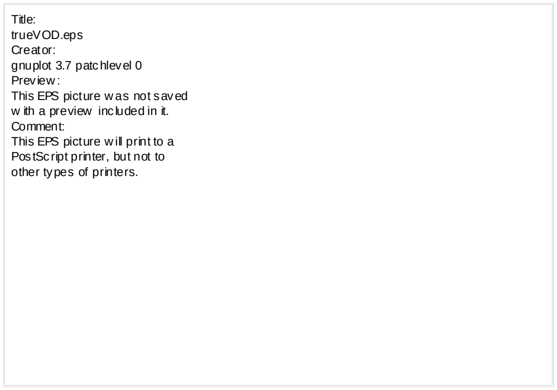

Title:
trueVOD.eps
Creator:
gnuplot 3.7 patchlevel 0
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

Figure 3.15: Normalised Transmission Rate vs W (mins) at different values of λ(requests/min) for Patching of a 90 Minute Video.

In Figure 3.13 the minima of the curves for λ equal to ∞, 2, 1, 0.5 and 0.25 are located at W equal to 12, 13, 15, 19 and 24 minutes respectively. Using this information the server could continuously update its value of W to reflect the optimal value for the current arrival rate, λ. Alternatively, in order to simplify implementation of the server the value of W could be set to a fixed value that was close to optimal across a wide range of arrival rates. For example in Figure 3.13 with W equal to 17 minutes the value of R/r on each curve is no more than 5% greater than its minimum value for that curve.

In Figure 3.14 the minima of the curves for λ equal to ∞, 2, 1, 0.5 and 0.25 are located at W equal to 18, 19, 22, 28 and 36 minutes respectively. In addition, with W equal to 25 the value of R/r on each curve is no more than 6% greater than its minimum value for that curve.

Comparision of Figure 3.13 and Figure 3.15 clearly illustrates the bandwidth savings of Optimised Batch Patching compared to Optimised Patching, particularly for larger values of λ. In the Batch Patching plots of Figure 3.13, with optimal setting of W the value of R/r can be kept below 13 no matter how high the arrival rate. By contrast in the Patching plots of Figure 3.15, the value of R/r rises rapidly with λ and at a value of λ=8 requests/epoch optimal setting of W gives a high value of R/r of 38.

3.2.3 OBP with CoS

In the previous section we showed that for a given arrival rate, video run time, and epoch period, a specific value of W exists that yields a minimum in the bandwidth consumption on the server output link in the case of batch patching. We now extend the basic principles of OBP to allow a latency class to be associated with each request. We call this extended scheme OBP with CoS(Classes of Service). Here the latency class serves the same function as the class in AMP with CoS in that it indicates the maximum number of epochs that the server is allowed to hold the request for before making a final service decision.

Request Parameters
VideoFileName
B (buffer size)
Class
MaxPrice
Type

Table 3-1: Request Parameters for OBP with CoS

Table 3-1 shows that each client request for a specific video file also indicates the client’s current receiver buffer setting together with desired latency class. The class determines the cost of receiving a specific video. In addition if the value of B is insufficient to permit optimal duration patching then a cost penalty may be incurred as discussed in more detail in section 3.2.8. The MaxPrice parameter indicates the maximum price the user is willing to pay for fulfillment of the request.

3.2.4 OBP with CoS Request Type

Each OBP with CoS request includes a type parameter which indicates whether the request is tentative or final. Only final requests with a MaxPrice value less than or equal to the actual cost are submitted to the service scheduler at the server. Any final request where this is not the case and any tentative request will trigger a response containing a buffer/class price matrix such as that shown in Table 3-2.

Class	Buffer Space			
	100M	400M	700M	1G
1	£4.00	£3.40	£2.70	£2.00
2	£3.20	£2.70	£2.10	£1.60
3	£2.60	£2.10	£1.70	£1.30
4	£2.00	£1.70	£1.35	£1.00

Table 3-2: Example Buffer/Class Matrix Returned in Response to a Tentative Request.

If returned in response to a tentative request the buffer/class matrix indicates what the cost of fulfilling the tentative request would be should it be resent with type set to final. The buffer/class matrix may also contain additional pricing information indicating how the price would change if the request was resent with class and/or buffer size set to different values. The buffer/class matrix could be embedded inside a response containing client-side interactive functionality, e.g. a java applet or a page of javascript code. The initial settings contained in the tentative request could be default values that were configured at the client. Upon receiving the response the user would have the option of altering the request settings at the client side and viewing the corresponding cost before dispatching the request as type ‘final’ for processing by the server scheduling algorithm. The MaxPrice parameter in the request would reflect the cost indicated to the user for the final settings.

3.2.5 OBP with CoS Service Scheduling Algorithm

Batch Class instance variables
VideoFileName
B (buffer size)
count
class

Table 3-3: Instance Variables of the Batch Class

Table 3-3 shows the instance variables of the batch class which stores information about a batch of requests for the same file. B is the minimum buffer size of all requests in the batch and class is the minimum ‘carried over’ class of all requests in the batch. Figure 3.16 shows the service scheduling algorithm that is applied at each epoch boundary using holdback[26] mode. First each new request that arrived in the previous epoch is added to the corresponding batch and the batch parameters updated. Then all mature batches are scheduled for service while the class of all remaining batches is decremented.

```
For each new request, r {
        batch = getBatch(r.videoFileName);
        If (r.B < batch.B) {
                batch.B=r.B;
                batch.count++;
        }
        if (r.class<batch.class) {
                batch.class=r.class;
        }
}

For each batch {
        if (batch.class>1)
            batch.class --;
        else {
            calculate dpatch;
            calculate W;
            Breq←BufferReq(batch)
            if ( dpatch>W ) | ( Breq>batch.B )
                start new MRRC in kb seconds
            else
                patch(batch);
        }
}

function Patch(batch) {
        if (batch.count>1)
                Start Multicast of patch in kb seconds
        else
                Start unicast patch immediately
}

function BufferReq(batch) {
        if (batch.count==1)
                Breq=B_M
        else
                Breq=B_U
        return Breq
}
```

Figure 3.16: Service Scheduling Algorithm for OBP with CoS and Holdback.

[26] As for AMP with CoS either holdback or non-holdback modes could be used for OBP with CoS.

3.2.6 Simulation of OBP with CoS

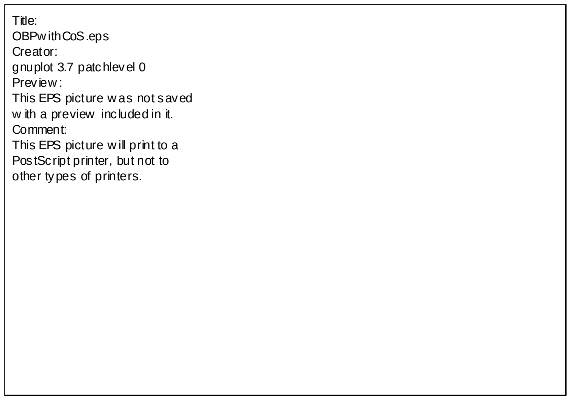

Title:
OBPwithCoS.eps
Creator:
gnuplot 3.7 patchlevel 0
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

Figure 3.17: Simulation Results of Normalised transmission rate vs W/b for a 90 minute video using firstly, Optimised Batching, secondly, OBP with CoS and different CoS ratios(λ=1)

In Figure 3.17 we plot our simulation results for OBP with CoS[27] for a 90 minute video using OBP with CoS and different CoS ratios(λ=1). As can be seen the higher the proportion of lower latency class requests the higher the bandwidth consumption. Comparing (1:1:1:1) to (1:0:0:0) is equivalent to comparing OBP with CoS to classless OBP in an environment where each class of request is equi-probable.

The curve of Optimised Patching is equivalent to all requests having 0 latency and consequently has the highest bandwidth consumption of all the curves. Hence the plots clearly demonstrate the bandwidth savings attainable using OBP with CoS compared to Optimised Patching. The bandwidth savings of OBP with CoS compared to Optimised Patching will become even greater as λ increases.

For OBP with CoS the server could set the value of W for a video to the optimal value for classless OBP for the current value of λ as given by equation (3-34). In the case of λ=1 as in Figure 3.17 this strategy yields a value of W=15 mins. With this setting of W the value of R/r on each OBP with CoS curve in Figure 3.17 is within 6% of the minimum value for that curve. In other words our formula, equation (3-34), for optimal setting of W in the case of classless OBP gives near optimal setting of W across the full range of traffic mixes in the case of OBP with CoS.

[27] Strictly speaking each curve can only be said to truly represent OBP with CoS around the minima of the curve since outside this region of W the scheme is no longer optimised.

3.2.7 Dimensioning of OBP with CoS

Figure 3.16 assumes that a spare multicast address is always available for each multicast transmission. If this is not the case then additional unicast transmissions will be required which will increase bandwidth consumption.

In order to dimension the system so that a spare multicast address is always available for any desired multicast transmission we need to calculate the maximum number of multicast transmissions that may be active at any one time.

For each video the maximum instantaneous number of multicast addresses required by regular transmissions will be given by T/W.

To calculate the number of addresses required for patching let us consider a popular video. For a popular video the number of epochs in between successive RM epochs will be equal to W. Let us refer to each run of non-RM epochs as a frame. In each non-RM epoch a new multicast patch will commence. The duration of a patch commencing in non-RM epoch s (1<=s<=W) will be equal to s epochs.

If we consider an epoch k then a multicast patch started in any epoch s(1 ≤ s ≤ k) in the current frame will still be active in epoch k provided 2s-k >= 0, i.e. 2s+1-k >= 1. In addition a multicast patch started in any epoch s(k<s≤W) in the previous frame will still be active in epoch k in the current frame provided 2s-W-k ≥ 1. These observations yield the following equation for the number,Npatch(k) of multicast patches that are active in epoch k.

$$Npatch(k) = \sum_{s=1}^{k} \min(1, \max(0, 2s+1-k) + \sum_{s=k+1}^{W} \min(1, \max(0, 2s-W-k) \qquad (3\text{-}37)$$

The maximum number of multicast addresses required by a video for a given value of W and T is given by (3-38).

$$Naddr(W) = \frac{T}{W} + \max(Npatch(k) \mid k = 1,2,3...W) \qquad (3\text{-}38)$$

If insufficient multicast addresses are available to satisfy the requirement of (3-38) over all values of W[28] then at a minimum the system should be designed and dimensioned so that a spare multicast address is always available for any desired regular transmission. To be able to do this we need to calculate the maximum instantaneous number of desired regular multicasts of all videos which we denote as Nreg. If the number of multicast addresses at the server's disposal is greater than Nreg but not guaranteed to be large enough to accommodate all patch multicasts as well as regular multicasts then the server would need to use some kind of address allocation policy such as only multicasting a

[28] Assuming of course that W is set to its optimum value for the given request traffic mix and rate.

desired patch provided doing so would leave at least one spare multicast address per video. The value Nreg will be given by the following equation

$$Nreg = \sum_{i=1}^{V} \frac{T_i}{W \min_i} \tag{3-39}$$

where V is the number of videos and Wmin$_i$ is the smallest value of W for video i under all conditions. From Figure 3.17 it can be seen that the minimum optimal value of W occurs for a traffic mix of (1:0:0:0). OBP with this traffic mix behaves the same as classless OBP. From Figure 3.13 and Figure 3.14 it can be seen that with classless OBP the value of W is at a minimum for $\lambda=\infty$ in which case P will be equal to 0. Hence inserting P=0 into equation (3-35) yields the minimum value of W as follows.

$$W \min = b \operatorname{int}\left(\frac{-b + \sqrt{2bT}}{b} + \frac{1}{2} \right) \tag{3-40}$$

3.2.8 Handling and Costing of Requests

Specification of a lower latency class increases both the QoS experienced by the user and the network bandwidth consumed. Consequently it is reasonable to expect the end-user to be charged more for a lower latency class than a higher latency one in order to compensate the network provider for the associated increase in bandwidth consumption. In addition if the value of B in a client's request is insufficient to permit optimal duration patching then a cost penalty may be added by the server. For this purpose the server would check for fulfillment of equation (3-11) with dpatchM set to the projected value of dpatch for a final service decision made at the latest[29] possible epoch boundary for the class of the request.

In the case of video-on-demand one could imagine at least two likely pricing structures. Firstly pricing could be done in accordance with the 4-tuple (popularity, running time, latency class, buffer cost penalty). Less popular videos would incur a higher cost since the incremental bandwidth consumption per request is higher for a multicast tree with fewer receivers.

A second pricing structure one could imagine would simply be according to the 2-tuple (latency class, buffer cost penalty). This is workable in the case of a video server delivering feature films which are likely to be of similar length. The server could monitor the popularity of each video and remove from its archive those videos that were not sufficiently popular and consequently whose

[29] A final service decision on a request of class n must be made by the nth subsequent epoch boundary at the latest.

earnings/overhead ratio was poor. The overhead of a video comprises storage overhead at the server as well as bandwidth consumption in the network.

3.3 Chapter Summary

In this chapter we have presented two new methods for service scheduling of on-demand multicast servers.

Our first method known as Asynchronous Multicast Push with Classes of Service (AMP with CoS) is applicable to on-demand multicast servers where the probability of transmission overlap is small while the second scheme known as Optimised Batch Patching with Classes of Service (OBP with CoS) is applicable to on-demand multicast servers where the probability of transmission overlap is large.

Both of our new methods allow the client to indicate its latency requirements in its request so that the server is able to maximise multicast batch size within the constraints of the individual client latency requirements. As a result, both AMP with CoS and OBP with CoS achieve bandwidth savings compared to classless AMP and Optimised Patching respectively in an environment with heterogeneous client latency requirements.

We presented an implementation of AMP with CoS to be used for delivery of web-pages to web-browsers and noted that for maximum benefit intra-page pipelining should be used. As proof of concept we also built a working prototype of this implementation albeit without intra-page pipelining.

We emulated AMP with CoS with an HTTP log access pattern as input and applying full intra-page pipelining. Our results clearly demonstrated the relationship between request class mix and bandwidth consumption, the greater the proportion of lower latency class requests the higher the bandwidth consumption. In a real-life scenario some cost-penalty would need to be incurred by the user for registering a lower latency class of request. Our simulation results also highlighted the bandwidth savings achievable using AMP with CoS compared to classless AMP in an environment with heterogeneous client latency requirements. We also discussed various implementation issues such as reliability, multicast join time and definition of the multicast suitability factor.

In our presentation of OBP we derived equations that allow the optimal patching window size to be calculated for specific values of request arrival rate, epoch period and video run time. In addition we discussed how these analysis results of OBP could be used to estimate the optimal maximum patch duration for the more general case of OBP with CoS. Following this we simulated OBP with CoS in order to illustrate its bandwidth consumption characteristics. We finished the chapter by discussing some implementation issues of OBP with CoS, such as dimensioning of the service as well as handling and costing of requests.

4 A New Resource Reservation Protocol

In section 1.2 we identified the need for a resource reservation mechanism as part of our overall QoS framework for on-demand multicast services. The state of the art with regards to resource reservation protocols was presented in section 2.2 together with a discussion of their relative strengths and weaknesses. We now present our scheme called DRP(Dynamic Reservation Protocol) which combines selective features of these protocols together with several new ones 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 change session-wide reservation levels in large-scale multicast sessions.
- Support for heterogeneity of reservation QoS classes among receivers of a multicast session.
- No need to synchronise reservation styles among hosts of a multicast session.

Although motivated by the needs of on-demand multicast services, DRP has sufficient generality to make it suitable for a wide range of communications scenarios including unicast as well as multicast. DRP allows reservations to be set up 'on-the-fly' by sending Reservation packets, RES in-line with the data flow.

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 and sender with certain feedback and end-to-end path information. As with RSVP and YESSIR, all flow-specific node state set up using DRP is so-called 'soft-state' which means that it will be deleted in the absence of any refreshes within a certain timeout period.

4.1 Design Principles

4.1.1 Sender initiated reservations

The use of in-line reservation packets allows the sender to set up new reservations, or alter existing reservations, on demand at any point in the data transfer. With DRP, the sender application wishing to facilitate 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 the RES packet passes through each router along the end-to-end path it is subjected to the router's admission control mechanism which if successful results in setup of the desired reservation at the appropriate router output port.

After sending the first RES message the sender is free to send further RES packets, possibly requesting different levels of resources, at any time. 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[30] the low QoS control dynamics would usually preclude such action. Although ATM's ABT/IT outlined in section 2.2.4 allows reservations to be sent in-line with data, unlike DRP, ABT/IT requires a connection-establishment phase prior to sending any reservation messages. Although YESSIR also uses in-line explicit[31] reservation messages, it does not allow the sender as much control over the timing of their generation since YESSIR explicit reservations can only be carried in periodically generated RTCP sender reports.

Unlike ABT/IT, a sustainable cell rate for a data flow is not specified with either DRP or YESSIR. This leads to two notable differences in the operation of ABT/IT compared to DRP and YESSIR.

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[32]. By contrast, with both DRP and YESSIR, 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. As discussed in section 2.2.4.1.3, in order for ABT/IT to promise a specified probability of a reservation change succeeding, the connection must have an associated Sustainable Cell Rate(SCR) which is negotiated at connection establishment. The price for offering an SCR to a given connection is a greater willingness by nodes to reject connection requests for other flows.

A second notable difference between ABT/IT and the protocols DRP and YESSIR concerns the QoS commitments delivered to the data flow once a reservation has been accepted. With both DRP and YESSIR, 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

[30] Traditional ATM signalling(e,g, Q.2931 and UNI) requires end-to-end handshaking.
[31] YESSIR also facilitates in-line implicit reservation messages.

this will be true even if the data flow violates the traffic specification agreed at reservation setup in which case any excess traffic will experience best-effort forwarding by routers. 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. This network-initiated renegotiation of the reserved BCR may be necessary to ensure that the sustainable cell-rates of other connections is not compromised.

4.1.2 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[59] and Guaranteed Service[45] 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[33]. 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 experienced 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[34] 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.

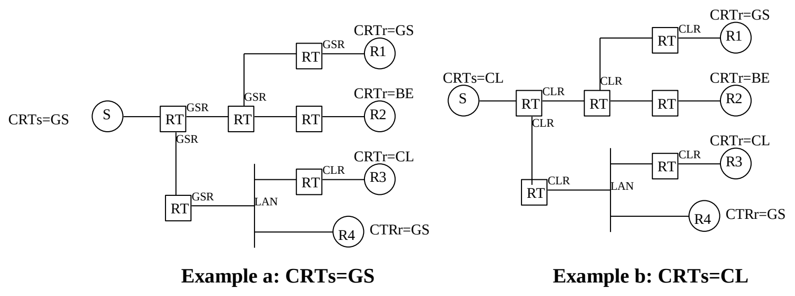
[32] However the probability that the initial connection request will be accepted at the requested SCR is not quantified.

[33] A receiver might wish to downgrade QoS class if doing so might reduce cost if being billed. Alternatively the receiver may be aware of bandwidth limitations that preclude full support of the higher class along the path.

[34] 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 arrive on one of the node's outgoing interfaces. In the case of a source-based tree the terms incoming and outgoing interface are unambiguous. In the case of a bi-directional 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.

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' required 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 4.1 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 4.1: Heterogeneity of Reservation Classes Between Receivers of Same Session

Table 4-1 compares the DRP approach to providing end-to-end delay bounds with those of RSVP, ABT/IT and YESSIR. 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[35]. 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 controlled by end nodes.

Although YESSIR facilitates sender control of end-to-end delay, the mechanism by which it is achieved is less optimal than DRP and can result in over-reservations along some of the paths to receivers, as was noted in 2.2.5.1.2.

Like DRP, RSVP facilitates end node control of end-to-end delay albeit by receivers rather than 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

[35] 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.

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 simple merging of RTN messages and RTN state in routers to ensure scalability to large multicast sessions as described in section 4.2.

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

Table 4-1: 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 initiate session-wide changes to Guaranteed Service experienced by receivers of a multicast session. As described in section 2.2.3.1.6, 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 set up reservations only once on each on-tree outgoing interface it passes through.

4.2 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 nodes 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 fulfil a similar role to Path messages in RSVP.

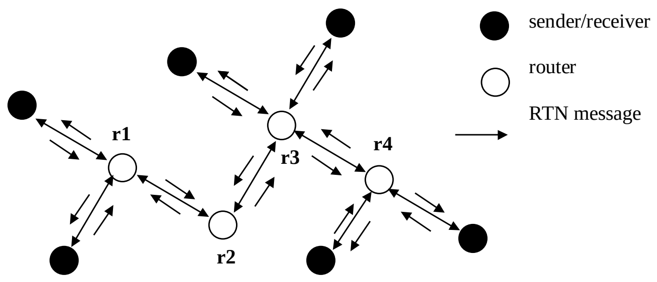


Figure 4.2: DRP RTN Messages on a Shared Tree

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[9] 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 regard to the state and message overhead associated with the supplementary[36] control messages, that is RTN messages in the case of DRP and predominantly Path messages in the case of RSVP. These supplementary control messages facilitate among other things the One Pass With Advertising(OPWA) model[51].

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 4.2. 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 2.4.

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 equal to the number of on-tree outgoing interfaces. 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.

In cases where the number of senders to a group is less than the mean number of outgoing interfaces per on-tree router the amount of flow-specific router-state associated with supplementary

[36] That is control messages which do not themselves directly instantiate the reservations.

control messages produced by DRP is likely to be higher[37] than with RSVP. This can happen 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 the flow-specific state for either RSVP or DRP should still remain at a manageable level.

4.2.1 Shared-Session Reservations

As mentioned in section 2.2.3.1.3, shared-style reservations that are set up 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[38] 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 4.3. Although YESSIR permits sender-initiated explicit reservations, their generation is restricted to inclusion within periodically generated RTCP messages which does not allow the sender application enough timing flexibility to set up and tear down reservations on a per activity burst basis.

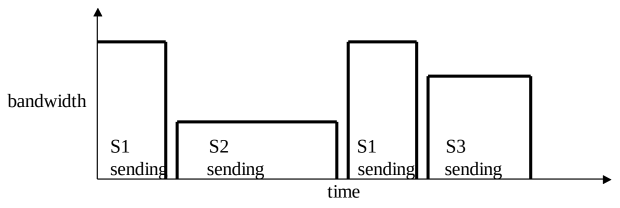
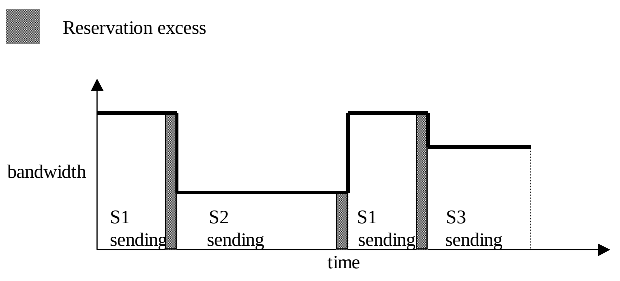


Figure 4.3: Bandwidth of Just-In-Time Sender-Specific Reservations using DRP in the case of a Shared Session with No Sender Transmission Overlap



[37] 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.

Figure 4.4: Bandwidth of a Simple Shared Reservation for a Shared Session with No Sender Transmission Overlap

In using just-in-time sender-specific reservations on a per-activity burst basis to accommodate sender-specific reservations 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 4.4 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. Here the reservation is shared among all senders to the session and is left in place between activity bursts and modified at the start of each 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 4.4, 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 4.6 for the example traffic pattern of Figure 4.5. 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[39] 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.

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

[39] 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.

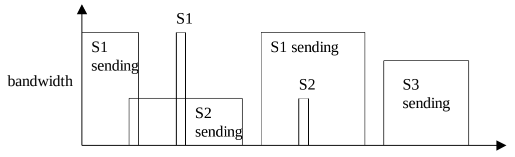


Figure 4.5: Traffic Pattern for a Shared Session with some Sender Transmission Overlap

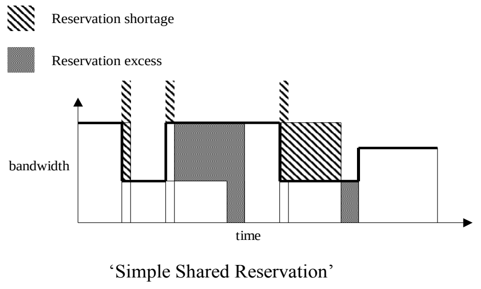


Figure 4.6: Bandwidth Reserved using a 'Simple Shared Reservation' for a Shared Session with some Sender Transmission Overlap

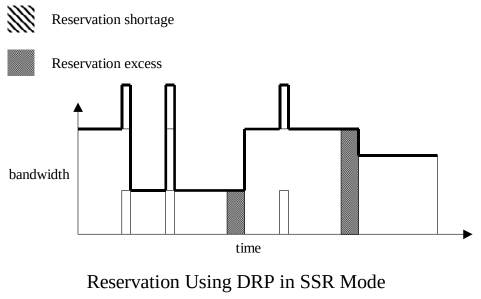


Figure 4.7: Bandwidth Reserved using DRP in SSR Mode for a Shared Session with some Sender Transmission Overlap

Figure 4.7 illustrates the operation of SSR mode for the traffic pattern of Figure 4.5. When only one sender is active at once, the operation of SSR mode is essentially the same as the simple shared reservation of Figure 4.4 and so will suffer from resource wastage when all of the senders go simultaneously quiet[40]. 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 4.4.

[40] 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.

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 DRP accommodates co-existence of reservation modes within the same multicast session 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.
        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)
```

4.3 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.

4.3.1 Explicit vs implicit reservation requests

With RSVP and YESSIR, 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.

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. The approach taken by YESSIR with regard to admission control failure at a node more closely resembles that of DRP[41] than RSVP in that the request is allowed to travel down the tree which results in a partial end-to-end reservation.

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

[41] When DRP is operating in partial reservation mode.

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. In this case 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 [42] but the accumulated bound at a node exceeds the target delay bound for the first time, the action taken is as described in section 4.3.2.

In the case of Guaranteed Service reservations in a multicast tree, there are some interesting differences between DRP's sender-based reservations and RSVP's receiver-based reservations. In particular there are differences in the aggregate bandwidth reservation, that is the sum of the Rspecs associated with each individual reservation. We denote Rsum to equal the ratio of aggregate bandwidth reserved using DRP to that reserved using RSVP. A value of Rsum < 1 therefore indicates that DRP is more bandwidth efficient than RSVP. The value of Rsum for a particular scenario is influenced by, among other things, the existence or otherwise of worst-case merging of DRP RTN messages. In order to define worst-case merging we must first introduce what we call the Total Rate Independent Delay(TRID) which is obtained as follows

$$TRID = Dtot + propdelay \tag{4-1}$$

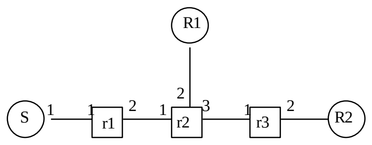
As we shall see in more detail in section 4.6 each RTN message contains both a Dtot and a propdelay field in addition to a so-called Ctot field. To understand the significance of Ctot and Dtot it is first necessary to describe two parameters associated with an output interface of a Guaranteed-Service capable router, namely the so-called C and D parameters[45]. D is the router interface's rate-independent contribution to delay while C/R gives the rate-dependent contribution to delay where R is the service rate of the GS reservation. propDelay is simply the link propagation delay.

As far as DRP is concerned, the Ctot, Dtot and propdelay values in an RTN message equal the aggregation of individual router C terms, D terms and link propagation delays respectively along specific paths. Dtot and propdelay always reflect the same path which may or may not be the same path reflected by Ctot. The RTN merging process, described in 4.6, ensures that when an RTN message in a multicast tree reaches the sender, the value of TRID will reflect the maximum TRID path while the value of Ctot will reflect the maximum Ctot path.

Worst-case merging is then said to have occurred if, when the RTN message reaches the sender, the associated values of TRID and Ctot reflect different paths. In other words worst-case merging will occur if no single (sender, receiver) path dominates in terms of both Ctot and TRID. If no worst-case merging of DRP exists then DRP will always be more bandwidth-efficient than RSVP for installation of Guaranteed Service reservations.

An example of a scenario involving no worst-case merging of DRP RTN messages is given in Figure 4.8 which 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[45] from the routers are shown together with the token bucket parameters of the Sender Tspec. Here the path (S, R2) dominates in terms of both Ctot and TRID and hence worst-case merging of RTN messages does not occur. We will assume that both receivers, R1 and R2 require a queuing delay bound of 200ms 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 740.84Kbytes/s while R2 calculates an Rspec of 991.29 Kbytes/s. At router r2 these two requests are merged so that the Rspec propagated to router r1 is 991.29Kbytes/s. Packets from S to receiver R1 will now experience a reservation bandwidth of 991.29kbyte/s in router r1, interface 2 rather than the requested 740.84 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 740.84 Kbytes/s while still achieving R1's end-to-end delay bound. However RSVP 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 at r2, interface 2 where 497.20 Kbytes/s is then reserved rather than 740.84 Kbyte/s as in RSVP.

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



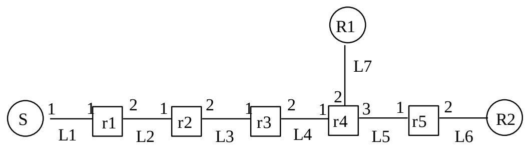
Token bucket parameters	
Peak rate(p) Bytes/s	200K
Token bucket rate(r) Bytes/s	50K
Token bucket depth(b) bytes	6K

interface	Rspec reserved(Kbytes/s)	
	RSVP	DRP
S interface 1	991.29	991.29
r1 interface 2	991.29	991.29
r2 interface 2	740.84	497.20
r2 interface 3	991.29	991.29
r3 interface 2	991.29	991.29
TOTAL	4697.06	4462.36

GS Link Parameters	
C(bytes/s)	48,000
D(μs)	1200
Link propagation delay	0

Rsum=0.95

Figure 4.8: Example of RSVP/DRP reservations with no worst-case merging of DRP RTN messages



Token bucket parameters	
Peak rate(p) Bytes/s	200K
Token bucket rate(r) Bytes/s	50K
Token bucket depth(b) bytes	6K

interface	Rspec reserved(Kbytes/s)	
	RSVP	DRP
S interface 1	2032.56	3040.97
r1 interface 2	2032.56	3040.97
r2 interface 2	2032.56	3040.97
r3 interface 2	2032.56	3040.97
r4 interface 2	2032.56	50
r4 interface 3	1501.56	751.95
r5 interface 2	1501.56	751.95
TOTAL	13165.92	13717.77

GS Link Parameters		
	L1-L6	**L7**
C(bytes/s)	48,000	0
D(μs)	1200	0.1
propDelay	0	0

Rsum=1.04

Figure 4.9: Example of RSVP/DRP reservations with worst-case merging of DRP RTN messages

In the example of Figure 4.9 no single path dominates in terms of both Ctot and TRID. Rather, path (S, R1) dominates in terms of TRID while path (S, R2) dominates in terms of Ctot. Consequently worst-case merging of DRP RTN messages exists and as a result nodes S, r1, r2 and r3 over-estimate the values of their local reservations. Each of these 'over-reservations' causes a reduction in the local node queuing delay compared to the RSVP case. In turn this means that DRP allows an increase in the local queuing delay at nodes downstream of r3 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 effect is illustrated in Figure 4.9 where Rsum=1.04.

4.3.2 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[51] into the reservation flow specification. Use of the slack term in the case of RSVP 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 4.10, 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, del in its local queueing delay 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.

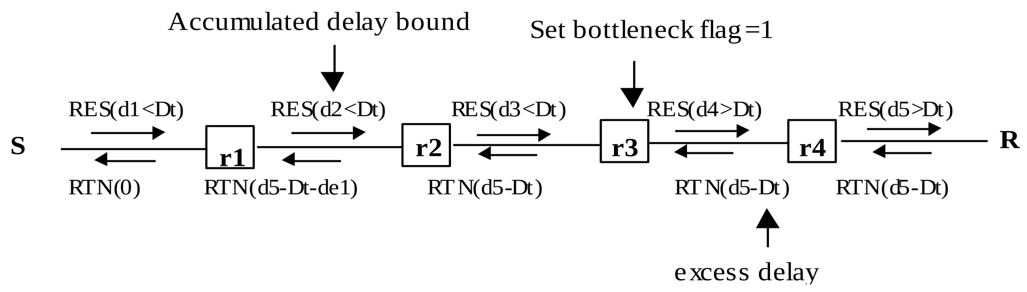


Figure 4.10: Use of DRP feedback Guaranteed Service reservation mechanism to maximise chances of meeting target delay bound

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.

4.4 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[33] is used to ensure that the RES message is intercepted and processed by intermediate nodes.

4.4.1 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 4.7.

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 [45].

- p = peak rate of flow (bytes/second)
- b = bucket depth (bytes)
- r = token bucket rate (bytes/second)
- m = minimum policed unit (bytes)[43]
- 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.

4.4.2 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 [45]).

DSum - accumulation of D values since last upstream reshaping point (see [45]).

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 zero queuing loss cannot be guaranteed)

4.5 Node Processing of RES messages

The primary purpose of the RES message is to install and refresh reservations in the distribution tree. It also results in installation of additional state such as Phop state which is used to correctly reverse route RTN messages. In addition it is involved in the process by which link delay is determined as detailed in section 4.7.

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 4.3. 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 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 [45].

$$Qdelay_{end2end} = \frac{(b-M)(p-R)}{R(p-r)} + \frac{(M+Ctot)}{R} + Dtot \quad \text{(case p > R} \geq \text{r)} \qquad \text{(4-2)}$$

$$Qdelay_{end2end} = \frac{(M+Ctot)}{R} + Dtot \quad \text{(case R} \geq \text{p} \geq \text{r)} \qquad \text{(4-3)}$$

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

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 (4-4)

Qdelay = target-bound - accumulated-bound – dnext – propdelay (4-4)

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 (4-2) and (4-3) along with the other mentioned parameters, a value of service bandwidth R to be installed at the outgoing interface is obtained. Once the service bandwidth of the reservation has been modified as necessary to reflect the calculated value of R, the RES message is forwarded down the distribution tree after updating the appropriate fields in the packet's 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 2 updates must be made to the RES packet:

1) 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 (4-5) along with the local values of C and D.

$$Qlocal = \frac{C}{R} + D \qquad (4\text{-}5)$$

2) If reshaping to the sender Tspec is being performed at the outgoing interface
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.

4.6 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.

4.6.1 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 4.7.

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 4.7.

propdelay field. This equals the data packet propagation delay along the 'worst-case rate-independent delay path'[45] between the node incoming[44] interface out of which the RTN packet was sent and each receiver downstream of that incoming interface.

[44] 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.

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[45].

Dtot This equals the maximum accumulated Dtot value along the 'worst-case rate-independent delay path'[45] 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[45].

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.

4.6.2 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 interface). The first three parameters of this 4-tuple are contained within the received RTN message while the outgoing interface(oif) is

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

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 4-2. In addition, for each outgoing interface, i a single Merged RTN State Entry (MRTNSE_{i}) is created from the set of entries {RTNSE_{ij}} for that outgoing interface. There will be multiple RTNSE_{ij}s for a given outgoing interface if the outgoing interface has multiple next hops on the distribution tree which can occur if the outgoing interface connects to a shared medium LAN(e.g. Ethernet). The parameters of the MRTNSE_{i} and how they are formed from {RTNSE_{ij}} are also shown in Table 4-2. The 'merged values' of various parameters in each RTN message sent out of an incoming interface to a previous hop upstream are obtained from {MRTNSE_{i}}, the set of MRTNSE for the outgoing interfaces as shown in Table 4-2.

The last three rows of Table 4-2 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, $\text{delayReduction}_{si}$ 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 5.2 are overridden by setting Ctot=Dtot=propdelay=0 in order to ensure that only those links 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.

$RTNSE_{ij}$	$MRTNSE_{I}$	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 [46])
$Dtot_{ij}$	$Dtot_i=Dtot_{ij}$ 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[46]
$Propdelay_{ij}$	$Propdelay_i=propdelay_{ij}$ 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[46]
$PathBandwidth_{ij}$	$PathBandwidth_i=$ $\max\{PathBandwidth_{ij}\}$	$PathBandwidth_k=$ $\max\{\min(pathbandwidth_i, \text{link rate}_i)\}$
$PathMTU_{ij}$	$PathMTU_i= \min\{pathMTU_{ij}\}$	$PathMTU_k=$ $\min\{\min(\text{path } MTU_i, linkMTU_i)\}$
$dnext_{ij}$	$dnext_i=dnext_{ij}$ 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*$_s$
$excessDelay_{sij}$	$excessDelay_{si}=$ $\max\{excessDelay_{sij}\}$	$excessDelay_s=$ $\max\{excessDelay_{si}–delayReduction_{si}\}$
$bottleneckFlag_{sij}$	$BottleneckFlag_{si}=$ $\max\{bottleneckFlag_{sij}\}$	$\text{bottleneck } flag_s=\max\{\text{bottleneck } flag_{si}\}$

Table 4-2: relationship between RTN state entries, MRTN state entries and merged RTN packets.

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

4.7 Determining link delay

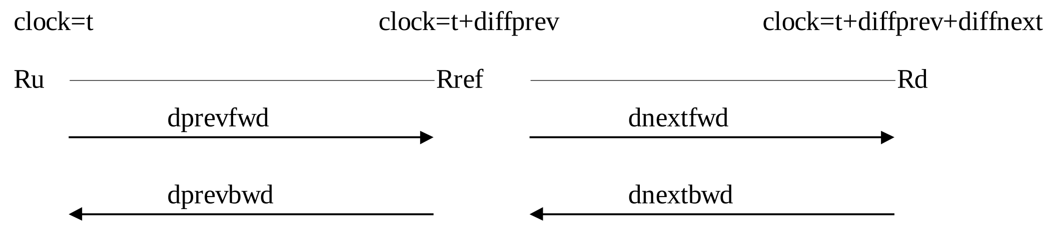


Figure 4.11: Reference Model for Determining Link Delay

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 4-3: Description of Variables for Calculation of Link Delay

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

4.7.1 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

$$tarrival - tstamp = dprevfwd + diffprev = timedeltaprev \qquad (4\text{-}6)$$

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.

4.7.2 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:

$$timedelta = dnextfwd + diffnext \qquad (4\text{-}7)$$

$$tarrival - tstamp = dnextbwd - diffnext \qquad (4\text{-}8)$$

Adding (4-7) and (4-8) gives

$$dnextfwd + dnextbwd = timedelta + tarrival - tstamp \qquad (4\text{-}9)$$

If it is known that the link delay is symmetrical then equation (4-9) simplifies to

$$dnextfwd = (timedelta + tarrival - tstamp)/2 \qquad (4\text{-}10)$$

Otherwise we have to assume the worst case for dnextfwd which gives

$$dnextfwd = timedelta + tarrival - tstamp \qquad (4\text{-}11)$$

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.

4.8 Simulation

In this section we compare the efficiency of DRP to that of RSVP and YESSIR with regard to installation and modification of Guaranteed Service reservations in multicast scenarios. Specifically we measure the aggregate reservation bandwidth, denoted Rsum, installed in each case. Rsum is given by the summation of the service bandwidths of the individual reservations in the network. In addition we also measure the amount of reservation messages generated whenever there is a session-wide change in the Guaranteed Service QoS requirement.

In our simulations we assume two different combinations of link layers for the LAN, MAN and WAN parts of an internet as shown in Table 4-4 where B is the link bandwidth and C, D and propDelay are the Guaranteed Service(GS) parameters advertised by network elements. These GS parameters were briefly described in section 4.3.1.

For switched ethernet and PPP/SDH both the C term and D term are fixed, the former being equal to the maximum sized datagram and the latter equal to the time required to place a maximum sized datagram onto the output port. For ATM links we assume that network elements will advertise propDelay=0 and C=0 while lumping the real delay contribution of these two parameters into the Dterm as suggested in [22].

For PPP/SDH links we use a truncated normal distribution to generate random values for propDelay. We obtain the standard deviation, σ and mean, μ of the gaussian distribution used for this purpose from the following two equations

σ = (propDelayMax-propDelTyp)/3 (4-12)

μ = propDelayTyp (4-13)

propDelayMax is the propagation delay for what our simulation assumes to be the longest length of a WAN fibre optic link, namely 5000km which is the approximate distance between London and New York. The probability of a randomly generated deviate described by the gaussian distribution of equations (4-12) and (4-13) being greater than propDelayMax is less than 0.3% which we believe is representative of the real-world. PropDelTyp is the propagation delay of what we consider to be a typical physical length of a WAN fibre link, namely 50km.

Each PPP/SDH link, i is assigned a separate random value of propDelay as follows. First a random gaussian deviate, d_i is generated according to the parameters in equations (4-12) and (4-13). Next equation (4-14) is applied to the deviate in order to limit its value to the range (propDelMin, propDelMax).

d_i = min(propDelMax, (max(propDelMin, d_i)) (4-14)

where propDelMin is the propagation delay for what our simulation assumes to be the shortest length of a fibre optic WAN link, namely 100m.

For the D term advertised by routers on an ATM network we use a truncated gaussian distribution described by the parameters Dmin, Dmean and Dmax where Dmax is 3 standard deviations away from Dmean. The values that we chose were influenced by service offerings of ATM network providers together with some simple delay measurements of our own across ATM WAN links.

Link Layer Mix	LAN	MAN	WAN

1	Switched Ethernet
	B = 100Mbps
	C = 1500 bytes
	D = 120us
	PropDelay = 0
		ATM/SDH
		B = 155Mbps
		C = 0
		Dmin = 100us
		Dmean = 0.5ms
		Dmax = 5ms
		propDelay = 0
			ATM/SDH
			B = 155Mbps
			C = 0
			Dmin = 100us
			Dmean = 5ms
			Dmax = 50ms
			propDelay = 0

2	Switched Ethernet
	B = 100Mbps
	C = 1500 bytes
	D = 120us
	PropDelay = 0
		ATM/SDH
		B = 155Mbps
		C = 0
		Dmin = 100us
		Dmean = 0.5ms
		Dmax = 5ms
		propDelay = 0
			PPP/SDH
			B = 155Mbps
			C = 1500 bytes
			D = 77us
			PropDelMin = 0.5us(100m)
			PropDelTyp = 250us (50km)
			PropDelMax = 25ms(5000km)

Table 4-4: The Two Different Kinds of Link Layer Mixes used in the Simulations

All simulations were done using ns[2] with a 3 tier network topology(WAN, MAN, LAN) generated by the tiers topology generator[1] with no network redundancy.

4.9 Example Scenario

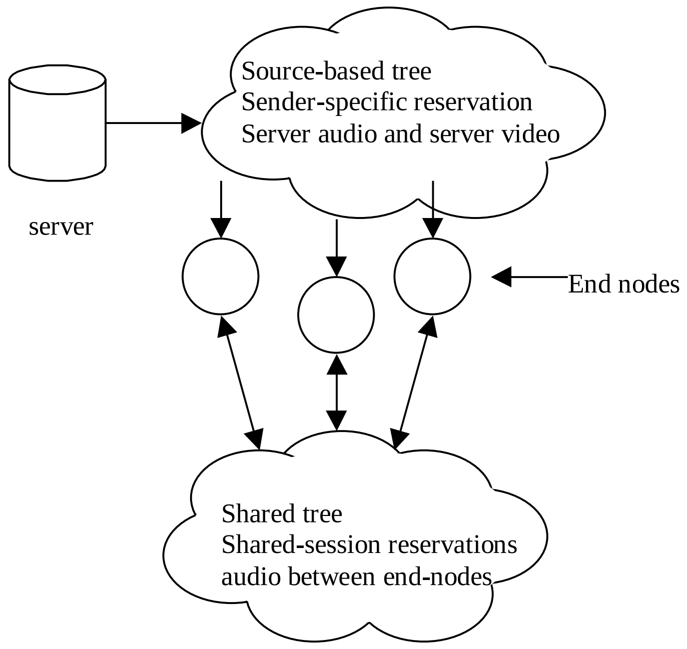

Source-based tree
Sender-specific reservation
Server audio and server video
server
End nodes
Shared tree
Shared-session reservations
audio between end-nodes

Figure 4.12: Scenario to be simulated

Figure 4.12 shows an example scenario to which the simulations are applicable. Here a group of end-nodes are playing out real-time video and associated audio, both received from the server over source-based multicast trees. Each end-node requires Guaranteed Service delivery of the media components from the server and each reservation installed in the source-based tree will be sender-specific. In addition end-nodes wish to communicate with each other[47] in real-time using audio multicast over a shared tree and delivered with QoS according to the Guaranteed Service reservation model.

In the rest of this section we present simulation results for firstly, sender-specific reservations between a server and a group of end-nodes, and secondly, shared-session reservations between a group of end-nodes. Figure 4.12 is included as an example of where both sender-specific and shared-reservation Guaranteed Service reservations may be used within the context of on-demand multicast services. We do not discuss the manner in which setup of the sessions in Figure 4.12 is coordinated as this is outside the scope of this thesis.

[47] Perhaps to discuss the video

4.10 Traffic Sources

4.10.1 Traffic Source used by Sender-Specific Simulation

Here a single end node is selected at random to act as the sender while all other end-nodes are receivers and wish to receive traffic using Guaranteed Service. Table 4-5 shows the Tspec parameters of the traffic source together with the target end to end delay bound required. In this case the traffic source is intended to represent MPEG video. The values of peak and mean (token) traffic rate are those of a commercial MPEG traffic source[65]. To derive a sensible value for b we define the parameter, Tpeak, which, assuming the token bucket is initially full, is the duration for which the traffic source can transmit at the peak rate before queuing delay is introduced.

$$T_{peak} = \frac{b}{(p - r)} \tag{4-15}$$

With the given values of p and r, the value of b was chosen such that Tpeak = 0.33 seconds which we believe to be a reasonable[48] value for MPEG.

peak rate (p)	200, 000 bytes/s
token rate (r)	80, 000 bytes/s
token bucket depth (b)	40,000 bytes
max datagram size (M)	1500 bytes
Target delay	0.2 s

Table 4-5: Tspec and target delay parameters for codec used in sender-specific reservation simulations.

4.10.2 Traffic Sources used by Shared-Session Simulation

In our shared-session simulation we model an audio-conference based on the following assumptions.

- Each end-node has both send and receive capability.
- Each end-node wants to receive traffic from all other nodes using Guaranteed Service.
- Media distribution is done via a single bi-directional shared tree.
- Exactly half of the traffic sources are chosen at random to be codec A while the remainder are codec B.

[48] Only occasional MPEG frames representing a scene change and containing little intra-frame redundancy would approach this peak rate.

Parameters	Codec A	Codec B
peak rate (p)	9, 000 bytes/s	2, 000 bytes/s
token rate (r)	3, 600 bytes/s	800 bytes/s
token bucket depth (b)	8,100 bytes	1800 bytes
max datagram size (M)	360 bytes	80 bytes
Target delay	0.2 s	0.2 s

Table 4-6: Tspec and Target Delay Parameters for Codecs Used in Shared-Session Reservation Simulations

Table 4-6 shows the parameters for the 2 codecs. CodecA is equivalent to a G.711 codec while codec B is equivalent to a G.729 codec. Figure 4.13 and Figure 4.14 illustrate how we calculated each of the parameters in the Tspec. The maximum payload sizes are chosen to limit the packetisation delay to 40ms. Adding the packet headers to these maximum payload sizes gives the maximum datagram size. The peak rate can be calculated by considering the rate of payload fill by the voice and the total packet size. The mean packet rates are obtained assuming silence suppression is used and that the voice activity rate is 40% which is a typical value obtained from [44]. The values of b chosen give a value of Tpeak equal to 1.5s which we believe to be a reasonable[49] value for a voice source.

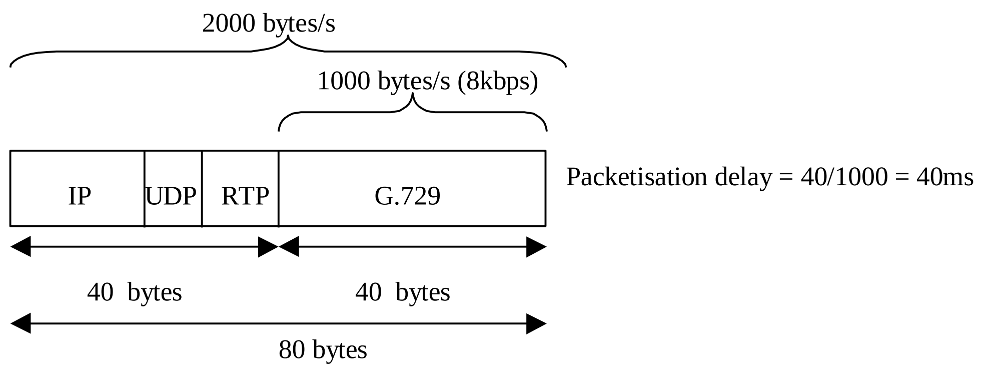


Figure 4.13: Packet Parameters for G.729 voice

[49] A voice source with silence suppression only ever transmits at 2 rates, namely the peak rate during an activity burst and 0 in between bursts. For a voice source, Tpeak equates to the maximum burst duration before queuing delay is introduced, assuming the token bucket was full at the start of the burst.

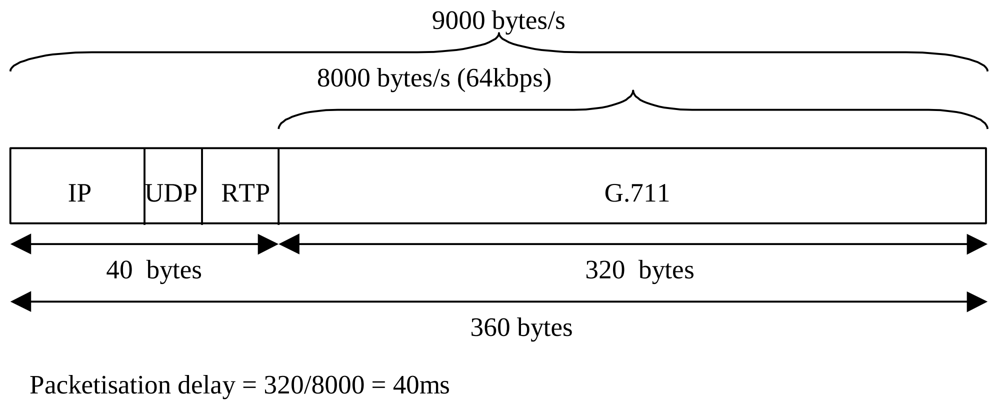


Figure 4.14: Packet Parameters for G.711 voice

4.11 Sender-Specific Reservation Simulations - Results

Sections 4.11.1 to 4.11.4 contain simulation results for a single sender in both a small and a large network as defined in Table 4-7 for different link layer mixes as defined in Table 4-4. Each plotted point represents the mean of 10 simulation runs, each with a different random seed value for tiers topology generation.

In representation of mean values obtained from experiment or simulation it is quite common to use error bars to show 90, 95 or 99% confidence intervals obtained using the t-distribution on the assumption that the values used to obtain the mean approximate a gaussian distribution. For many of our simulations however such an approach would be highly suspect since the distributions are clearly not gaussian. For example in Figure 4.15 the value of Rsum of a given simulation can never be greater than 1 since this is an example of no-worst-case merging of DRP as was explained in 4.3.1. If we assumed a gaussian distribution of the readings used to obtain a plotted point and used the t-distribution to obtain confidence intervals then for some of the points this would result in an upper error bar which extended above 1 which is clearly wrong. Consequently in this section we choose not to use error bars that attempt to represent confidence intervals. Instead the error bars around each plotted point represent the minimum and maximum of the 10 simulations used to obtain the plotted point(i.e. the mean).

	WAN	MAN	LANs per MAN	LAN hops between each end node and MAN
Small network	1 x 8 node WAN	4 nodes per MAN	1	2
Large network	1 x 30 node WAN	10 nodes per MAN	1	2

Table 4-7: Parameters for Small and Large Network Topologies used in Simulations

4.11.1 Small network – link layer mix 1 – sender specific reservations

Title:
/cs/research/nets/home/marine/ucacppw/eps/tot_R_DRP%RSVP_1_8_4_FF_A.eps
Creator:
gnuplot 3.7 patchlevel 0
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

Figure 4.15: DRP Rsum(all nodes) Normalised to RSVP

Title:
/cs/research/nets/home/marine/ucacppw/eps/wan_R_DRP%RSVP_1_8_4_FF_A.eps
Creator:
gnuplot 3.7 patchlevel 0
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

Figure 4.16: DRP Rsum(WAN nodes only) Normalised to RSVP

Title:
/cs/research/nets/home/marine/ucacppw/eps/tot_yessir%RSVP_1_8_4_FF_A.eps
Creator:
gnuplot 3.7 patchlevel 0
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

Figure 4.17: YESSIR Rsum(all nodes) Normalised to RSVP

Title:
/cs/research/nets/home/marine/ucacppw/eps/wan_yessir%RSVP_1_8_4_FF_A.eps
Creator:
gnuplot 3.7 patchlevel 0
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

Figure 4.18: YESSIR Rsum (WAN nodes only) Normalised to RSVP

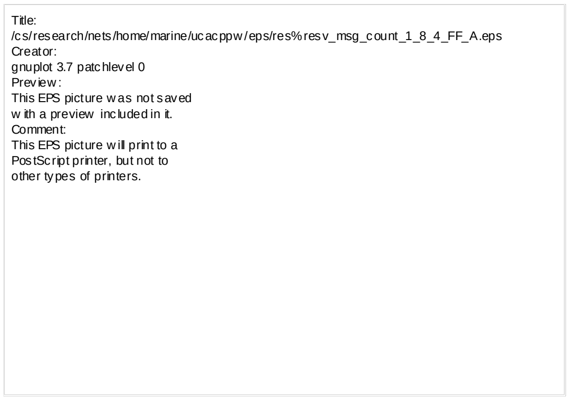

Title:
/cs/research/nets/home/marine/ucacppw/eps/res%resv_msg_count_1_8_4_FF_A.eps
Creator:
gnuplot 3.7 patchlevel 0
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

Figure 4.19: Message Ratio: DRP(RES)/RSVP(ResV)

For the small network sender-specific simulations with link layer mix 1, the value of Rsum for DRP relative to RSVP is always less than 1 (Figure 4.15 and Figure 4.16). This is consistent with the theory outlined in 4.3.1 in that any worst-case merging efficiencies of DRP are only ever introduced if the paths to different receivers are variable in terms of both the Ctot and TRID. For a type 1 link layer mix only the LANs contribute to the value of Ctot and this contribution will be fixed at 600 bytes (4 LAN hops – 2 each end) for each path. In other words the paths to each receiver will be variable only in terms of Dtot and not Ctot. Consequently worst-case merging inefficiencies will not be introduced by DRP in this particular case and as a result the value of Rsum will be less for DRP than RSVP. However this mean aggregate reservation bandwidth saving is insignificant and across the simulations covered by both Figure 4.15(all nodes) and Figure 4.16(WAN nodes only) is between 0.05 % and 0.20 %.

In Figure 4.17(all nodes) and Figure 4.18(WAN nodes only) YESSIR yields a higher value of Rsum than RSVP due to the worst case merging effects at the sender that were explained in section 2.2.5.1, although the mean increase is quite low at between 1.3 % and 4.9 % across the range of simulations.

Figure 4.19 shows, for all network links, the amount of DRP RES messages required to initiate a session-wide change in Guaranteed Service QoS compared to the amount of RSVP ResV messages. Across the range of simulations, the volume of DRP messages was on average between 31% and 43% less than that of RSVP. This finding corroborates the discussion in section 2.2.3.1.6.

4.11.2 Small network – link layer mix 2 – sender specific reservations

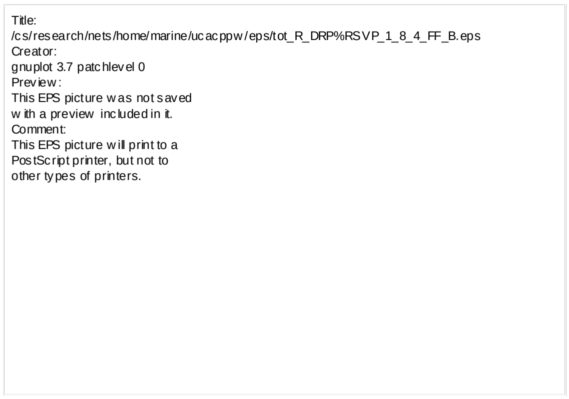

Title:
/cs/research/nets/home/marine/ucacppw/eps/tot_R_DRP%RSVP_1_8_4_FF_B.eps
Creator:
gnuplot 3.7 patchlevel 0
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

Figure 4.20: DRP Rsum(all nodes) Normalised to RSVP, Sender-Specific

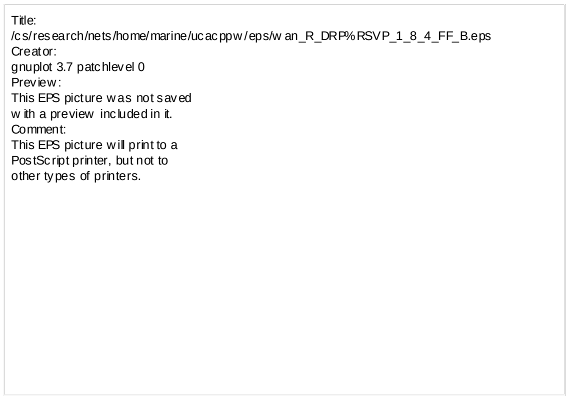

Title:
/cs/research/nets/home/marine/ucacppw/eps/wan_R_DRP%RSVP_1_8_4_FF_B.eps
Creator:
gnuplot 3.7 patchlevel 0
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

Figure 4.21: DRP Rsum(WAN nodes only) Normalised to RSVP, Sender-Specific

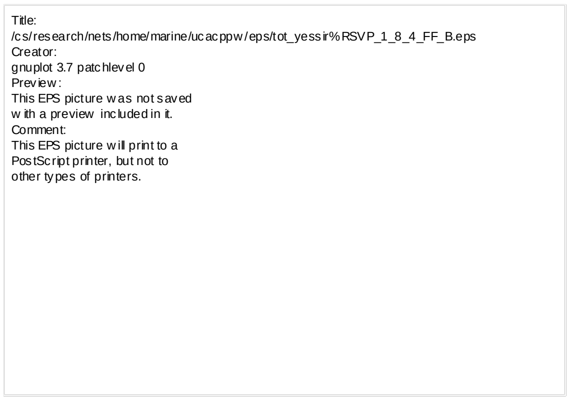

Title:
/cs/research/nets/home/marine/ucacppw/eps/tot_yessir%RSVP_1_8_4_FF_B.eps
Creator:
gnuplot 3.7 patchlevel 0
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

Figure 4.22: YESSIR Rsum(all nodes) normalised to RSVP, Sender-Specific

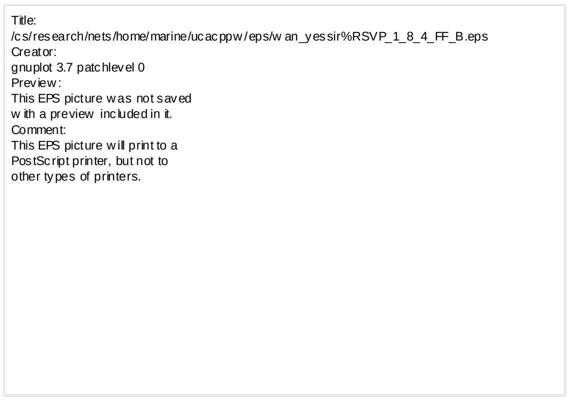

Title:
/cs/research/nets/home/marine/ucacppw/eps/wan_yessir%RSVP_1_8_4_FF_B.eps
Creator:
gnuplot 3.7 patchlevel 0
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

Figure 4.23: YESSIR RSum(WAN nodes only) Normalised to RSVP, Sender-Specific

For the small network sender-specific simulations with link layer mix 2, in Figure 4.20(all nodes) and Figure 4.21(WAN nodes only) DRP gives a mean value of Rsum that is between 0.08 % and 0.42 % less than RSVP across the range of simulations. As expected, in Figure 4.22(all nodes) and Figure 4.23(WAN nodes only), YESSIR gives a higher mean value of Rsum than RSVP although the mean increase is low at between 1.7 % and 6.1 % across the range of simulations.

4.11.3 Large Network - link layer mix 1 – sender specific reservations

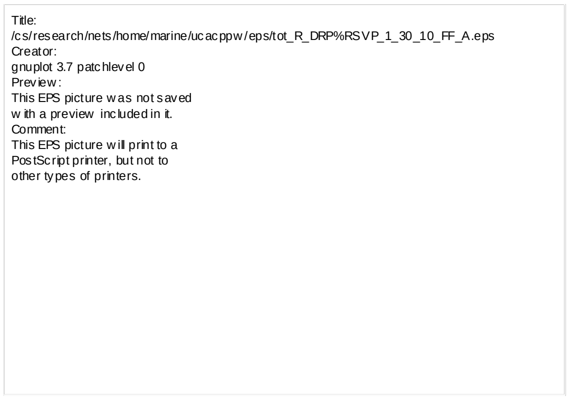

Title:
/cs/research/nets/home/marine/ucacppw/eps/tot_R_DRP%RSVP_1_30_10_FF_A.eps
Creator:
gnuplot 3.7 patchlevel 0
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

Figure 4.24: DRP Rsum(all nodes) Normalised to RSVP, Sender-Specific

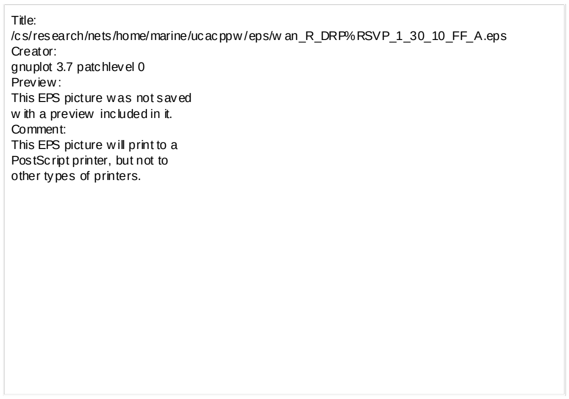

Title:
/cs/research/nets/home/marine/ucacppw/eps/wan_R_DRP%RSVP_1_30_10_FF_A.eps
Creator:
gnuplot 3.7 patchlevel 0
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

Figure 4.25: DRP Rsum(WAN nodes only) Normalised to RSVP, Sender-Specific

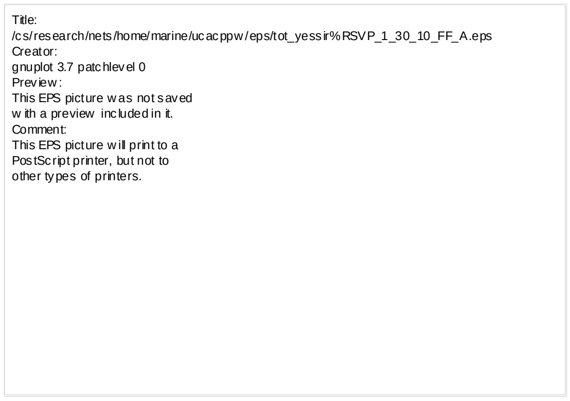

Title:
/cs/research/nets/home/marine/ucacppw/eps/tot_yessir%RSVP_1_30_10_FF_A.eps
Creator:
gnuplot 3.7 patchlevel 0
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

Figure 4.26: YESSIR RSum(all nodes) Normalised to RSVP, Sender-Specific

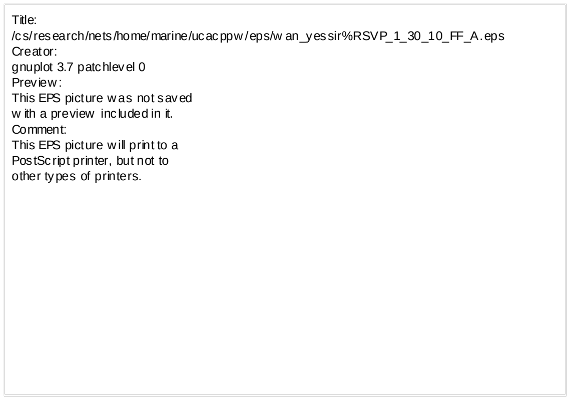

Title:
/cs/research/nets/home/marine/ucacppw/eps/wan_yessir%RSVP_1_30_10_FF_A.eps
Creator:
gnuplot 3.7 patchlevel 0
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

Figure 4.27: YESSIR Rsum(WAN nodes only) Normalised to RSVP, Sender-Specific

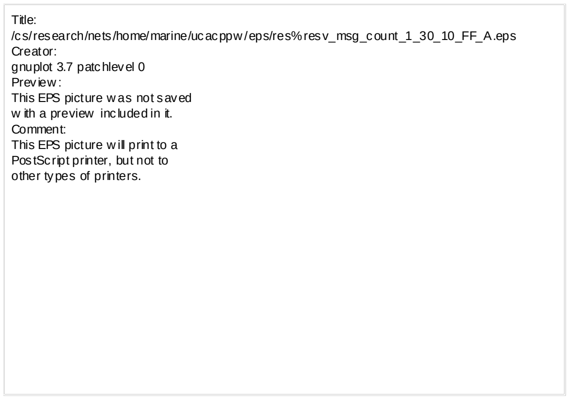

Title:
/cs/research/nets/home/marine/ucacppw/eps/res%resv_msg_count_1_30_10_FF_A.eps
Creator:
gnuplot 3.7 patchlevel 0
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

Figure 4.28: Message Ratio: DRP(RES)/RSVP(ResV), Sender-Specific

4.11.4 Large Network – link layer mix 2 – sender specific simulations

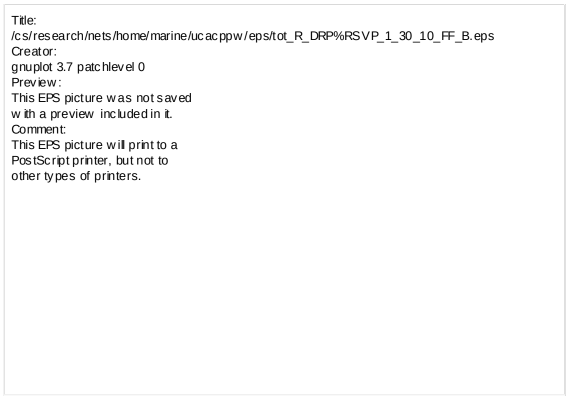

Title:
/cs/research/nets/home/marine/ucacppw/eps/tot_R_DRP%RSVP_1_30_10_FF_B.eps
Creator:
gnuplot 3.7 patchlevel 0
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

Figure 4.29: DRP Rsum(all nodes) Normalised to RSVP, Sender-Specific

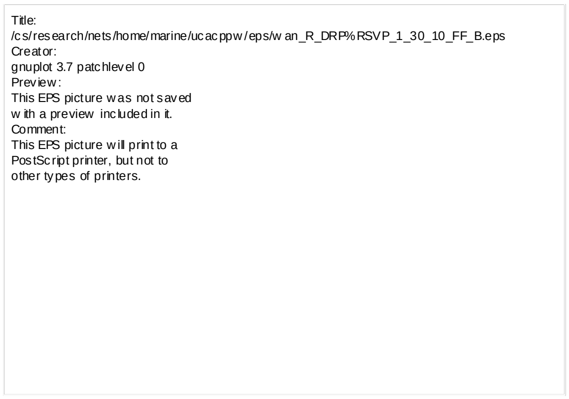

Title:
/cs/research/nets/home/marine/ucacppw/eps/wan_R_DRP%RSVP_1_30_10_FF_B.eps
Creator:
gnuplot 3.7 patchlevel 0
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

Figure 4.30: DRP Rsum(WAN nodes only) Normalised to RSVP, Sender-Specific

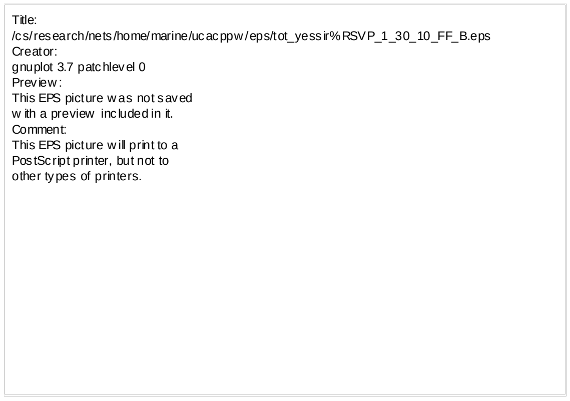

Title:
/cs/research/nets/home/marine/ucacppw/eps/tot_yessir%RSVP_1_30_10_FF_B.eps
Creator:
gnuplot 3.7 patchlevel 0
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

Figure 4.31: YESSIR RSum(all nodes) Normalised to RSVP, Sender-Specific

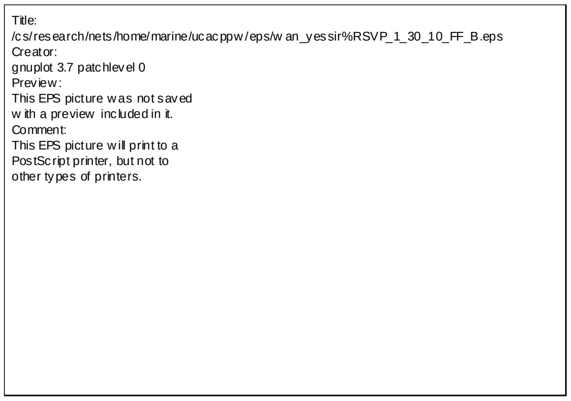

Title:
/cs/research/nets/home/marine/ucacppw/eps/wan_yessir%RSVP_1_30_10_FF_B.eps
Creator:
gnuplot 3.7 patchlevel 0
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

Figure 4.32: YESSIR Rsum(WAN nodes only) Normalised to RSVP, Sender-Specific

Like the small network simulations in sections 4.11.1 and 4.11.2, in the large network simulation results of sections 4.11.3 and 4.11.4, DRP and YESSIR yield lower and higher values of Rsum respectively compared to RSVP. In these larger network cases Rsum for DRP is lower than in the smaller network cases although the aggregate bandwidth saving compared to using RSVP is still small.

For link layer mix 1, for the large network as a whole Figure 4.24 shows that DRP gives a mean value of RSum between 0.11 % and 0.43 % less than RSVP across the range of simulations. Figure 4.25 shows that for the WAN alone DRP gives a mean value of Rsum between 0.07 % and 0.36 % less than RSVP across the range of simulations. In the YESSIR results for the large network as a whole in Figure 4.26, YESSIR gives a mean value of Rsum between 2.8 % and 17 % greater than RSVP across the range of simulations. For the WAN-only nodes in Figure 4.27, YESSIR gives a mean value of Rsum between 1.7 % and 11 % greater than RSVP across the range of simulations. In Figure 4.28, for all network links, the mean volume of control messages produced by DRP to initiate a session-wide change in Guaranteed Service QoS is between 34 % and 46% less than that of RSVP across the range of simulations.

For link layer mix 2, Figure 4.29 shows that for the large network as a whole DRP gives a mean value of Rsum between 0.46 % and 2.2 % less than RSVP. In addition Figure 4.30 shows that for the WAN alone DRP gives a value of Rsum between 0.18 % and 1.3 % less than RSVP. In the YESSIR results for the large network as a whole in Figure 4.31, YESSIR gives a mean value of Rsum between 4.4 % and 17 % greater than RSVP across the range of simulations. For the WAN-only nodes in Figure 4.32, YESSIR gives a value of Rsum between 2.2 % and 11 % greater than RSVP across the range of simulations.

4.12 Shared Session Reservation Simulations

Here we consider DRP operating in two different modes, Sender-Specific(SS) and Sender-Specific with Residue(SSR).

In the SS case the number of active reservations installed and consequently the value of Rsum at any moment in time depends on which sender is currently transmitting as exemplified in Figure 4.33a) for S1 transmitting. In order to gain a meaningful measure of Rsum for the group as a whole we calculate Rsum for each sender in turn and take the mean.

In the SSR case the value of Rsum at any moment in time is given by the sum of the active reservations installed by the current sender in addition to any residual passive reservations that exist in the network as exemplified in Figure 4.33b). The value of the residual passive reservations will depend upon which sequence of senders were recently active prior to the current sender. In our simulation we assume the worse case scenario whereby the sequence of recent senders is such that the passive reservations while the current sender is active are at their maximum possible values. Based on this assumption, the mean value of Rsum used for comparing DRP in SSR mode to other protocols is given by the algorithm shown in Figure 4.34.

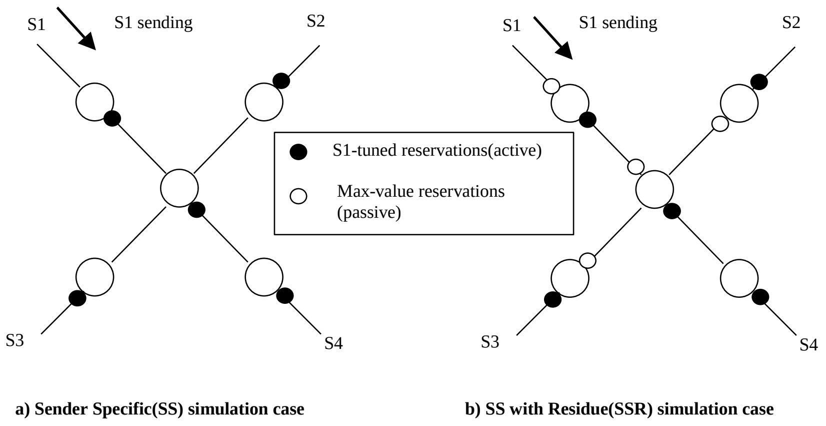

S1
S1 sending
S2
S1-tuned reservations(active)
Max-value reservations (passive)
S3
S4
a) Sender Specific(SS) simulation case
S1
S1 sending
S2
S3
S4
b) SS with Residue(SSR) simulation case

Figure 4.33: Example of Contribution to Rsum for SS and SSR Simulation Cases.

```
For the set of on-tree router outgoing interfaces {$oif_i$}
        $Rmax_i \leftarrow 0$
For each sender {
        Generate a RES message
        At each on-tree router oif, i that the RES passes through {
                Calc and reserve service bandwidth, $R_i$
                $Rmax_i \leftarrow max(R_i, Rmax_i)$
        }
        Generate a RES teardown message
}
Rtotal=0
For each sender {
        Generate a RES message
        At each on-tree router oif, i that the RES passes through {
                Calc and reserve service bandwidth, $R_i$
        }
        For each on-tree oif, i {
                If Ri exists
                        $Rtotal=Rtotal+R_i$
                else
                        $Rtotal=Rtotal+Rmax_i$
        }
        Generate a RES teardown message
}
Rsum=Rtotal/(Number of senders)
```

Figure 4.34: Algorithm for Calculating Rsum for Simulations of DRP in SSR Mode

4.13 Shared Session Reservation Simulations – Results

As for the sender-specific simulation results, each plotted point represents the mean of 10 simulations while the associated error bars represent the minimum and maximum values of the 10 simulations.

4.13.1 Small network – link layer mix 2 – shared reservations

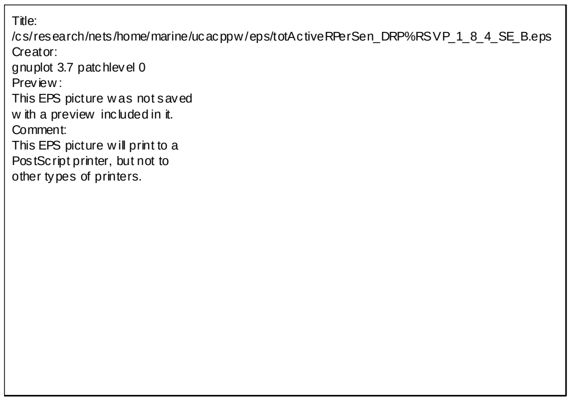

Title:
/cs/research/nets/home/marine/ucacppw/eps/totActiveRPerSen_DRP%RSVP_1_8_4_SE_B.eps
Creator:
gnuplot 3.7 patchlevel 0
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

Figure 4.35: Rsum(all nodes) for DRP in SS mode normalised to RSVP case, shared session

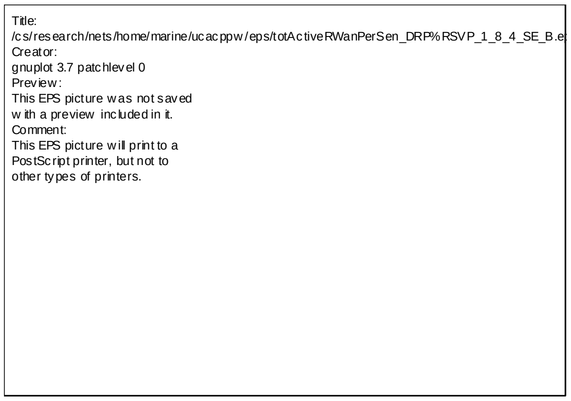

Title:
/cs/research/nets/home/marine/ucacppw/eps/totActiveRWanPerSen_DRP%RSVP_1_8_4_SE_B.e
Creator:
gnuplot 3.7 patchlevel 0
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

Figure 4.36: Rsum(WAN nodes only) for DRP in SS mode normalised to RSVP case, shared session

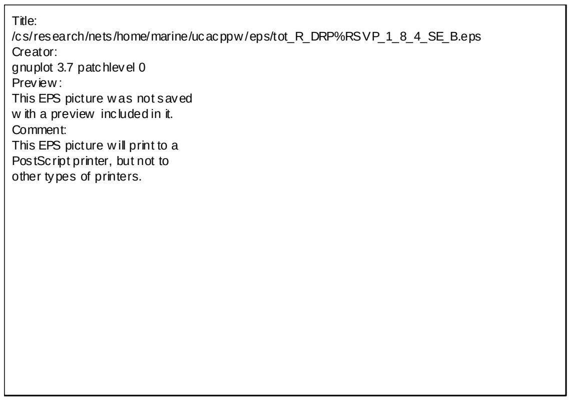

Title:
/cs/research/nets/home/marine/ucacppw/eps/tot_R_DRP%RSVP_1_8_4_SE_B.eps
Creator:
gnuplot 3.7 patchlevel 0
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

Figure 4.37: Rsum(all nodes) for DRP in SSR mode normalised to RSVP case, shared session

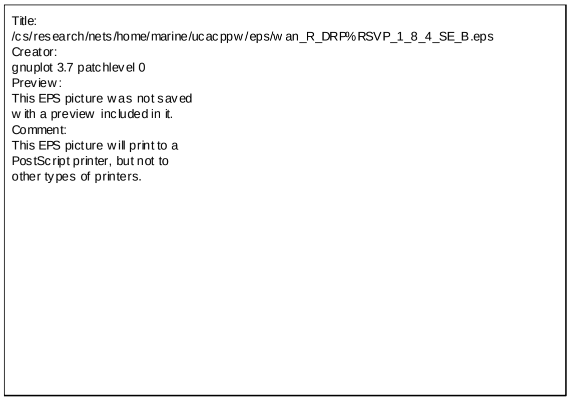

Title:
/cs/research/nets/home/marine/ucacppw/eps/wan_R_DRP%RSVP_1_8_4_SE_B.eps
Creator:
gnuplot 3.7 patchlevel 0
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

Figure 4.38: Rsum(WAN nodes only) for DRP in SSR mode normalised to RSVP case, shared session

Figure 4.35 to Figure 4.38 illustrate the significant aggregate bandwidth savings that are attainable using DRP compared to RSVP for shared-session multicast scenarios with heterogeneous sender traffic characteristics. The bandwidth savings are especially high when the number of hosts is high.

For DRP in SS mode, Figure 4.35 shows that for the small network as a whole DRP has a mean value of Rsum between 58 % and 72 % less than that of RSVP across the range of simulations. For DRP in SS mode, Figure 4.36 shows that for the WAN alone DRP has a mean value of Rsum between 50 % and 56 % less than that of RSVP across the range of simulations

For DRP in SSR mode, Figure 4.37 shows that for the small network as a whole DRP has a mean value of Rsum between 12 % and 30 % less than that of RSVP across the range of simulations. For DRP in SSR mode, Figure 4.38 shows that for the WAN alone DRP has a mean value of Rsum between 10% and 35 % less than that of RSVP across the range of simulations

4.13.2 Large network – link layer mix 2

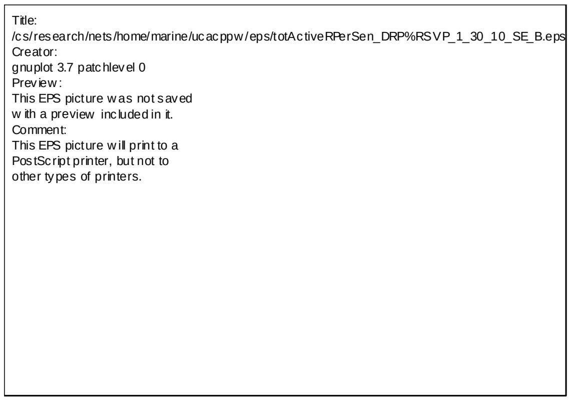

Title:
/cs/research/nets/home/marine/ucacppw/eps/totActiveRPerSen_DRP%RSVP_1_30_10_SE_B.eps
Creator:
gnuplot 3.7 patchlevel 0
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

Figure 4.39: Rsum(all nodes) for DRP in SS mode normalised to RSVP case, shared session

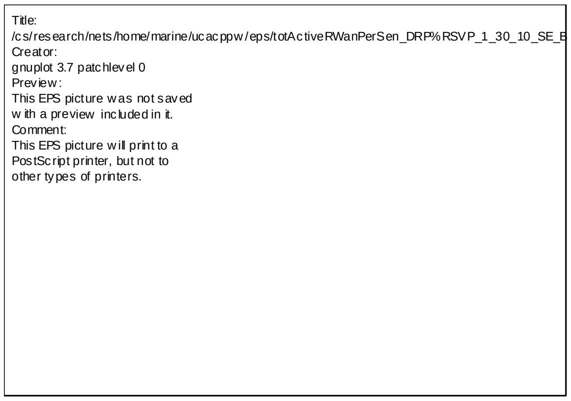

Title:
/cs/research/nets/home/marine/ucacppw/eps/totActiveRWanPerSen_DRP%RSVP_1_30_10_SE_
Creator:
gnuplot 3.7 patchlevel 0
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

Figure 4.40: Rsum(all nodes) for DRP in SSR mode normalised to RSVP case, shared session

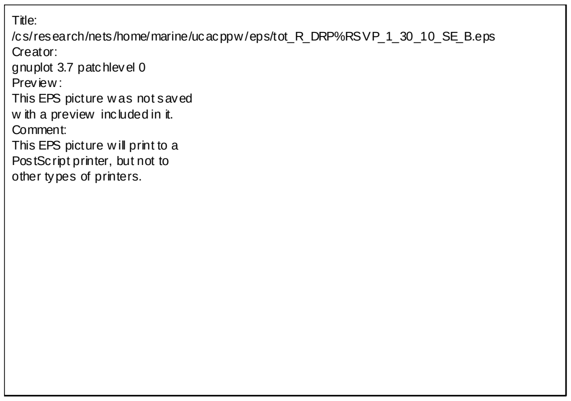

Title:
/cs/research/nets/home/marine/ucacppw/eps/tot_R_DRP%RSVP_1_30_10_SE_B.eps
Creator:
gnuplot 3.7 patchlevel 0
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

Figure 4.41: Rsum(all nodes) for DRP in SSR mode normalised to RSVP case, shared session

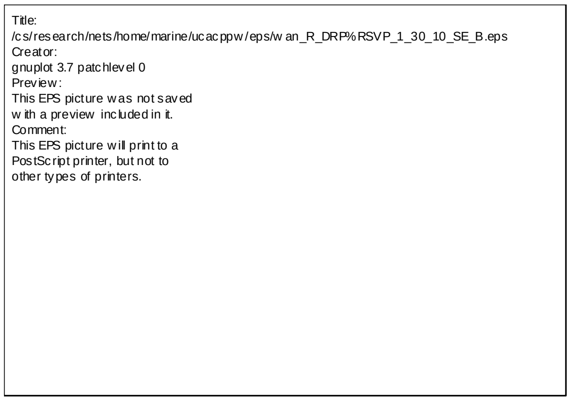

Title:
/cs/research/nets/home/marine/ucacppw/eps/wan_R_DRP%RSVP_1_30_10_SE_B.eps
Creator:
gnuplot 3.7 patchlevel 0
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

Figure 4.42: Rsum(WAN nodes only) for DRP in SSR mode normalised to RSVP case, shared session

For DRP in SS mode, Figure 4.39 shows that for the large network as a whole DRP has a mean value of Rsum between 63 % and 82 % less than that of RSVP across the range of simulations. For DRP in SS mode, Figure 4.40 shows that for the WAN alone DRP has a mean value of Rsum between 57 % and 70 % less than that of RSVP across the range of simulations

For DRP in SSR mode, Figure 4.41 shows that for the large network as a whole DRP has a mean value of Rsum between 17 % and 40 % less than that of RSVP across the range of simulations. For DRP in SSR mode, Figure 4.42 shows that for the WAN-alone DRP has a mean value of Rsum between 12 % and 37 % less than that of RSVP across the range of simulations

4.14 Chapter Summary

In this chapter we presented a new reservation protocol known as DRP which can be used in the context of on-demand multicast services in the network in order to provide a user-perceived Quality of Service commitment. Although motivated by the QoS needs of on-demand multicast services, DRP is sufficiently general to be useful for many other communication scenarios, in particular large-scale-multicast applications where it can expect to achieve a router state saving of several orders of magnitude compared to RSVP.

Due to the worst case merging of DRP RTN messages in multicast trees we showed in 4.3.1 how it was feasible in some scenarios(usually single sender) that DRP might install a higher amount of aggregate reservation bandwidth than RSVP. However in our simulations of single sender video distribution in a multicast tree for typical link layer mixes DRP always achieved a mean aggregate bandwidth saving compared to RSVP although the saving was less than 2.2 % in all cases and less than 1 % typically.

In 4.13 we showed that unlike the single-sender case, substantial aggregate bandwidth savings are possible with DRP compared to RSVP for shared sessions where a substantial amount of heterogeneity exists between sender traffic characteristics. We also expect DRP to achieve a reservation bandwidth saving, albeit to a lesser degree, in shared session scenarios where no heterogeneity between sender traffic characteristics exists, although this was not simulated.

Compared to DRP, the YESSIR approach to facilitating Guaranteed Service in multicast sessions results in installation of a much higher aggregate reservation bandwidth, particularly as the size of the multicast tree grows.

5 A New Method of Reliable Multicast

5.1 Introduction

In section 2.3.2 we discussed PGM[5] as a method of realising a multicast retransmission tree that constrains retransmissions to the minimum subset of links required to deliver the lost packet only to those receivers that are actually missing it. With PGM the default retransmission tree is empty and any receiver requiring reception of a particular lost packet must explicitly join the retransmission tree for that lost packet. Joining the retransmission tree results in the installation of retransmission state indicating the subset of on-tree outgoing interfaces that are to forward the retransmitted packet. We call such approaches 'Reliable Multicast with Retransmission Join'(RMRJ).

We propose an alternative class of approaches which we call 'Reliable Multicast with Retransmission Prune' (RMRP), whereby the default retransmission tree is identical to the multicast tree and any receiver that does not wish to receive the retransmission of an advertised lost packet must explicitly prune itself from the retransmission tree for the lost packet. Pruning of the retransmission tree results in the installation of retransmission state indicating those on-tree interfaces that are not to forward the retransmitted packet.

In the rest of this chapter we discuss the minimum functionality for the fundamental concepts of RMRJ and RMRP to be realised. In the case of RMRJ this minimum functionality is obtained from the PGM specification with appropriate terminology differences for generality. Although PGM contains additional functionality beyond that presented here we ignore it for the purpose of this chapter which is to compare the fundamental characteristics of RMRJ and RMRP.

We present simulation results that compare the efficiency of RMRJ and RMRP in terms of amount of control message overhead and network state for varying numbers of receivers and network sizes. In addition we discuss the environments in which each scheme is most suited and how they might be combined to maintain highly efficient retransmissions over a wider range of network topologies.

5.2 Terminology

In RMRJ and RMRP, retransmisson tree building is on a per (sender, multicast group, lost packet sequence number) basis. Hence, for simplicity of explanation, in the following discussion all packets, retransmission tree state and interface state mentioned refer to the same sender, packet sequence number of lost packet and the same multicast group. Although we consider only the single sender case both RMRJ and RMRP can be used without modification in a multiple sender environment which will result in a mesh of retransmission trees on a per (sender, group, lost packet sequence number) basis. We now present some terminology used in the remainder of this chapter for our analysis of the retransmission tree set up for a specific lost packet of a specific group sender which we refer to as the sender under consideration.

- **Sender** – an end-node that is the sender under consideration.
- **Receiver** – an end-node that has joined the multicast group but is not the sender under consideration.
- **Multicast routing tree** – the multicast routing tree setup by the underlying multicast routing protocol.
- **On-tree interface** – a node interface that is on the multicast routing tree.
- **Parent interface –** a node's on-tree interface on which packets from the sender under consideration arrive.
- **Multicast leaf router** – an on-tree router that has directly-connected receivers.
- **Downstream** – towards the receivers.
- **Upstream** – towards the sender under consideration.
- **RM(Reliable Multicast) interface** – a logical interface stored at a node on a per (on-tree outgoing interface, adjacent on-tree downstream node) basis.

5.3 Sender generated messages

For both RMRJ and RMRP there are 3 kinds of packets generated by the sender, namely Original Data(ODATA), Retransmitted Data(RDATA) and Source Path message(SPM). The main information stored in each of these message headers is shown in Figure 5.1.

ODATA/RDATA	SPM
Sender	Sender
Sequence number	Sequence number
	Previous hop address

Figure 5.1: Main information in ODATA, RDATA and SPM messages.

The IP protocol facilitates a number of options which can be set by the sender on a per-packet basis. One of these is known as the router alert option[3] which can be used to indicate to routers that the packet contains special information that requires higher layer processing.

ODATA packets do not have the router alert option set and the packets travel down the multicast tree in the conventional manner with packet forwarding being determined entirely by the installed underlying multicast forwarding state.
RDATA packets on the other hand do have the router alert option set which results in the packet being passed up to the reliable multicast daemon(RMRJ or RMRP) at each hop. Upon noting that the packet is of type RDATA the reliable multicast daemon will then take account of any installed matching[50] retransmission state as well as multicast forwarding state to determine which interfaces to forward the RDATA packet out of. The daemon will then remove all matching retransmission state.

SPM packets also have the router alert option set so that the reliable multicast daemon at each router can intercept the packet and store the information contained therein as SPM state of the form (sender, group, previous hop address). The router then forwards a copy of the packet out of each of its on-tree outgoing interfaces and in the process updates the Previous Hop Address field in each copy with the IP address of the outgoing interface out of which it is to be forwarded. A router will use the stored Previous Hop Address for a specific sender to correctly reverse-route any receiver generated control messages it may later receive from downstream.

5.4 *RMRJ operation*

Any receiver detecting packet loss for a specific sender must send a Retransmission Join(Rjoin) message to the Previous Hop Address for that sender as obtained from stored SPM state.

Rjoin/RPrune message
Sender IP address
Sequence number

Figure 5.2: main fields in a Rjoin message header.
Reception of a Rjoin message at a router interface causes it to install retransmission Join state at the interface. In addition if the router was not already on the retransmission tree then the RJoin message must be propagated up the tree in which case the router sends it to the Previous Hop Address obtained from its own stored SPM state for the sender. A router is deemed to already be on the retransmission tree if at the moment at which the Rjoin message arrived, matching[50] retransmission join state already existed at any of the node's on-tree outgoing interfaces. Propagation of the RJoin message hop-by-hop towards the sender continues until the RJoin message reaches either the sender itself or a node that is already on the retransmission tree. This procedure will result in the sender receiving a single Rjoin message for each lost packet as illustrated in the example of Figure 5.3.

Upon receiving a Rjoin message the sender must wait for a period of time, dSend, which is no less than the maximum round trip time to any receiver, before sending the retransmission as an RDATA packet. This waiting period is necessary to allow the retransmission state time to settle in the network. Upon reception of an RDATA packet a router will forward the packet out of all outgoing interfaces that contain matching[50] RJoin state. The router will then remove all of the matching Rjoin state.

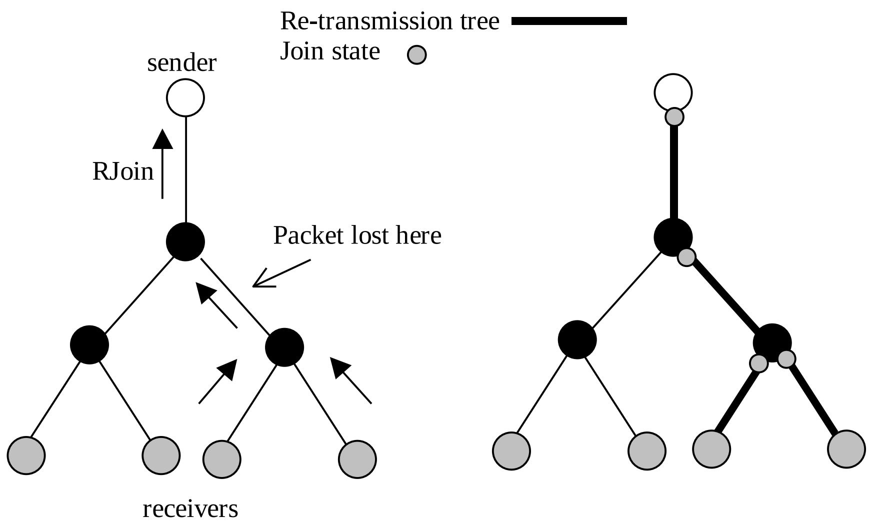


Figure 5.3: Example of RMRJ retransmission tree

5.5 RMRP operation

5.5.1 RM interfaces

Operation of RMRP is more complex than that of RMRJ. In particular at any on-tree outgoing interfaces attached to broadcast-capable links, RMRP requires the inclusion of what we call Reliable Multicast(RM) interfaces which are logical interfaces indicating on-tree connectivity with neighbouring downstream nodes. RM interfaces are on a per (sender, group, outgoing on-tree interface, adjacent downstream on-tree node) basis and consequently a single on-tree interface may encompass multiple RM interfaces for the same sender if that on-tree interface has more than one adjacent downstream node. In the case of an on-tree outgoing interface with directly attached end-

[50] Matching in terms of same sender, group and packet sequence number.

nodes, connectivity to all of the end-nodes is represented using a single RM interface of the form (sender, group, outgoing on-tree interface, *).

5.5.2 Hello Protocol

In order to allow RM interface information to be installed at each outgoing on-tree interface attached to a broadcast-capable link, RMRP makes use of a Hello protocol so that the interface can obtain the list of on-tree downstream adjacent nodes. Each on-tree node with a broadcast-capable parent interface schedules multicast of an Hello message out of its parent on-tree interface every Hello_Interval seconds to the ‘all-RMRP nodes’ multicast address. The Hello message sent out of a given on-tree interface contains a list of the multicast trees[51] for which that on-tree interface is a parent interface. On a broadcast-capable LAN, end-nodes continuously check for Hello messages on this address and only send their own scheduled Hello message provided a duplicate one was not detected in the previous Hello_Interval seconds.

5.5.3 Retransmission tree building with RMRP

In RMRP, any node detecting packet loss schedules the multicast of a group-wide Request message for some point in the future in accordance with a message suppression algorithm such as that used in [4]. With such an algorithm any scheduled Request transmission will be cancelled if a duplicate message is detected before expiration of the scheduler timer.

Request message
Sender IP address
Sequence number

Figure 5.4: main fields in a Request message header.

The group-wide multicast of the Request message which identifies the sender and sequence number of the lost packet, serves two purposes. Firstly, it requests a retransmission from the sender. Secondly, it informs other receivers of the pending retransmission. The action taken by a receiver upon reception of a Request message varies depending on whether or not its parent link is broadcast-capable.

If the parent link is not broadcast-capable then the action taken is very simple in that if the end-node already possesses the packet indicated in the Request message then it must send an Rprune message to the sender’s previous hop address indicated in stored SPM state. Sending an RPrune message

[51] In the case of a source-based tree, the tree is identified by (source, group) whereas in the case of a shared bidirectional tree it is identified by group alone.

effectively prunes the receiver from the retransmission tree for the (sender, sequence number) contained in the RPrune message.

If the parent link is broadcast-capable then upon reception of a Request message the receiver either schedules a local multicast of an Rprune message if it has the lost packet or an Rjoin message if it hasn't. In either case, message suppression is used to prevent duplicate messages being generated on the same link. If transmitted, the Rprune or Rjoin will be multicast with a TTL of one to the 'all-RMRP-nodes' multicast address. Upon reception of a Rprune message a router will only install prune state at that interface provided no matching Rjoin was received in the previous CAL1 seconds or is received in the next CAL2 secs where CAL1 and CAL2 are protocol calibration constants.

Whenever a node receives a prune on an outgoing interface such that installation of associated prune state would result in matching prune state on all of the node's outgoing on-tree interfaces it sends a Rprune message to the Previous Hop Address before deleting all matching prune state. We refer to any prune state deleted in this manner as transient prune state while any prune state that still exists at the instant at which the retransmission tree building is complete is known as steady prune state. Deletion of transient prune state ensures that the amount of prune state is kept to a minimum in the steady state retransmission tree. This process is illustrated in Figure 5.5 where router R2 receives a prune message on each of its two downstream interfaces. The first of these prune messages sets up transient prune state on the interface at which it arrives at R2. When the second prune arrives installation of associated prune state would result in prune state on all of R2's downstream interfaces. Hence R2 deletes the transient prune state and sends the prune up the tree which results in the installation of steady prune state at router R1. The complete retransmission tree building process in this example has therefore consisted of the installation of a single instance of transient prune state and a single instance of steady prune state.

Upon receiving a group-wide Request message, the sender waits a period of time, dSend, which is no less than the maximum RTT for any receiver, before sending the retransmission as a RDATA packet. At each router this RDATA packet will be forwarded out of each on-tree outgoing interface that does not contain matching Rprune state. After forwarding an RDATA packet the router then deletes any Rprune state that matched on the packet.

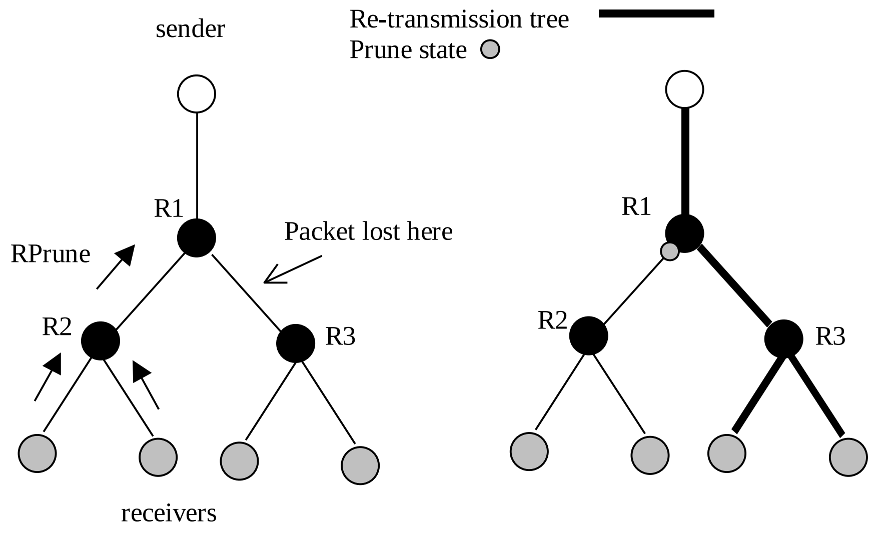


Figure 5.5: Example of RMRP retransmission tree building

5.6 Simulation

We have compared our scheme, RMRP with that of RMRJ using ns[2] with a 3 tier network topology(WAN, MAN, LAN) generated by the tiers topology generator[1] configured for no network redundancy. 2 sets of network simulations were performed for small and large networks with parameters as given in Table 5-1. In each case a single LAN was allocated to each MAN while a single sender was allocated to one of the LANs, the sender LAN, and interested receivers to all remaining LANs, the receiver LANs.

	Small Network	Large Network
Nodes per WAN	8	30
Nodes per MAN	4	10

Table 5-1: Small and Large Network Parameters

Each simulation run comprises the following steps. Firstly all nodes are configured to use RMRJ. Then one of the on-tree links is configured to drop exactly one packet. The sender then sends two packets, the first of which will be dropped at the configured link. The second packet will not be dropped and therefore reaches all receivers. Receivers that missed the first packet detect the packet loss by virtue of a gap in the sequence number space of received packets and inititate recovery of the lost packet according to the reliable multicast protocol. Once all packet loss-recovery is complete the control message traffic and retransmission state measured for this particular packet drop are added to the respective totals. The whole process is then repeated for each of the remaining links in turn to obtain the total amount of per-hop control messages and retransmission state for RMRJ. The reliable multicast protocol is then set to RMRP and the whole sequence of steps repeated for each link in turn to obtain corresponding results for RMRP.

The delays of the MAN and WAN links were allocated randomly using truncated normal distributions[52] with the parameters shown in Table 5-2. The minimum link delay in the LAN case is given by the time required to transmit a maximum length frame assuming the LAN to be 10Mbps ethernet.

	LAN	MAN	WAN
Min link delay	1200us	100us	100us
Mean link delay	1800us	0.5ms	5ms
Max link delay	3600us	5ms	50ms

Table 5-2: Delay parameters for simulations

The results of the simulations are shown in Figure 5.6 to Figure 5.21 where each point was obtained by taking the mean from 10 runs, each of which was given a different seed for the tiers random topology generation. The range between the error bars represents the 90% confidence interval obtained using the t-distribution on the assumption that the mean is that of a gaussian distribution to a first approximation.

[52] A description of truncated normal distribution is given in section 4.8.

5.6.1 Small Network – Simulation Results

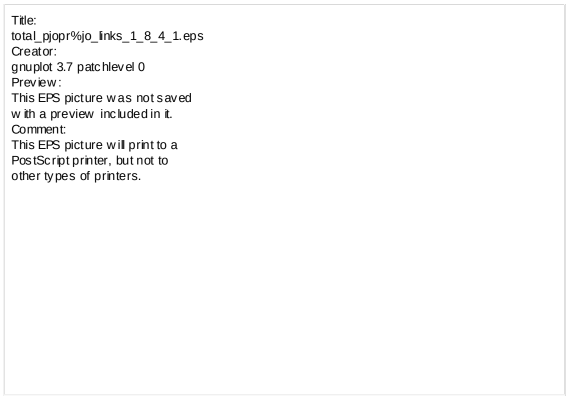

Title:
total_pjopr%jo_links_1_8_4_1.eps
Creator:
gnuplot 3.7 patchlevel 0
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

Figure 5.6: CMR(RMRP/RMRJ): 1*8 node WAN, 4 nodes per MAN, 1 LAN per MAN (all nodes)

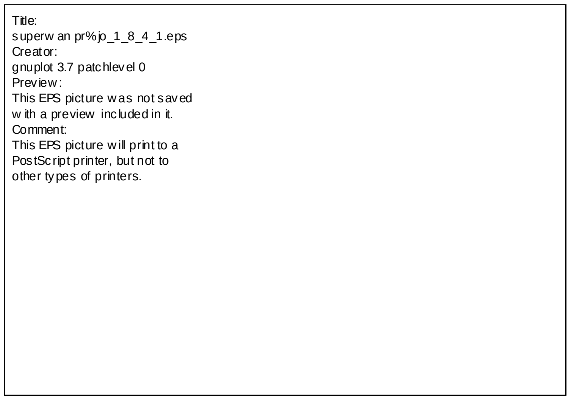

Title:
superwan pr%jo_1_8_4_1.eps
Creator:
gnuplot 3.7 patchlevel 0
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

Figure 5.7: CMR(RMRP/RMRJ): 1*8 node WAN, 4 nodes per MAN, 1 LAN per MAN (WAN nodes only)

Figure 5.6 to Figure 5.7 show the Control Message Ratio, CMR(RMRP/RMRJ), that is the amount of control message overhead for RMRP divided by that for RMRJ. Figure 5.6 is for the network as a whole and Figure 5.7 is just for those network links comprising the WAN. We consider the WAN-only CMR plot to be the more important since control message overhead is likely to be of greatest concern in the WAN where the number of reliable multicast distribution trees is potentially quite large.

Figure 5.7 shows that for the smaller network simulation, when the number of receiver LANs is less than 6 the mean CMR for WAN-only nodes is less than 1 which means that RMRP generates less WAN control traffic than RMRJ. When the number of receiver LANs is equal to 6 or above RMRP generates more WAN-only control message overhead than RMRJ and this difference becomes greater as the number of receiver LANs increase. When the number of receiver LANs is equal to 17, RMRP generates on average over 3.4 times as much WAN-only control message traffic as RMRJ.

Now in Figure 5.6 and Figure 5.7 we assume that control messages for both RMRJ and RMRP require positive acknowledgement with retransmission on a hop-by-hop basis to ensure their reliable delivery. However it is worth noting that control message loss of RMRP is fail-safe in so far as the recovery of lost-packets is concerned whereas with RMRJ it is not. In other words, with RMRP control message loss will not interfere with the distribution of the retransmitted packet to those receivers that are missing it, although it might of course result in distribution of the lost packet to some receivers that already have it. With RMRJ on the other hand loss of a control message will result in non-delivery of the retransmission to one or more receivers that require it. Noting these different failure mechanisms in the retransmission tree building processes of RMRP and RMRJ a case can be made for removing hop-by-hop acknowledgments of control messages in the case of RMRP but not in RMRJ. If this is done then the CMR values in Figure 5.6 to Figure 5.7 are halved which results in RMRP having lower WAN-only control message overhead than RMRJ even when the number of receiver LANs is as high as 11.

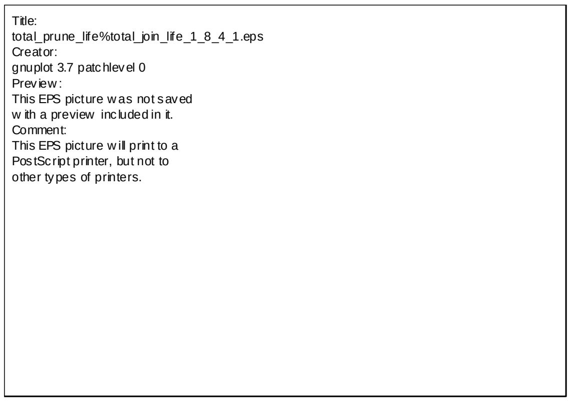

Title:
total_prune_life%total_join_life_1_8_4_1.eps
Creator:
gnuplot 3.7 patchlevel 0
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

Figure 5.8: RSR(RMRP/RMRJ) 1*8 node WAN, 4 nodes per MAN, 1 LAN per MAN, dSendFactor=1 (all nodes)

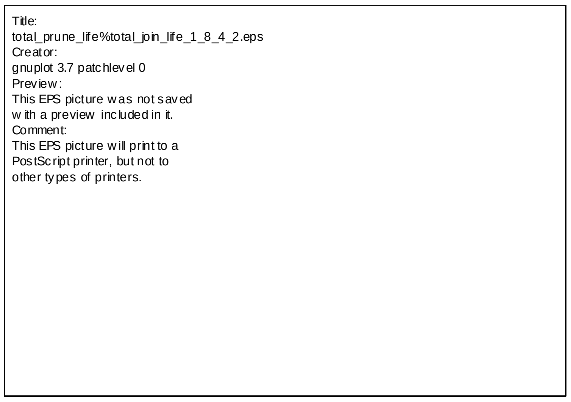

Title:
total_prune_life%total_join_life_1_8_4_2.eps
Creator:
gnuplot 3.7 patchlevel 0
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

Figure 5.9: RSR(RMRP/RMRJ) 1*8 node WAN, 4 nodes per MAN, 1 LAN per MAN, dSendFactor=2 (all nodes)

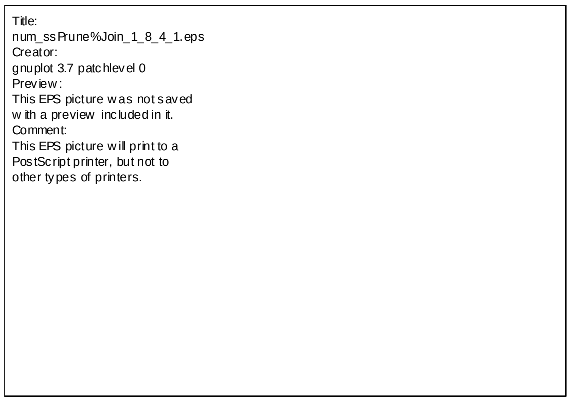
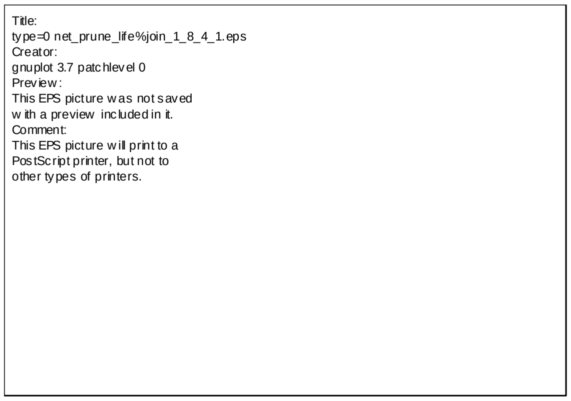
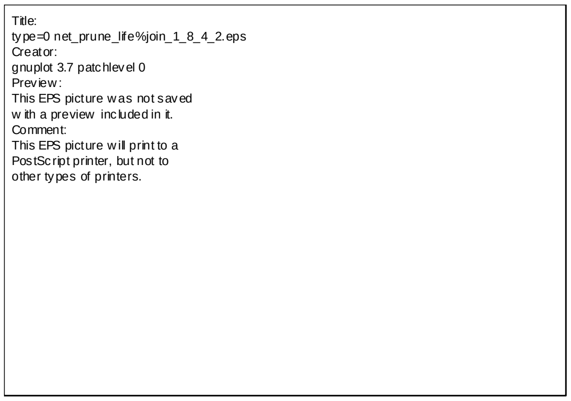

Title:
num_ssPrune%Join_1_8_4_1.eps
Creator:
gnuplot 3.7 patchlevel 0
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

Figure 5.10: RSR(RMRP/RMRJ) 1*8 node WAN, 4 nodes per MAN, 1 LAN per MAN, dSendFactor=∞ (all nodes)



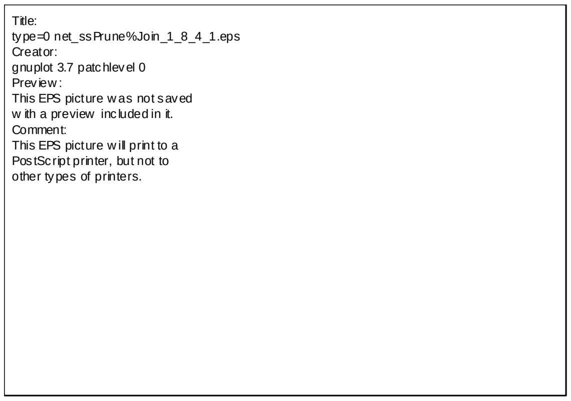
Title:
type=0 net_prune_life%join_1_8_4_1.eps
Creator:
gnuplot 3.7 patchlevel 0
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

Figure 5.11: RSR(RMRP/RMRJ) 1*8 node WAN, 4 nodes per MAN, 1 LAN per MAN, dSendFactor=1 (WAN nodes only)



Title:
type=0 net_prune_life%join_1_8_4_2.eps
Creator:
gnuplot 3.7 patchlevel 0
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

Figure 5.12: RSR(RMRP/RMRJ) 1*8 node WAN, 4 nodes per MAN, 1 LAN per MAN, dSendFactor=2 (WAN nodes only)



Title:
type=0 net_ssPrune%Join_1_8_4_1.eps
Creator:
gnuplot 3.7 patchlevel 0
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

Figure 5.13: RSR(RMRP/RMRJ) 1*8 node WAN, 4 nodes per MAN, 1 LAN per MAN, dSendFactor=∞ (WAN nodes only)

Figure 5.8 to Figure 5.13 show the Retransmission State Ratio (RSR) of the small network for different values of dSendFactor, for both the network as a whole and WAN-only nodes. The RSR is given by dividing the total retransmission state lifetime of RMRP(prune state[53]) by that of RMRJ(join state) while dSendFactor is given by dividing the sender retransmission delay, dSend[54] by RTTmax, the maximum value of round-trip time between the sender and the set of receivers.

[53] And edge join state in the all nodes cases.

[54] dSend is the time between the sender receiving a request for retransmission and actually commencing the retransmission. In the case of RMRJ a retransmission request is conveyed using a Join message while in the case of RMRP it is conveyed using a Request message.

In RMRP the number of instances of steady prune state is usually less, and sometimes much less, than the total number of instances of prune state(steady + transient). The absolute contribution of transient prune state to the total prune state lifetime is fixed and unaffected by dSend. On the other hand the absolute contribution of steady prune state to the total prune state lifetime will increase as dSend increases. Consequently the percentage contribution of transient prune state to the total prune state lifetime will decrease with dSend to a value of 0 as dSend approaches infinity. Hence if we increase dSend we can expect the RSR to decrease. Values of dSend of RTTmax(minimum value of dSend) and infinity will therefore give the maximum and minimum values of RSR respectively. In practice the real value of dSend would be equal to B*RTTmax where B>1 to allow a safety margin[55]. We expect a value of B = 2 to represent a typical configuration.

For the smaller network simulation, Figure 5.11 and Figure 5.12 show the RSR in the WAN-only case for minimum and typical values of dSend respectively. In both cases, the graphs indicate that provided the number of receiver LANs is less than 7, RMRP generates on average less WAN-only reservation state than RMRJ. Figure 5.13 shows the RSR for dSend equal to its maximum value of infinity which is equivalent to the steady state RSR. For this maximum value of dSend, RMRP generates less WAN-only reservation state than RMRJ provided the number of receiver LANs is less than 10.

We can summarise the smaller network simulation results by saying that typically RMRP generates less WAN-only control traffic and WAN-only reservation state than RMRJ provided the number of receiver LANs is less than 6 and 7 respectively. For example when the number of receiver LANs is equal to 4 the control message overhead and reservation state in the WAN for RMRP with a typical value of dSend is about 45 % and 20 % less respectively than what it is for RMRJ.

[55] The senders value of RTTmax will be a continuously updated estimate. A safety margin is therefore required to allow for instantaneous errors in this estimate and so guarantee that dSend is never less than the instantaneous maximum of the RTTs to each receiver.

5.6.2 Large Network – Simulation Results.

Title:
total_pjopr%jo_links_1_30_10_1.eps
Creator:
gnuplot 3.7 patchlevel 0
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

Figure 5.14: CMR(RMRP/RMRJ): 1*30 node WAN, 10 nodes per MAN, 1 LAN per MAN (all nodes)

Title:
superwan pr%jo_1_30_10_1.eps
Creator:
gnuplot 3.7 patchlevel 0
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

Figure 5.15: CMR(RMRP/RMRJ): 1*30 node WAN, 10 nodes per MAN, 1 LAN per MAN (WAN nodes only)

Figure 5.14 and Figure 5.15 show the CMR in the large network for all nodes and WAN-only nodes respectively. Figure 5.15 shows that when the number of receiver LANs is less than 6 RMRP on average generates less WAN control traffic than RMRJ. When the number of receiver LANs is equal to 6 or above RMRP on average generates more WAN-only control message overhead than RMRJ and this difference becomes greater as the number of receiver LANs increase. When the number of receiver LANs is equal to 17, RMRP generates on average 2.5 times as much WAN-only control message traffic as RMRJ.

Title:
total_prune_life%total_join_life_1_30_10_1.eps
Creator:
gnuplot 3.7 patchlevel 0
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

Figure 5.16: RSR(RMRP/RMRJ) 1*30 node WAN, 10 nodes per MAN, 1 LAN per MAN, dSendFactor=1 (all nodes)

Title:
total_prune_life%total_join_life_1_30_10_2.eps
Creator:
gnuplot 3.7 patchlevel 0
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

Figure 5.17: RSR(RMRP/RMRJ) 1*30 node WAN, 10 nodes per MAN, 1 LAN per MAN, dSendFactor=2 (all nodes)

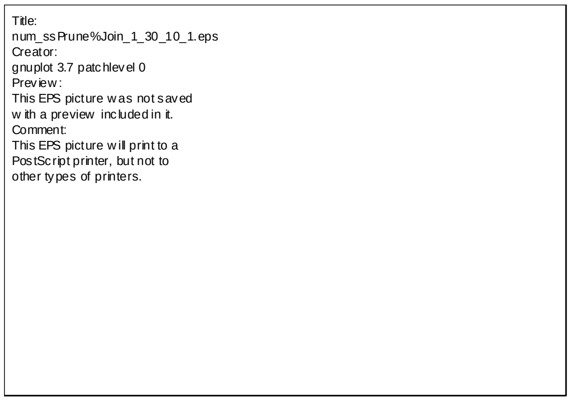

Title:
num_ssPrune%Join_1_30_10_1.eps
Creator:
gnuplot 3.7 patchlevel 0
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

Figure 5.18: RSR(RMRP/RMRJ) 1*30 node WAN, 10 nodes per MAN, 1 LAN per MAN, dSendFactor=∞ (all nodes)

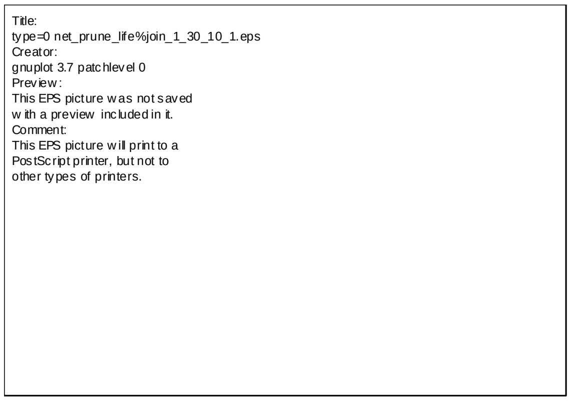

Title:
type=0 net_prune_life%join_1_30_10_1.eps
Creator:
gnuplot 3.7 patchlevel 0
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

Figure 5.19: RSR(RMRP/RMRJ) 1*30 node WAN, 10 nodes per MAN, 1 LAN per MAN, dSendFactor=1 (WAN nodes only)

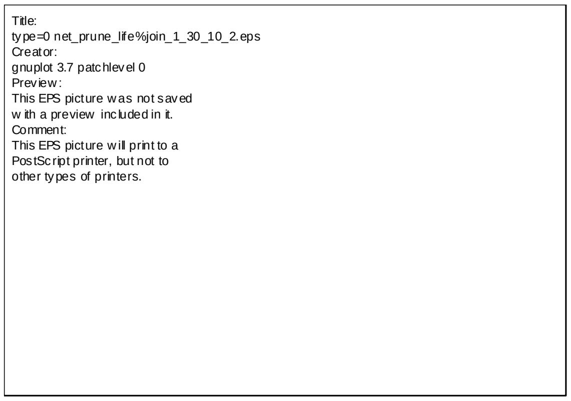

Title:
type=0 net_prune_life%join_1_30_10_2.eps
Creator:
gnuplot 3.7 patchlevel 0
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

Figure 5.20: RSR(RMRP/RMRJ) 1*30 node WAN, 10 nodes per MAN, 1 LAN per MAN, dSendFactor=2 (WAN nodes only)

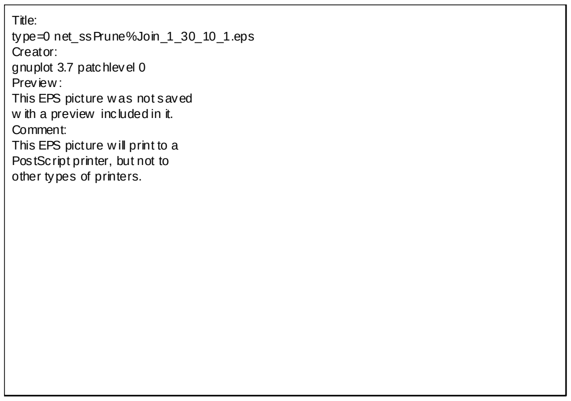

Title:
type=0 net_ssPrune%Join_1_30_10_1.eps
Creator:
gnuplot 3.7 patchlevel 0
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

Figure 5.21: CMR(RMRP/RMRJ) 1*8 node WAN, 4 nodes per MAN, 1 LAN per MAN, dSendFactor=∞ (WAN nodes only)

Figure 5.19 and Figure 5.20 show the RSR in the WAN-only case for minimum and typical values of dSend respectively. Figure 5.19 shows that with a minimum value of dSend RMRP generates less WAN-only reservation state than RMRJ provided the number of receiver LANs is less than 8. Figure 5.20 shows that with a typical value of dSend, RMRP generates less WAN-only reservation state than RMRJ provided the number of receiver LANs is less than 11.

We can summarise the larger network simulation results by saying that typically RMRP generates less WAN-only control traffic and WAN-only reservation state than RMRJ provided the number of receiver LANs is less than 6 and 11 respectively.

5.7 Chapter Summary

We have presented a pruning approach to reliable multicast retransmission tree building as an alternative to the existing join approach used by PGM. Our simulation results indicate that when group membership is low the prune approach is more efficient than the join approach in terms of both control message overhead and control message state.

In the case of on-demand multicast servers requiring explicit requests from clients, the server will have knowledge of the number of clients to be served by any given multicast transmission. In addition by inspecting the IP address of each requesting client the server could also estimate the number of receiver LANs and consequently which method of reliable multicast, RMRJ or RMRP is most suitable. The server could then indicate its chosen method of reliable multicast to each client as part of the redirect message containing the multicast address to be used

6 Summary and Conclusions

6.1 Summary of Thesis

Our thesis has focussed on the provision of a QoS framework to support on-demand multicast services in the Internet. In section 1.1 we discussed the classical IP model and the potential efficiency benefits that can be obtained by using multicast delivery to multiple recipients requiring the same information. In section 1.2 we highlighted how multicast could be used within the context of on-demand services and the three major components required to provide an associated QoS framework, namely on-demand service scheduling, resource reservation/allocation and packet loss recovery. In chapter 2 we presented a taxonomy of the current state of the art with regard to these three components together with some of their limitations. In chapters 3, 4 and 5 we presented our contributions with regard to each of these three components.

6.2 Contributions of thesis

6.2.1 New Methods of Service Scheduling

In chapter 3 we presented two new methods of service scheduling for on-demand multicast services, the first known as AMP with CoS being applicable to resources unlikely to experience transmission overlap and the second, known as OBP with CoS being applicable to resources likely to experience transmission overlap.

Compared to conventional AMP, AMP with CoS yields a superior resource saving in an environment with heterogeneous client latency requirements. A working prototype of AMP with CoS has been built which we evaluated through emulation using a request access pattern obtained from a WWW server.

The scheme of OBP with CoS essentially applies AMP with CoS to the principles of Optimised Patching in order to achieve a very bandwidth efficient near on-demand service whereby the client can set their own latency/cost tradeoff . For OBP (without CoS) we derived equations that allow the optimal maximum patch duration to be calculated while for OBP with CoS we provided guidelines as to how it may be calculated. We believe that OBP with CoS is particularly suited to services like video-on-demand and as such our analysis and simulation results will be particularly useful to service providers intending to implement such a service.

6.2.2 A New Resource Reservation Protocol

In chapter 4 we presented a new resource reservation protocol known as DRP which may be used to accommodate end-applications' QoS needs in the context of on-demand multicast services as well as more general communication scenarios. DRP is a sender-based reservation protocol which may be used to set up the IETF's Integrated Services models 'on-the-fly' in a very bandwidth-efficient manner. We compared the reservation bandwidth efficiency of DRP to that of two other protocols that may also be used to set up IETF integrated services, namely the sender-initiated protocol known as YESSIR and the receiver-initiated one known as RSVP. Simulations found DRP to be comparable to RSVP in the case of sender-specific reservations and more efficient in the case of shared sessions, particularly for large network and group sizes. In our presentation of DRP we also illustrated how it can be used to provide certain features not supported by RSVP and YESSIR such as reservation class heterogeneity among receivers of the same session.

6.2.3 A New Reliable Multicast Protocol

With regard to reliable multicast, in chapter 5 we defined two basic approaches to building an optimal retransmission tree, firstly RMRJ, an example of which is Cisco's PGM and secondly our approach known as RMRP. Unlike RMRJ where the default retransmission tree is empty and any end-host requiring a retransmission must explicitly join the retransmission tree, with RMRP the default retransmission tree is full and any host not wishing to received an advertised retransmission must explicity prune itself from the retransmission tree. Our simulation results show that when the number of receiver LANs is small, RMRP is more efficient than RMRJ in terms of control message overhead and retransmission state in the network. In the case of on-demand multicast services, a server has knowledge of the number of receivers. Moreover by inspection of receiver IP addresses the server can estimate the number of receiver LANs and consequently determine whether RMRP or RMRJ is likely to be more efficient for that particular transmission. The server can then indicate to the set of receivers the chosen method, RMRP or RMRJ for packet loss recovery.

6.3 Network Modifications Required to Support the Framework Presented in this Thesis

As mentioned in the thesis, the work presented is relevant to ISPs and network service providers interested in deploying multicast in the context of on-demand services. Two likely scenarios that we drew attention to are firstly in the delivery of popular web pages and secondly in the delivery of streamed audio and video.

In order to implement our framework the network provider would need to upgrade network components as follows

- The network would need to be made multicast-capable by installing the appropriate multicast protocols including IGMP in the edge devices.
- routers in the network would need to be upgraded to include the integrated RMRJ/RMRP software together with that of DRP.
- those servers acting as sources of on-demand multicast services would need to be upgraded with the appropriate service scheduling mechanisms.
- Web browsers would need upgrading with plug-ins that supported the service and were capable of receiving multicast.

6.4 Limitations and Future Work

Pricing models for implementation of our QoS framework for on-demand multicast services were deemed outside the scope of this thesis and were therefore not addressed in any great detail. However, extensive work on developing workable pricing models would be necessary before such a framework could be made a commercial reality. Other limitations of our work and opportunities for future research are now discussed.

6.4.1 New Methods of Service Scheduling for On-Demand Multicast Services.

In section 3.1.4 we highlighted some implementation issues of AMP with CoS that require further investigation in order to ensure robustness of the technique, namely reliability, multicast join time and the handling of duplicate requests.

With regard to OBP with CoS in 3.2.4 we discussed the concept of a buffer/class cost matrix that may be associated with a request. In a full implementation of our QoS framework however, the cost would be influenced by one more selection parameter in addition to latency class and client buffer capabilities, namely packet level QoS requirements which can be achieved using the other two components of our QoS framework, that is DRP and RMRJ/RMRP. In the case of video-on-demand there are two main points of consideration. Firstly, packets are buffered at the receiver prior to playout which means that any lost packet could potentially be recovered using RMRJ/RMRP before its playout point. Secondly, user-perceived QoS may be acceptable for a non-zero packet loss rate provided the loss rate is small. These factors mean that packet level QoS at the client simply equates to packet loss rate and any target packet loss rate could be achieved using either RMRJ/RMRP or DRP or maybe a combination of both. The pros and cons of these different alternatives for meeting any target packet loss objectives is an area for further study.

6.4.2 A New Resource Reservation Protocol.

Although our work demonstrated many benefits of our protocol, DRP compared to the current state of the art there are several aspects of DRP that require further analysis.

- Modelling the cost tradeoff of in-line resource reservations to alter the reservation over different time granularities, i.e. cost saving of reduced bandwidth consumption versus the cost of router processing of extra reservation messages to achieve this saving.
- The router processing time required to implement a reservation change according to an in-line reservation message.
- The effectiveness of the RTN Guaranteed Service feedback technique discussed in 4.3.2.

6.4.3 A New Method of Reliable Multicast.

Although our work demonstrated the efficiency benefits of RMRP vs RMRJ when the number of receiver LANs is low it did not address the tradeoff of added router complexity(via support of RMRP as well as RMRJ) vs potential resource saving. In addition our simulations only represented a small set of possible network topologies and as such more work would be required in order to obtain the following.

- A robust set of rules describing the parameter space under which RMRP is favourable to RMRJ. These parameters might include the number of hops traversed by each request and the IP addresses of routers traversed. Such information might potentially be obtained at the server by examining TTL count of received requests and also any IP record route option information contained therein. The issues associated with, and reliability of, the use of additional decision metrics beyond simply the number of receiver LANs would require extensive evaluation.
- Procedures indicating how the server can obtain the full set of parameters that define this space.

The PGM implementation of RMRJ includes a mechanism whereby an end-node other than the original sender is able to service retransmission requests. This may happen if that end-node is closer to the point of original packet drop than the sender and as such represents an optimisation that can reduce retransmission bandwidth consumption compared to an approach that only allows retransmissions to be generated by the original sender. The ability of retransmissions to originate from nodes other than the original sender is currently not facilitated by RMRP but can be considered an area for further study.

6.4.4 Billing

It is likely that many on-demand multicast services, particularly video-on-demand(VOD), will wish to bill users. For VOD and other services there may be several parameters that affect the cost to the end-user.

For example in our OBP with CoS scheme presented in 3.2, the cost to the end-user is influenced by the request latency class along with appropriate cost penalties if the user has inadequate buffer space to facilitate the current patch size. A more comprehensive billing scheme would need to consider the cost of usage of DRP and RMRP/RMRJ within the framework. The impact of DRP on billing is particularly interesting since it allows the end-user to downgrade their QoS class below the server-suggested class, CRTs by sending a DRP RTN message. Naturally the user would expect a cost reduction for any such action although the manner in which this could be implemented requires further investigation. A complete billing analysis would also need to consider settlement between intervening networks for the costs of implementing resource reservations.

Another issue that needs to be considered is how to safeguard against end-users obtaining a pay-per-view movie for free. Although comprehensive coverage of such an issue was beyond the scope of this thesis we now raise a basic point with regard to this in that with the traditional IP multicast model the sender is unaware of who has joined the group and as such there exists the possibility of clients joining multicast groups and receiving transmissions anonymously without paying. One basic way to minimise this possibility is for the server to apply a unique encryption to each separate multicast transmission, be it a regular transmission or a patch. At the service point of a batch, over the existing TCP connections to the clients in the batch, the server could transmit the encryption keys in a secure fashion(e.g. using Secure Sockets Layer(SSL)) together with details of the multicast addresses that the clients must join. Such an approach assumes that the only way a client can get hold of the encryption key(s) is over the TCP connection between client and server which cannot be set up without a client identifying itself and therefore be subject to billing. Of course one very simple way to break this assumption is for a non-paying user to obtain the key from a cooperating paying user.

6.5 The Future of QoS Enabled On-Demand Multicast Services

We expect the demand for QoS-enabled on-demand multicast services in the Internet to increase in the future for the following reasons

Collaboration between content and network providers

There is a trend within the communications industry towards increased collaboration between content and network providers as exemplified by the recent Time Warner/AOL merger. For content providers, networks provide a means to increase availability of their content while for network providers, content represents added value to their network. In order to satisfy a high customer demand for content within the constraints of available network bandwidth, on-demand multicast techniques will be needed together with a suitable QoS framework to ensure that end-users QoS expectations are met.

2) Increased bandwidth to the end-user

Many services that are suitable for on-demand multicast services such as video-on-demand require a bandwidth in excess of that currently available to the home user in the UK. However this will almost certainly change over the next decade as adoption of new technologies such as ADSL will make high bandwidth domestic access to IP networks commonplace.

3) Popularisation of the Set-top box

Digital TV transmission and acceptance of the set-top box in homes will remove the distinction between TV and data distribution and create a domestic environment that caters for the interactive needs of end-users.

7 References

[1] C.C. Aggarwal, J.L. Wolf and P.S. Yu. A Permutation-Based Pyramid Broadcasting Scheme for Metropolitan Video-On-Demand Systems. In Proc. of the IEEE International Conference on Multimedia Systems '96, Hiroshima, Japan, June 1996.

[2] K. Almeroth, M. Ammar, Z. Fei. Scalable Delivery of Web Pages Using Cyclic Best-Effort (UDP) Multicast, Networking and Telecommunications Group Georgia Institute of Technology Atlanta, Georgia. INFOCOM '98.

[3] Kevin Almeroth, email regarding Mbone size estimate, http://www.ipmulticast.com/hypermail/ipmulticast/1997/0328.html

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

[5] M. Arlitt and C. Williamson. Web Server Workload Characterization: The Search for Invariants, proceedings of the 1996 ACM SIGMETRICS Conference on the Measurement and Modeling of Computer Systems, Philadelphia, PA, May 23-26, 1996.

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

[7] A. Ballardie, Core Based Trees (CBT version 2) Multicast Routing, RFC2189, September 1997

[8] A. Ballardie. A New Approach to Multicast Communication in a Datagram Internetwork, PhD thesis, University College London, May 1995.

[9] 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.

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

[11] Y. Cai, K. Hua and K. Vu. Optimizing Patching Performance

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

[13] A. Chankhunthod et al. A Hierarchical Internet Object Cache. Proc. 1996 USENIX Technical Conference, San Diego, CA, Jan 1996.

[14] R. Clark, M. Ammar. Providing Scalable Web Service Using Multicast Delivery, Proceedings of 2nd International Workshop on Services in Distributed and Networked Environments (SDNE 95), June, 1995.

[15] A. Dan, D. Sitaram, and P. Shahabuddin. Scheduling Policies for an On-Demand Video Server with Batching. *Proc. of ACM Multimedia*, pages 15-23, San Francsisco, California, October 1994

[16] A. Dan, D. Sitaram, and P. Shahabuddin. Scheduling Policies for an On-Demand Video Server with Batching. In Proc. of ACM Multimedia, pages 15-23, San Francsisco, California, October 1994.

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

[18] Estrin et al. Protocol Independent Multicast(PIM-SM), RFC2362, June 1998

[19] R. Fielding et al. Hypertext Transfer Protocol – HTTP 1.1, Request for Comments, RFC2068, January 1997.

[20] S. Floyd, V. Jacobson and Steve McCanne. A Reliable Multicast Framework for Light-Weight Sessions and Application Level Framing, Corrected version of SIGCOMM '95 paper, ftp://ftp.ee.lbl.gov/papers/srm1.ps.Z

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

[22] M. Garrett and M. Borden. Interoperation of Controlled-Load Service and Guaranteed Service with ATM, IETF Request for Comments, RFC2381, August 1998.

[23] K.A. Hua and S.Sheu. Skyscraper Broadcasting: A new Broadcasting Scheme for Metropolitan Video-On-Demand Systems. In Proc. of the ACM SIGCOMM'97, Cannes, France, September 1997.

[24] K. Hua and Y. Cai. Patching: A Multicast Technique for True On-Demand Services, School of Computer Science, University of Central Florida, published in ?????

[25] K.A. Hua and S.Sheu. Skyscraper Broadcasting: A new Broadcasting Scheme for Metropolitan VOD Systems. *Proc. of the ACM SIGCOMM'97*, Cannes, France, September 1997.

[26] K. Hua and Y. Cai. Patching: A Multicast Technique for True On-Demand Services, School of Computer Science, University of Central Florida,

[27] S. Herzog, S. Schenker and D. Estrin. Sharing the "Cost" of Multicast Trees: An Axiomatic Analysis. ACM SIGCOMM '95, August 1995, Cambridge, Massachusetts, USA.

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

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

[30] IETF, Source-Specific Multicast(ssm) Charter, http://www.ietf.org/html.charters/ssm-charter.html

[31] 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.

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

[33] D. Katz. IP Router Alert Option, IETF Request for Comments, RFC2113.

[34] Kolikalapudi Sriram, R. McKinney and Moslafa Hashen Sherif. Voice Packetisation and Compression in Broadband ATM Networks. IEEE Journal on Selected Areas in Communications, vol 9, no 3, April 1991.

[35] T. Lee et al. Hypertext Transfer Protocol – HTTP/1.0. RFC1945, May 1996.

[36] Levinson, E., "The MIME Multipart/Related Content- Type", RFC 2389, August 1998.

[37] G. Q. McGuire Jnr, Mbone growth, http://www.it.kth.se/edu/gru/Fingerinfo/telesys.finger/INW.VT96/Lectures/Module5/Multicasting-23.html

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

[39] J. Nonnenmacher and E. W. Biersack. Asynchronous multicast push: Amp. In Proc. of ICCC'97 International Conference on Computer Communications, Cannes, France, November 1997. http://www.eurecom.fr/~nonnen/

[40] P.Pan and H.Schulzrinne. YESSIR: A Simple Reservation Mechanism for the Internet. IBM Research Report.http://www.watson.ibm.com:8080/PS/8717.ps.gz

[41] J. Palme et al. MIME Encapsulation of Aggregate Documents, such as HTML (MHTML), RFC 2557, March 1999

[42] V. Paxson and S. Floyd. Wide-Area Traffic: The Failure of Poisson Modeling, Proceedings of ACM SIGCOMM '94, London, England, pp 257-268, August 1994.

[43] P. Rodriguez, E. Biersack. Continuous Multicast distribution of web documents over the Internet, Institut Eurocom, France, Nov 1997.

[44] 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.

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

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

[47] Tony Speakman et al. PGM Reliable Transport Protocol Specification, IETF Internet Draft, work in progress, draft-speakman-pgm-spec-*.txt

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

[49] S. Viswanathan and T. Imielinski. Metropolitan Area Video-On-Demand Service using Pyramid Broadcasting. *Multimedia Systems*, 4(4):179-208, August 1996.

[50] D. Waitzman, C. Partridge, S. Deering. Distance Vector Multicast Routing Protocol, RFC1075, November 1988

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

[52] P. White and J. Crowcroft. Integrated Services in the Internet: State of the Art(extended version of publication #4), Proceedings of IEEE, December 1997, Volume 85, No. 12, pp 1934-1946.

[53] P. White. ATM Switching and IP Routing Integration: The next Stage in Internet Evolution? 2nd IEEE International Workshop on Broadband Switching Systems in Taipei, Taiwan, December 3 - 4, 1997. Also appears in April 1998 issue of the IEEE Communications magazine.

[54] P. White and J. Crowcroft. A dynamic sender-initiated reservation protocol for the Internet. 8th IFIP Conference on High Performance Networking (HPN'98) The Millennium Push of Internet, Vienna University of Technology, Vienna, Austria, September 21-25, 1998

[55] P.White and J.Crowcroft. WWW Multicast Delivery with Classes of Service. HIPPARCH '98, University College London, June 15-16 '98, ftp://cs.ucl.ac.uk/darpa/WWWcos.ps.gz

[56] P. White and J. Crowcroft. A Dynamic Sender-Initiated Reservation Protocol for the Internet. Paul White and Jon Crowcroft. A Case for Dynamic Sender-Initiated Reservations in the Internet. Journal of High Speed Networks. Special issue on QoS Routing and Signaling. Vol 7 No 2, 1998.

[57] P. White and J. Crowcroft. A New Technique to Improve the Retransmission Efficiency of Reliable IP Multicast. First International Workshop on Networked Group Communication, Pisa 17-20 November 1999 (poster paper)

[58] P. White and J. Crowcroft. Optimised Batch Patching with Classes of Service. submitted to ACM/CCR, July 2000.

[59] J. Wroclawski. Specification of the Controlled-Load Network Element Service, Request for Comments, September 1997, RFC2211.

[60] Tiers v1.1 random topology generator, http://www-mash.cs.berkeley.edu/ns/ns-topogen.html

[61] UCL/LBNL/VINT Network Simulator – ns (version 2), http://www-mash.cs.berkeley.edu/ns/

[62] S. Viswanathan and T. Imielinski. Metropolitan Area Video-On-Demand Service using Pyramid Broadcasting. Multimedia Systems, 4(4):179-208, August 1996.

[63] Comparison of reliable multicast protocols web page, http://www.tascnets.com/mist/doc/mcpCompare.html

[64] Internet Domain Survey Host Count. Internet Software Consortium, http://www.isc.org/ds/hosts.html

[65] MPEG home page, http://www.cselt.it/mpeg