UNIVERSITY OF CALIFORNIA,
IRVINE

SM: Real-Time Multicast Protocols to Meet the Requirement of Simultaneous Message Delivery

THESIS

submitted in partial satisfaction of the requirements for the degree of

MASTER OF SCIENCE

in Electrical and Computer Engineering

by

Jose - Miguel Pulido

Thesis committee:
Professor Kwei - Jay Lin, Chair
Professor Douglas M. Blough
Professor Wei Kang (Kevin) Tsai

1998

The thesis of Jose - Miguel Pulido is approved:

_____

_____

_____

Committee Chair

University of California, Irvine
1998

DEDICATIONS

To Agatha, for her support and patience

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENT

The author gratefully acknowledges the following individuals and institutions:

- Professor Kwei - Jay Lin, for his continuous guidance and support, and for the freedom provided during the research process.

- Professors Douglas Blough and Wei Kang (Kevin) Tsai, for their insightful comments.

- Members of the Direct research group for their valuable comments.

- Peter Balsells and Generalitat of Catalunya for the economic support provided through the Balsells/Generalitat fellowship.

- Professor Roger H. Rangel for his support and friendship.

Abstract

SM: Real-Time Multicast Protocols to Meet the Requirement of Simultaneous Message Delivery

by

Jose - Miguel Pulido

Master of Science in Electrical and Computer Engineering

University of California, Irvine, 1998

Professor Kwei - Jay Lin, Chair

Simultaneous Multicast (SM) applications require that all receivers receive the same message from the sender simultaneously. SM applications need to meet a strict deadline, that is, to ensure that each receiver will receive a message at the same time, regardless of the network conditions, heterogeneity of the environment and differences between the local clocks in the receivers. In this thesis, a new real-time multicast protocol is proposed so that the coordinated deadline can be met by controlling the reserved bandwidth and without the overhead of clock synchronization. A sensing protocol is also proposed to continuously search for better levels of quality and avoiding the overhead of

an external feedback protocol. The protocols are bandwidth-conscious, enforce fairness and provide scalability.

Chapter 1

Introduction

1.1 Overview

The future of the Internet will rely very much in how successful it will be to become an integrated services network. Integrated services networks transmit different kinds of information through the same infrastructure. This is possible thanks to the digital technology, where the network only sees a sequence of 1's and 0's, and it is the responsibility of the ends of the communication to reconvert the signal to the proper format. For example, the same network can transmit both a voice conversation or a text

file. Each format may demand different types of services. It is essential for a network technology to be able to discriminate among different types of traffic and ensure that less important traffic cannot dramatically interfere with the traffic that has a higher priority. The file transfer application may demand that a file must be transmitted as soon as possible. The audio application may demand a constant rate between samples to maintain a proper conversation. Applications that need to meet a time deadline, either between samples or for the complete transmission, are called real-time applications. In general, real time applications have a higher priority than file transfer applications, even though it does not mean that file transfer applications should starve while the network tries to meet the real time requirements of real time applications. The information is immediately useful at the time when it arrives, such as each word in a conversation. In file transfer applications the information is not really useful until it has been fully transmitted, or at least a basic part, such as the text of a web page. The current Internet provides the support for file transfer applications, through a best effort service. The network allocates bandwidth among all simultaneous users as best as it can, without any explicit commitment as to rate, delay or any other service.

The next goal is to incorporate services to provide the support for real-time applications and become a truly integrated services network. Much work has been done to create network services to support real-time applications. This thesis explores if that support is enough for a specific type of real-time applications called the simultaneous multicast applications, where a coordinated delivery between different receivers must be achieved in a timely basis.

1.2 Real-Time Systems

Real-time applications are required to complete their work and deliver their services in a timely basis. In other words, timing constraints are associated with the applications, and they only work well if the timing constraints are met. Timing constraints are divided into two types: hard and soft. According to a common definition, a timing constraint or deadline is *hard* if the failure to meet it is considered to be catastrophic. In contrast, a late completion of a *soft* deadline is undesirable, but can be tolerated under some circumstances. This definition is somewhat imprecise because it can lead to a question of how catastrophic is catastrophic. Therefore, a more operational criteria can be adopted: an application is considered a hard real-time application if a validation that the application has met the timing constraint is always required. And soft real-time applications are those where the validation is not required, only a demonstration that the application meets some statistical constraint, i.e., less than three missed deadlines every hour.

Most of the real-time applications in the Internet are considered soft real-time applications. The main examples are the multimedia applications, where a constant rate between samples is required, but where the loss of some samples or a little jitter between them does not alter appreciably the communication.

1.3 Adaptive Applications

Two parallel initiatives have been followed to provide the support for real-time applications in the Internet. One of them is adopted by the adaptive applications.

Adaptive applications are soft real time applications that try to guarantee a bounded delay by adapting themselves to the dynamic network conditions. Among them, multimedia applications are the ones that have received the most attention. Multimedia applications generate isochronous traffic[8] and place a requirement of bounded delay to the underlying network. Isochronous traffic demands a constant rate between samples. Some examples are audio and video. In audio, any discontinuity between samples can lead either to a long silence, or to two contiguous phonemes superposed. It was a general belief that the best-effort service provided by IP was inappropriate to carry isochronous traffic, because the transmission delays for the packets on the Internet depend very much on the number of simultaneous users and the current loads of the routers between the source and the destination. In this service, the network allocates bandwidth among all the instantaneous users as best as it can, and attempts to serve all of them without making any explicit commitment as to rate or any other service quality. But work done in adaptive applications has shown that the application can adapt itself to the network circumstances without losing quality[3]. The application "senses" the network to characterize it and determine the available bandwidth. This could be done by some network signaling procedure, such as an ICMP message sent by routers, or by an end-to-end procedure such as receiving feedback messages from receivers. The feedback information allows the source to adjust the transmission rate to the current network conditions.

1.4 Resource Reservation

Resource reservation is the other initiative to provide support for real-time applications in the Internet. Resource reservation protocols allow bandwidth reservations on the Internet to guarantee the requirements of real time applications. One of the most well-known protocols is the Resource ReSerVation Protocol (RSVP). RSVP[18] allows the receivers to reserve some specific bandwidth on each link leading to the source. By combining several requests on the same link, routers may allocate only a small amount of resources to meet the needs of many receivers. In this way, the packet transmission delay can be much better managed to meet the QoS requirements of real time applications.

The Resource ReSerVation Protocol (RSVP) is currently being accepted by the Internet Society to become the standard mechanism for reservations across the Internet. RSVP is a receiver-oriented protocol due to the expected heterogeneity of different receivers. Moreover, if the receiver is the one who is going to pay for the quality of service, the receiver should be able to decide. RSVP uses two types of messages. Path messages are sent by the source to describe the flow and record the path to the sources in the routers. Reservation messages are sent by the receivers to request the desired reservation according to the flow and its own resources. The routers determine if they are able to guarantee the reservation (through some kind of admission control mechanism) and forward back the reservation to the next router in the reverse path that has recorded after the path message has been sent.

Therefore, the router maintains the information about each flow. In other words, reservations install states in the routers. This is a new feature somewhat contradictory to the end to end principle that states that the network should be simple and fast, by

keeping it stateless. RSVP tries to retain part of this principle by defining these states as soft states. That is, if the reservations are not refreshed periodically, the routers eliminate them. If the route changes, the soft state will be deleted in the old routers and the periodic refreshing will install the state in the new ones.

Another RSVP feature that must be taken into account is the concept of filter. Filters indicate who can use the reservation, and the reservation simply indicates how many resources must be reserved. The receiver specifies that only packets from a determined source can use the reservations, and only packets coming from that determined source will be able to use the resources to reach the receiver that has done the reservation.

1.5 Multicast

Although RSVP has been designed to support both unicast and multicast connections, it provides more advantages in the latter case. Multicasting makes it possible for a single sender to deliver the same data to multiple receivers. Instead of having a different channel per receiver, a unique channel is created and the receivers join this channel. Multicast can be used at different levels of the communication. In the physical layer, multicast can be used to reach a selected group of network interfaces in the same local area network. Ethernet interfaces, for example, have a specific range of addresses reserved for multicast. At the network level, multicast is used to reach different hosts scattered around a wide or local area network, being bandwidth efficient by minimizing the number of intermediate links. At the user level, specific middleware such as Horus[16] has been designed to provide different processes in a distributed system with

services such as reliable multicast, message ordering and flow control. We are interested in multicast at the network level, specifically the multicast service provided by the Internet, called IP multicast.

In IP multicast, the multicast group is implemented by the routers using a spanning tree, which is an undirected graph formed by a connected subset of the routers in the subnet that contain no loops. Each router implements a routing algorithm that computes a spanning tree covering all the routers in the subnet. When a source sends a message to the multicast group, the first router examines the tree and prunes it, eliminating all the links that do not lead to members of the group, and finally sends the incoming message through the remaining outgoing links. Various ways of pruning are possible, and they depend on the routing algorithms used by the routers, such as MOSPF or CBT[15]. Multicast Open Short Path First (MOSPF) adds multicast extensions to OSPF, an adaptive routing algorithm. This kind of routing algorithm changes its routing decisions to reflect changes in the topology and in the traffic. OSPF uses link state routing to compute the route tables. The link state algorithm works as follows: Each router discovers its neighbors, learns their network addresses and measures the delay or cost to each of them. Then it constructs a packet telling all it has just learned, and sends the packet to all other routers in the subnet. These packets are flooded by each router in the subnet, that is, they are sent to each outgoing link except the one where the packet has come. At the end, each router has information about all the routers in the subnet, and can compute locally the shortest path to every other router in the subnet. In MOSPF, as link state is used, each router has a complete knowledge of all the routers in the subnet, and therefore a complete local spanning tree. It can be pruned by starting at the end of

each path and working toward the root, removing all routers that do not belong to the group in question.

It is the responsibility of the receiver to subscribe to that group to be able to receive the message. The subscription is done using a very simple protocol called Internet Group Management Protocol (IGMP). Hosts in a local network use the protocol to inform their router if they want to join/leave a multicast group. Later, the router will communicate the enrollment to the rest of the routers in the subnet.

1.6 Clock synchronization

Some multicast groups may require some kind of clock synchronization among the receivers, or between the source and the receivers. For example, all receivers of a multicast group should receive a stock quote update at the same real time, no matter how far they are from the source. Computer clocks choose the Universal Coordinated Time (UTC) as a baseline. That means that (in theory) all systems should have exactly the same time counter at any point in time. But computer clocks depend on the mechanical properties of the quartz crystal that forms the hardware clock, and therefore exhibit some natural drift. As no quartz crystals oscillate at precisely the same rate, each machine's clock drifts differently. Clock synchronization can be achieved in the Internet through the Network Time Protocol (NTP) [5], a network time protocol used to provide time distribution and synchronization services across the Internet.

A computer willing to synchronize its clock, usually connects to another computer with access to an accurate time source, requesting its current time value. This value is then sent back to the requester. The transmission delay must be added to the

time value received from the accurate sender in order to set the real current time. If the delay is unpredictable, the accuracy gained connecting to an accurate source is lost. In addition, network latency and scheduling latency are also contributing factors of unpredictability.

NTP connects to different sources and chooses the one with the least variable communication delay. There are several steps in the process: First, NTP computes the round trip delay and the offset with every source. It does so by collecting timestamps in both sides: when the request is sent (T1), when the request is received (T2), when the answer is sent back (T3) and when the answer is received (T4). The Round Trip Time (RTT) can be computed:

$$RTT = T4 - T1 - (T3 - T2)$$

as the pairs T4-T1, and T3-T2 are measured in the same clock, the difference computed is independent of the clock offset between the computers, and therefore very accurate. The offset is then computed as follows:

$$T2 = T1 + RTT/2 + Offset$$

$$T4 = T3 + RTT/2 - Offset$$

Two assumptions are made: the RTT is symmetric, and the offset does not vary greatly in a short interval of time. Therefore, the offset can be computed as:

$$Offset = 1/2 \; (\; (T2 - T1) - (T4 - T3) \;)$$

In NTP, the offset between the clocks is the value added to the receiver's clock in order to synchronize. The real offset must take into account the transmission delay, which is half of the RTT, plus scheduling and network latencies:

$$d = RTT/2 + \mu$$

where μ contains statistical latencies. And therefore the real offset th satisfies:

$$Offset - d < th < Offset + d$$

After computing the offset for different sources, an intersection algorithm is applied to discard the "false tickers". There are usually offset variations among the peers who survive the intersection algorithm, so a clustering algorithm is designed to select the best subset of the population, ranking the peers first by stratum, then by *d* in ascendant order. Finally, the offset is computed with all the survivals using a combining algorithm that assigns a weight to each survival and computes an average.

NTP obtains a high accuracy by using expensive algorithms that may add significant overheads. Therefore, it should be used only when it is really necessary.

1.7 Simultaneous Multicast Applications

This thesis presents yet another type of soft real time application: simultaneous multicast (SM) applications. Different from traditional multicasts, SM requires that all receivers receive the same message from the sender simultaneously regardless of the network conditions, heterogeneity of the environment and differences between the local clocks in the receivers.

There are many potential SM applications. For example, a news conference on the Internet should ensure all reporters receive the information at the same time. The stock market should provide all potential investors the same information at the same time to prevent unfair trading. Bidders in a real time auction [9] should be provided simultaneously with updated bids. Questions in an online contest should arrive to all the participants at the same time to avoid giving any response advantage. And players in online gaming should receive simultaneously coordinated updates to avoid asynchronous actions.

SM is a truly real-time requirement, in the sense that SM does not need to be fast but must be precisely in real time. SM applications share the same bounded delay requirement with multimedia applications, but they do not need to maintain a constant rate between samples. This is because SM applications may not be sending continuous media, but a single message at a time, probably of predefined size. They also differ from multimedia applications in that they need some kind of clock synchronization with the source to enforce simultaneity. In other words, the same message should be received at the same time no matter what differences there are between local clocks in different receivers. And yet another difference is that receivers can not request different levels of

QoS, because a strict time deadline must be enforced. Receivers who fail to meet the deadline can not request to meet a “lower quality” deadline. However, different levels of global QoS can be achieved, i. e., SM applications can detect opportunities to move up the deadline and allow some receivers to reduce their reservations, or to increase the amount of information in the same deadline.

In addition, SM receivers are supposed to react to the message sent and possibly to reply only to the source through a point to point communication outside the multicast group. For example, several participants in an online auction may decide to provide a higher bid after receiving the last update. In the worst case, the source will have to support several simultaneous point to point connections which requirements can exceed the source’s available resources. The source will not be able to handle all the simultaneous one-to-one connections, and some may be dropped. This phenomenon is known as a source implosion situation, which can happen if the multicast group is large and if there is no coordination between receivers’ feedback. One solution to that problem would be the use of the Real Time Control Protocol (RTCP) [12]. It has been deployed to allow a coordinated feedback between receivers. The protocol allows the receivers to learn how many members are in the group. Each receiver computes locally an estimation of the size of the group. The interval between packet transmission is then set to scale linearly with the estimated number of users. This has the effect of giving each member a fair share of the RTCP packet rate to the multicast group. But the RTCP protocol is an external protocol to the application and therefore adds extra overhead, that is always undesirable.

The thesis presents two new protocols that use the multicast and resource reservation services of the Internet to support SM applications. The first protocol is based on RSVP and provides a method to ensure a coordinated delivery by controlling the reserved bandwidth. It does so by avoiding the overhead of clock synchronization. The second protocol provides a constant search for a better global quality by sensing the network continuously and leading to stabilization at better levels of quality. In other words, the protocol allows the SM applications to continue meeting the coordinated delivery no matter the variations in dynamic conditions, and it achieves that without the overhead of an external feedback protocol such as RTCP.

The thesis is organized as follows. Chapter 2 presents the system requirements and the design goals along with the related work. The design of simultaneous multicast protocols is discussed in Chapter 3. Chapter 4 contains the formal specification of the protocols, which are demonstrated through an example in Chapter 5. Chapter 6 studies some improvements in the design and demonstrates the advantages of the protocols. The thesis concludes in Chapter 7.

Chapter 2

Requirements and Design Goals

2.1 Requirements

The main requirement of an SM application is to meet a strict time deadline, that is, to ensure that a message will be received by each receiver at the same real time, regardless of the network conditions, heterogeneity of the environment and differences between the local clocks in the receivers. To achieve it, a model with two main components is considered:

- Multicast and resource reservation services provided by the underlying network.

- Protocols developed to control the delivery time with the bandwidth and to search for a better quality.

SM applications demand multicast and resource reservation services to the underlying packet network. In the current Internet architecture, the network can create unpredictable delay, or delay jitter, because the congestion in routers may lead to delays or packets being dropped. Although a congestion control and avoidance algorithm is implemented in TCP[4], it depends on the responsible use of each host. The network itself does not have any mechanism to implement fairness because routers only keep information about routes and they have no knowledge about active sessions. Therefore, unpredictable delay may still occur (However, this stateless nature of the network - routers only keep information about routes but not about sessions- does not also provide support for flow definition and maintenance [7], but it is one of the key features to keep routers simple and fast[11]).

In the past few years, new network architectures and service models have been created to accommodate the multicast and quality of services capabilities demanded by real time applications that will convert the Internet into a truly integrated services packet network. Independent of any design, any architecture can be divided into five components [17]: Flow Specification, Multicast Routing, Reservation, Admission Control and Packet Scheduling. We assume that our underlying architecture supports at least three of these components (and probably all five of them because they are tightly related): routing algorithms to create multicast groups, receiver-oriented resource reservation to accommodate demands from heterogeneous receivers, and a flow specification mechanism to allow the source shape the traffic that it is going to generate.

It is also assumed that reservations are achieved using the RSVP protocol, although other protocols can be used as well.

2.2 Design Goals

The main goal in this work is to provide a coordinated multicast delivery and meet a uniform deadline. In addition, there are several secondary goals in the design. These goals are presented below.

- Meet the deadline. The time to receive a message, after it has been sent, can be divided in two components: the time it takes the first bit to arrive at the receiver, and the time at which the last bit arrives. The first component is the propagation delay and it depends on the physical distance between the source and the receiver, and the network conditions. The second component is the product of the inverse of the throughput with the message size, and it depends on the bandwidth. The delay is fixed and different for each receiver. But the bandwidth can be controlled. Intuitively, farther receivers will have a higher delay, and they will need a higher bandwidth to increase the throughput. Simultaneous delivery time can be achieved controlling the bandwidth. The only requirement is that the required bandwidth for each receiver must be granted by the network through reservations. In addition, dynamic variations of the delay may require dynamic corrections of the reserved bandwidth.
- Provide scalability. Possible source implosion situations must be minimized to allow SM applications scale well with large groups. Source implosion situations can happen if many receivers try to establish individual point to point communications with the source simultaneously, such as when the receivers reply during an active session.

- Enforce fairness. Heterogeneity for SM applications implies that some receivers may be very close to the source and others very far, in terms of delay between the source and the receivers. Fairness means that the farthest receiver should be able to get the message at the same time as closer receivers. Therefore, the farthest receiver will determine the value of the delivery time.
- Enforce quality. There must exist a threshold for the delivery time. Receivers with a delay larger than the threshold should not be allowed to join the SM group to avoid an intolerable degradation of the service for the rest of the participants. Moreover, the search for a better quality must be continuous. Two situations can lead to service improvement: the receiver with the longest delay leaving the session, and lower delays due to changes in network conditions. The protocol that senses the network must be able to detect both situations.
- Be bandwidth-conscious. Receivers should not reserve more bandwidth than that strictly necessary to meet the deadline. In addition, the source will provide the maximum bandwidth value to be reserved by any receiver.
- Simplicity. One of the key successes of the Internet has been the simplicity of its design, where the routers only forward packets and little else. We want to follow this philosophy by avoiding as much as possible the use of external protocols such as NTP for clock synchronization or RTCP for coordination of feedback from the receivers to the source.

2.3 Related Work

Several work has been done in adaptive multimedia applications, such as the INRIA IVS [2] or the LBL vat. Work is currently done to improve multicast protocols, such as the Core-Based Trees and PIM (dense and sparse mode) proposals [15], or the greedy routing heuristic [1]. The main work in resource reservations over the Internet to guarantee QoS has been done in the deployment of RSVP [17]. Several network protocols have been developed to provide time distribution and clock synchronization over the Internet [10], being NTP [6] the most complex and the one who provides a higher accuracy. Little or no attention has been paid to simultaneous multicast application. The closest work is the Real Time Protocol [12], the transport protocol for voice and video over the Internet. Its companion, the Real Time Control Protocol [13], provides a feedback synchronization mechanism that can be useful in the SM applications to avoid source implosion situations. To our knowledge, no prior work has been done to meet time deadlines over the Internet using bandwidth control.

Chapter 3

The Design of Simultaneous Multicast Protocols

This chapter presents the two SM protocols in detail:

- A protocol to meet a uniform deadline using dynamic bandwidth reservation.
- A protocol to continuously search for a better global quality.

The first protocol is used to meet the coordinated deadline without the overhead of an external clock synchronization algorithm. The second protocol is used to continue meeting that deadline no matter the variations in dynamic conditions such as network

load or group membership. It achieves its goal by sensing the network to achieve better global quality, and without the overhead of an external feedback control protocol.

The behavior of SM applications can be divided into three steps: registration process, requesting reservations and searching for better quality. The first protocol is used in the first two steps. During the registration process, the receiver registers to the source and both learn the propagation delay between them. The source uses that information to define the deadline. In the reservation part, the source informs the receivers about the deadline and the receivers request the appropriate reservations. In the third step, the second protocol is used to continue maintaining the coordinated deadline despite of the dynamic variations in network conditions or group membership and obtaining a higher global quality.

3. 1 Protocol SM-R: Reserve bandwidth to meet a uniform deadline

It is assumed that during the registration process, both the source and a receiver compute the RTT and offset using a similar algorithm as used in NTP.

The flow spec is defined in the source using the worst case RTT to be fair with the farthest receiver accepted in the session. The D parameter allows the farthest receiver to receive the bid, because the first part of the delivery time is its delay. The message size M is also specified, because in SM applications the assumption of predefined size is acceptable. The algorithm will be called every time there is an update in any of the parameters.

1. The flow spec is defined in the source with two components:

– Delivery time = (worst case RTT / 2) + D

– Message size = M

2. An RSVP path message is sent containing the flow spec.

3. After receiving the delivery time in the flow spec, the receiver will know how much time left has to receive the M bits of the message, and it will compute the appropriate bandwidth to meet the uniform deadline, and achieve the desired throughput. The receiver computes the bandwidth required with the equation:

$$delivery\ time - delay = M / B$$

Therefore, the required bandwidth will be:

$$B = 2M / (worst\ case\ RTT - current\ RTT\ in\ the\ receiver) + 2D$$

In the worst case, B = M / D.

4. The bandwidth requirement of B will be sent in the next RSVP reservation message.

During the registration process, the source and a receiver interchange timestamped messages to compute the Round Trip Time (RTT) and the offset between both clocks using a similar algorithm to NTP. They do so through a point to point communication outside the multicast session. If link state routing is used in the network, it can be assumed that the communication between the source and the receiver travels the same links either inside or outside the multicast group because each router has the

knowledge of the spanning tree, and the spanning tree should be the same in different routers. This is because the spanning tree has been built from the routing tables, which themselves are built locally using a shortest path algorithm. As the algorithm assumes the graph is undirected, the shortest path will be the same no matter from which router is computed (the only exception is when there are several shortest paths). Therefore, the remaining links in the pruned tree to reach the receiver should match the links used in a point to point communication. It is also possible to assume (as it is done in NTP) that in a reasonable period of time, the RTT is symmetric. Therefore, half of the RTT between the source and the receiver is the current delay of the receiver. The source keeps a variable to store the worst case RTT of any receiver. The value is updated every time a new receiver has a higher RTT than the current worst case. There is always a trade off between fairness and quality, or more precisely, between fairness and real-time services. Fairness is achieved by defining the deadline to include the worst case receiver. But the deadline can not be as high as possible, because an intolerable delay can occur. Therefore, the worst case RTT must be always smaller than a parameterizable threshold to avoid intolerable degradation of the service and achieve quality for the current members of the group. Participants with a RTT higher than the threshold value will be rejected.

Finally, the receiver joins the multicast group and waits for a flow description in order to request the reservation.

The source defines the flow spec by specifying the delivery time and the message size. M indicates the message size, and is defined in bytes. The delivery time is half of the worst case RTT plus an additional time D. The interval of time D allows the farthest

receiver to receive the message, because the delay of this receiver is exactly half of the current worst case RTT. The first bit of the message will arrive at the farthest receiver after this interval of time, and therefore that receiver will have exactly D seconds to receive the message and meet the deadline. Fairness is therefore achieved by defining the deadline to allow the farthest receiver meet the coordinated deadline.

The flow spec is then sent to the multicast group with the next RSVP path message. Upon the reception of the flow spec, each receiver uses an algorithm to compute the necessary amount of bandwidth to be reserved. The difference X between the delivery time and the current delay of the receiver is the time left to receive M bytes. In other words, M bytes in X seconds are the required bandwidth for the receiver. At this point, each receiver has calculated the exact bandwidth needed to meet the deadline. Therefore, being bandwidth efficient is achieved because each receiver requests the minimal reservation to meet the deadline and no more.

The receiver sends an RSVP reservation message to request the resources to guarantee the computed bandwidth. The reservation specifies a fixed packet filter where only packets coming from the source will be allowed to use the reserved resources. If the reservation fails, the deadline can not be guaranteed, and the receiver is then at its own risk, because SM applications can not offer differentiated services due to the strict requirements of time deadlines.

Using a fixed filter means that in a shared link, i. e., a common link in different receivers' path, the requested reservations will be merged, and the reserved bandwidth will be the higher one. As the requested bandwidth reservation by the farthest receiver will be M / D, this bandwidth will be reserved in many links of the multicast tree.

Therefore, the source has a mechanism to enforce bandwidth efficiency because it can control the maximum reserved bandwidth in many links by controlling the values of the message size M and the D parameter.

And yet another goal is achieved: simplicity. Meeting the coordinated delivery time by controlling the reserved bandwidth avoids the use and subsequent overhead of an external clock synchronization algorithm such as NTP because no clock synchronization is needed between the source and the receivers. The receiver does not need to verify that the delivery time is met by subtracting a timestamp included in the message and measured in the source's clock from the receiving time measured in its local clock. It is the network the one is responsible to guarantee a delivery service, and it does so by taking into account the reservations requested by each receiver. Therefore, the receivers know that they will meet the deadline when the reservation is granted by the network, and no additional verification will be needed.

SM applications are Variable Bit Rate (VBR) applications, that is, applications that send data at variable bit rates, but still require bounded delays. The source either will update continuously information to the group if the feedback from receivers is high, or it will sit idle waiting for that feedback. The frequency of the updates is unknown. But every update is required to be received by each receiver simultaneously in a deadline specified by the source. Therefore, it is unknown when the source is going to send information and in which frequency, but it is known when the message must arrive. As the size of the message, M, is also known, the flow spec is very predictable. That predictability allows us to treat SM applications as Constant Bit Rate (CBR) applications, that is, every time a message is sent, M bytes should be received in

RTTworst/2 + D seconds by every member of the group. Therefore, each receiver will reserve an appropriate bandwidth to meet that requirement. The reservation will not be in use permanently, as in traditional CBR application such as multimedia applications. Sometimes the reserved bandwidth will not be used because the source will not have information to send or update, whereas in other cases the source will be constantly sending messages without interruption and using the reserved bandwidth completely. The reserved amount will guarantee that every message will meet the deadline.

It must be said that the protocol can be very wasteful. For example, in an online auction scenario, the receivers will use the reservations to receive the updated bids from the auction server. Some of them may respond back. The source will give receivers a predefined time to respond. This time must include the random decision time as well as the transmission time. During this interval of time the source is sitting idle waiting for bids, and the reservations requested by the clients are wasted. The source will then determine the highest bid from all the received replies, and it will send the new message with the updated bid. As the source waiting time includes the decision time, this waiting time will be probably bigger than the deadline, and therefore the reservation can be wasted more than 50% of the time.

One possible solution is to synchronize the source waiting time with the period of the messages used by RSVP to refresh the soft connections. RSVP wants to enforce the simplicity of the Internet. One of the key factors is the simplicity of the routers, which its main job is to forward packets and they do not have the notion of connection, that is, no states are maintained in them. As a guaranteed service needs some kind of connection definition, the trade off implemented in RSVP is to define a soft connection. Soft states

are maintained in the routers storing information about the reserved resources. But, if the reservations are not refreshed periodically, the soft states are eliminated and the reservations are gone. Receivers can define the refreshing period equal to the deadline. And if the receivers know the duration of the source's waiting time, which is a fair assumption in an online auction because the source can define it proportional to the deadline and inform the users through the flow spec, they can synchronize the beginning of the period with the beginning of the deadline. At the end of the deadline, the receivers will decide not to refresh the reservation that will be deleted by the routers. In the next period, the receiver will request a new reservation. As the waiting time is bigger that the deadline and the refreshing period, the reservation will be granted before the next updated bid sent by the source. With this technique, the percentage of time when the reserved bandwidth is not used is greatly reduced. In any case, this type of solution is application dependent. It may work for the online auction but it may not be needed in the online gaming scenario, where the continuous action nature of the application will keep the reservations fully utilized.

In addition, it may happen that the receiver is willing to pay to keep a permanent reservation, even though sometimes the bandwidth is not used. That can happen for example if a brokerage company decides to keep a permanent bandwidth reservation to be sure that every stock update will be received by the deadline set by the source at the moment of delivery, which is unpredictable. That company is willing to pay for a permanent bandwidth reservation such a regular "permanent" connection in circuit networks, assuming that sometimes will not be used, because the benefits provided by not missing any deadline outplay the costs of maintaining unused bandwidth. And the

design of RSVP as receiver-oriented contributes to give that decision to the user, instead of to the network administration.

The solution proposed above leads us to a related discussion, which is the overhead inherent to RSVP to maintain soft connections. One of the goals of our design is to maintain the simplicity, thus avoiding the need of an external clock synchronization and an external feedback protocol. However, we accept the overhead introduced by RSVP because we would need to use it to obtain a guaranteed service, and we are able to reuse its services to achieve a coordinated guaranteed service without the use of an external clock synch protocol. Moreover, in applications such as the online auction scenario, the overhead can be controlled by the application if the refreshing period is made equal to the deadline. Therefore, another factor in the definition of the deadline will be to control the overhead of RSVP refreshing messages.

Finally, SM applications are well suited either for big chunks of data or small data such a stock value update. In the latter case, the propagation delay is the biggest part of the delivery time. The D parameter will be only a few milliseconds, and the message size M will be a few hundreds of bytes. In the former case, maybe D will be seconds, and M hundreds of thousands of bytes. In both cases, the amount of bandwidth needed to guarantee the deadline is pretty much the same. The only difference is the number of packets needed to deliver the information. Small data may be sent in one packet only.

3.2 Protocol SM-S: Sensing the network for a better quality

The main goal of the second protocol is a continuous search for a better quality by sensing the network, because the network itself will not indicate if more resources are available. The protocol asks the receivers if their situation have become worse. If there is no response, the protocol knows there is an opportunity to increase the QoS, and reduces the worst case RTT by a small increment, therefore reducing the delivery time. This increment is usually set to be relatively small compared to the current value of the worst case RTT.

1. The source sets a timer to the value 2*worst case RTT and sends a request message containing the current worst case RTT.

2. Upon reception, each receiver compares its stored RTT with the received worst case RTT, and replies only if its stored value is higher.

3.1 If a message is received by the source before the timer expires, the worst case RTT is updated.

3.2. If the timer expires at the source without receiving any response, the worst case RTT is decremented by a small value.

4. The protocol SM-R is executed to allow the receivers update their reservations and continue meeting the deadline.

5. SM-S is repeated until 3.1 is executed.

Receivers leaving the session and variable network conditions must be detected to enforce the search for a higher quality, one of the goals. The source periodically send a request message to the group containing the current worst case RTT, and sets a timer to double of the current worst case RTT. Every receiver in the group must respond immediately with a reply, but ONLY if it can be a candidate for the worst case RTT. A receiver can determine if it is a candidate by comparing its current RTT with the worst case RTT received in the message. This solution enforces scalability in a large group by minimizing the number of replies. Only a few members of the group, if any, will be candidates to have a worse RTT and therefore only these ones will reply.

If the source receives a reply before the timer expires, the worst case RTT will be updated and the timer reset. If the timer expires, either the worst case receiver has left the session, or the network conditions have improved leading to smaller delays. In any case, the worst case RTT is decremented in the source in a small quantity. The updated flow spec will be sent in the next path message, and the receivers will react modifying their reservations. The goal is to discover how far can the requirements to the network be pushed, because the network itself says nothing about if more resources are available. Decreasing RTT indiscriminately will probably lead to another receiver becoming the worst case, and the group will “get stable” at a higher level of quality. Note we have improved the global QoS by reducing the RTT and allowing receivers to receive the message earlier. The drawback is that receivers may also need to increase their

reservations to meet the new deadline. A different way to increase the global QoS of all receivers would be keeping the same deadline and increasing the message size M, allowing receivers to get more information with the same reservations. And also another way would be increasing the D to reduce bandwidth reservations across the network. We have chosen the first option in our examples, but other options may be explored.

Therefore, the simplicity goal is achieved again because there is no need of an external feedback protocol such as RTCP to coordinate the feedback from the receivers to the source. The need of coordinated feedback is avoided by reducing the number of possible candidates to reply to a very small number of receivers, or maybe no one at all, because only the candidates for a worst case propagation delay are allowed to respond. And even in large groups, this number will probably remain small, thus avoiding also the need of feedback in them.

Chapter 4

SM Protocols Example and Discussions

An example is presented in this chapter to show how the protocols work. After the example, different issues are described to improve the protocols, such as local estimations for the propagation delay of the receivers, and buffering of early messages. In addition, two scenarios are compared to show the advantages of the protocols. In one of them, the protocols are not used whereas in the other they are. The comparison states the advantages of the protocols because the benefits outplay the little overhead they provide.

4.1 Example

The following assumptions are made:

- The routing protocol has built a multicast tree so that the source can reach all the receivers.
- Receivers R1 and R2 were already registered, and R3 has just finished the registration process.
- The source has sent a RSVP path message containing the flow spec, all the routers have installed the path state, and the message has just arrived at the receivers.

All the reservations will be granted. Figure 1 presents the parameters of the initial situation: all three receivers have computed and stored the RTT and offset with the source. The source has stored the worst case RTT, corresponding to R2.

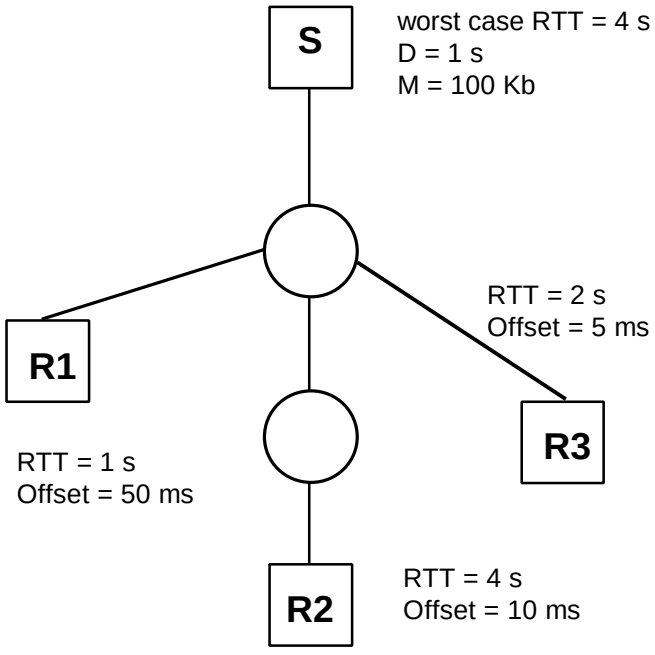

Figure 1. SM application scenario

The message size M is 100kb and the source defines as 1s (D) the lapse of time between the receptions of the first bit of the message and the last bit. Therefore, the maximum allowed reserved bandwidth will be 100kbps (M / D).

At this point, each receiver uses the Protocol SM-R to compute the necessary bandwidth to meet the reservations. The values are:

B1 = 40kbps B2 = 100kbps B3 = 50kbps

Now each receiver sends a RSVP reservation message that travels back to the source using the reversed path information installed in each router. Although it is out of the scope of this thesis to give a detailed description of how RSVP works, a brief description can be given: Each receiver will install a reservation for its requested bandwidth in the link between it and the closest router. But in shared links, that is, common links in the path from the source to the receivers, the higher reservation will be installed. Figure 2 shows the final state of the reservations:

The bandwidth needed for the farthest receiver defines the reserved bandwidth in many of the links. Therefore, an important feature to achieve the goal of being bandwidth-conscious is the control that the source has over the maximum bandwidth used by any receiver. Now consider that the receiver R2 leaves the session. Its absence will be discovered by the protocol SM-S. It will not receive any answer to its request, because all the receivers have a lower RTT than the current worst case. The protocol will react by decreasing the worst case RTT by a small increment. SM-S will be repeated until a new worst case RTT is found. The amount small increment is scaled to the current value of the worst case RTT. If, for example, the increment is 50 ms, the

protocol would need 40 (!) requests to go down from 4s to 2s, the RTT value of R3. After every request, the source will update the flow spec using the protocol SM-R and the receivers will update their reservations. The session will then "get stable" in a higher level of quality as shown in figure 3, with the updated reserved bandwidths being B1' = 66.6 Kbps and B2' = 100 Kbps.

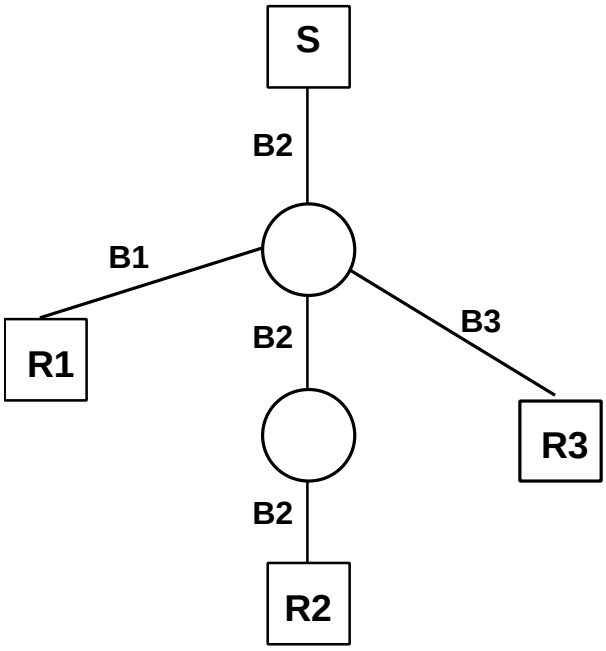

Figure 2. Reserved bandwidth

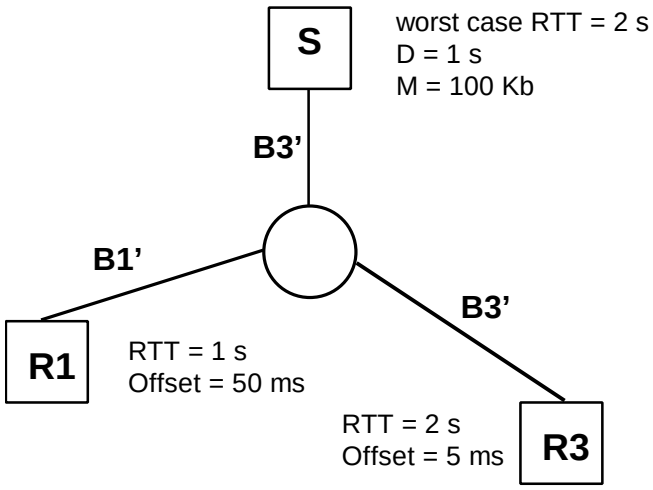

Figure 3. Stability after R2 left the session

4.2 Local estimations of RTT

The potential source implosion situation of the registration process is unavoidable because the process itself is unavoidable. The good news is that each receiver needs to do the process only once. Another potential source implosion situation has been avoided by the second protocol, when it requests the propagation delay of the receivers to detect variations in network conditions and in group membership. Only the receivers that are candidates to have a worse propagation delay than the current worst case are allowed to reply, thus minimizing the chances of source implosion and enforcing scalability, because also in large groups only a few receivers, if any, will reply.

Other source of potential source implosion that is not avoidable is the periodic update of the RTT values. It is important for the receivers to update these values periodically. For example, protocol SM-S detects variations in network conditions that may lead to a reduction in the delivery time. If the receiver's delay is not updated, the receiver will need to reserve more bandwidth than the previous time to meet the new smaller deadline. If the delay would have been updated may be they would reflect also

those improvements in network conditions. The periodic nature of the process may lead to periodic source implosion situations. One possible solution to this problem is to use local estimations.

With local RTT estimations, the communication is avoided and therefore the weakness can be eliminated from the design. The experience in TCP retransmission timer has shown that the estimation of the variance of the delay is as important as the estimation of the mean in order to obtain a reliable estimation [4]. Each receiver may therefore sense the network and update locally their delay with the source accordingly. The estimation can be computed using a low pass filter algorithm similar to the one described in [14].

However, an inaccurate local estimation of the delay may lead to either an inefficient reservation of bandwidth (more than needed), or to a miss of the deadline.

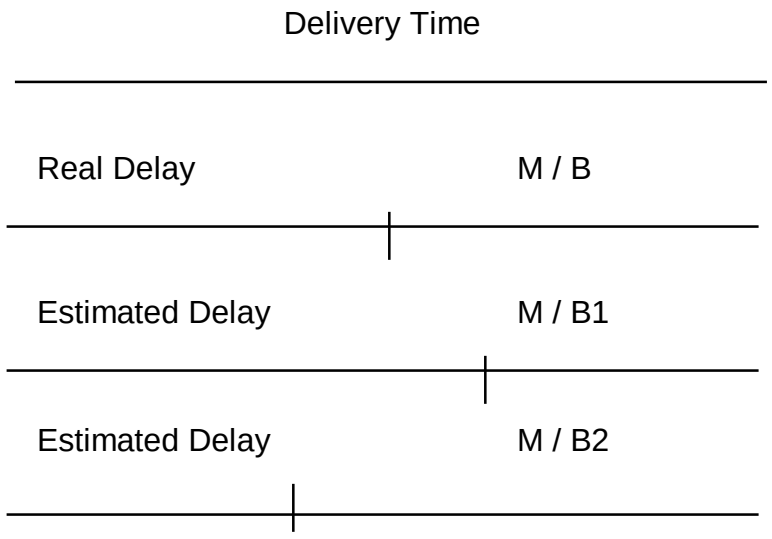

Figure 4. How delay estimations affect bandwidth reservations

In the figure above, the real delay will lead the receiver to request for B bandwidth. If the estimation of the delay is higher, higher bandwidth will be requested, and the deadline will be met, at a cost of some bandwidth inefficiency. But if the estimated delay is smaller, B2 bandwidth will be requested. The first bit will arrive later, and maybe the reserved bandwidth will not be enough to deliver the rest of the bits of the message in time. The priority is to avoid the second situation. Therefore a higher estimated delay value should be used to calculate the required bandwidth to ensure that the deadline will be met.

However, a periodic communication with the source may still be needed to correct the deviations in local estimations. But this communication has a bigger period and therefore the number of potential source implosion situations is greatly reduced.

4.3 SM applications without SM protocols

The overhead added by both protocols is minimum. In SM-R, the source uses one RSVP path message to indicate the flow, and the receivers use one RSVP reservation message. In SM-S, the source periodically uses one message to request the new worst case RTT. The frequency of the RSVP messages in SM-R depends on the period defined in SM-S. The value of the period depends on both the frequency of group leave, and the frequency of variations in network conditions. Finally, the overhead added by the update of the RTT value in each receiver is greatly reduced using local estimations, with both the timestamp request and timestamp reply messages sent with a higher period.

In contrast to that little overhead, the advantages seem to outplay them: without SM-R, the coordinated deadline can not be met. The source will have no knowledge about the farthest receiver. There is no way to the source to define an accurate flow spec because it does not know the worst case delay, it can only guess. The guessed worst case RTT may lead to several receivers not be able to meet the deadline. Of course if the guessed delay is big enough the deadline can be achieved, but the price is degrading the global quality, maybe to intolerable delays for some receivers.

It can also happen that the source implements only the initial part of SM-R: it has a mechanism to collect the propagation delay from the receivers and determine the worst case RTT. The source then will propose that receivers should reserve the same enough bandwidth to receive M bytes D seconds after the first bit arrived. Therefore, all the receivers will request a reservation of M/D bandwidth. Using the same values as in the example on Chapter 6, all links will have a reservation of B1 = 100 Kbps, as shown in figure 5.

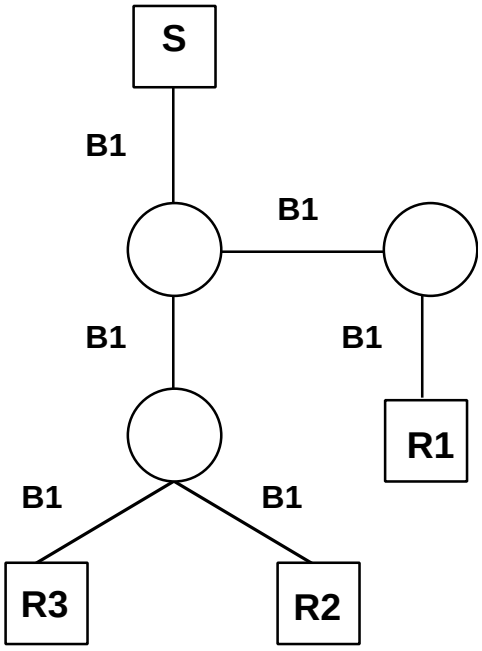

Figure 5. Reserved bandwidth without SM protocols.

SM-R provides more information to the source and the receivers. The source selects D taking into account the farthest receiver, and the receivers split the delivery time into propagation delay and rest of the delivery time, reserving bandwidth to only meet the rest of the delivery time. Therefore, all the receivers will request reservations according to the formula,

$$B = 2M / (\text{worst case RTT} - \text{current RTT in the receiver}) + 2D$$

which leads to lower bandwidths for all the receivers than in the case where the SM protocols are not used, except for the farthest receiver, for which the requested bandwidth is the same, M/D. Continuing with the figure above, we assume that R1 is the receiver most far away, with RTT/2 = 4 s. The source defines the delivery time as 5s. Figure 6 shows the bandwidth reservations. Comparing both scenarios, 100kbps bandwidth is reserved in all the links when no SM-R protocol is used, whereas only half of the links have that amount of bandwidth reserved. The bandwidth reserved in the rest of the links is smaller, leading to a more efficient resource reservation and a less loaded network.

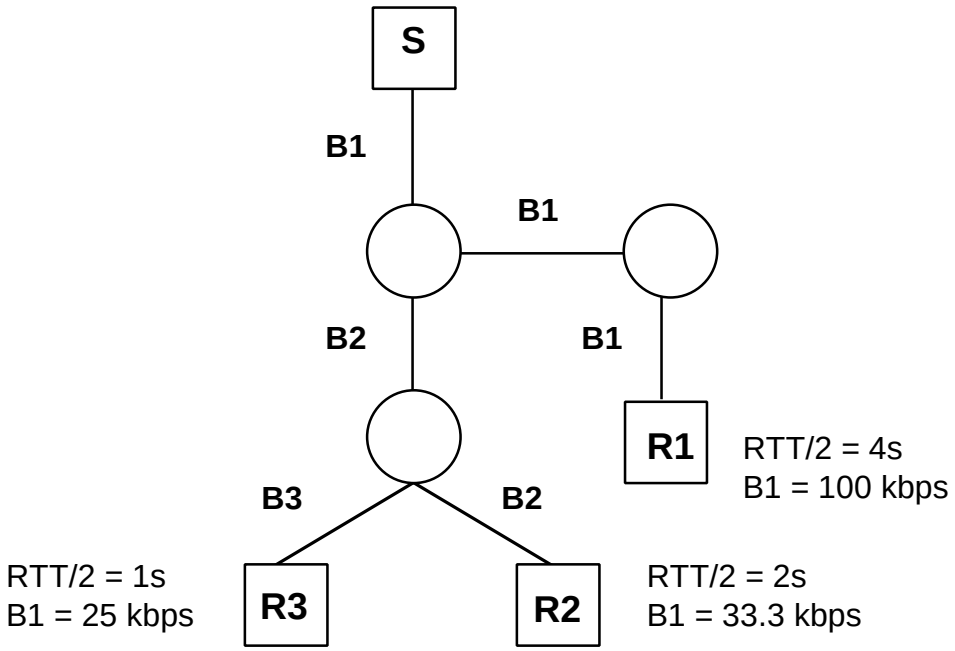

Figure 6. Reserved bandwidth with SM-R protocol.

In addition, the global quality of the applications will not be improved from the beginning of the session without using SM-S. That is, if the network and the receivers are not sensed, the source will not have a way to know that the receiver most far away has left the session. Only with the dynamic sensing provided by SM-S the global quality can be improved, providing a better service to the remaining receivers and opening the possibility to reduce the reservations on the network and reducing the load.

4.4 Buffering early messages

Protocol SM-R ensures a uniform deadline by controlling the delivery time with the bandwidth. The protocol also ensures fairness by defining the deadline using the RTT of the farthest receiver. But, if the guaranteed service provided by the network and requested through resource reservations only guarantees a bounded delay, an additional mechanism may be needed to continue maintaining the fairness goal. That is, if the

message arrives earlier, the message has to be buffered. To do that, the reception message must be timestamped in each receiver and compared with the expected delivery time. This mechanism is needed only for some applications where strict fairness is needed. If guaranteed delivery before a uniform deadline is enough, then buffering is not necessary.

Although an agent model can be used in the client side, where the agent acts as a liaison of the server granting a proper behavior, it may still be some concerns about how the client can be prevented from using earlier that information. One possibility would be to ask the network to provide this service. In this model, each router must have knowledge about the mean delay to the hosts in its local network, and to the adjacent routers. This information is currently available, because routing protocols store several variables on each link, such as delay or load[15]. Each packet may have a delay field, initialized to zero by the source. Each router can add the delay with the previous router in the path, and store the value in the packet. The last router will compare the delay field in the packet with the deadline (received in an RSVP path message), and it may delay the delivery of the packets if it is too early. This solution is complicated because it requires modifications in the routing algorithms. And it also adds complexity to them, which should be kept as simple as possible to allow them concentrate on doing their job: to route packets as fast as possible.

Chapter 5

Conclusions

5.1 Summary

Among others, multicast communications and resource reservation services have been adopted by the Internet to provide support for real-time applications, in its way to become a truly integrated services network. Those services are not enough for simultaneous multicast applications, where a coordinated deadline should be met, regardless of the network conditions, the heterogeneity of the receivers, and the differences between the clocks of the participants.

There are two main issues. First, it is not enough to have resource reservation capabilities to achieve a bounded delay that will guarantee the deadline. The deadline must be achieved coordinately among all receivers. Using our proposed protocol the source needs to have information about the delay with all the receivers to assure that the farthest receiver will also meet the deadline. The source will compute the flow specification to inform the receivers of the deadline knowing that all of them can meet it because it is defined to include the worst case. The receivers, upon reception of the deadline information, will request reservations accordingly. Depending on its own conditions they will reserve the minimum possible bandwidth that will guarantee the deadline for them. Each receiver will determine a different reservation. Closer receivers to the source will have a smaller propagation delay, and they have more remaining time than other receivers to receive the same amount of information. Receivers compute their reservations according to the formula:

$$B = 2M / (\text{worst case RTT} - \text{current RTT in the receiver}) + 2D$$

where M is the message size in bits, current RTT is the propagation delay for the receiver that computes the reservation, worst case RTT is the propagation delay for the farthest receiver, and D is the additional time determined by the source to allow the farthest receiver to meet the deadline.

Second, SM applications need to adapt themselves to dynamic network conditions and group membership. If the load of the network decreases or the farthest receiver leaves the session, there is a possibility to improve the global quality of the session by reducing the deadline or by reducing the bandwidth reserved. In both cases, receivers should recompute their reservations but the source is the responsible to provide

them with the necessary information to do that. Therefore, using SM-S, the source triggers the search for a better quality by sending a request message that it will only be responded by receivers with a worse delay than the current worst case.

These additional services have been introduced inside the SM applications using protocols. The thesis has presented the design of two protocols to meet the requirements of SM applications. The design accomplishes the stated goals: the first protocol meets the coordinated delivery time with controlled bandwidth being bandwidth-conscious, and avoids using clock synchronization. The second protocol achieves higher levels of global quality by sensing the network continuously and avoiding the use of an external feedback protocol. Additional goals are also met: fairness is achieved by defining the deadline to include the propagation delay of the farthest receiver; bandwidth-efficiency is achieved by both requesting the minimum necessary reservations to meet the coordinated deadline, and by controlling the value of the highest requested reservation with the message size and the D parameter; simplicity is achieved by avoiding the use and therefore the overhead of external protocols, and scalability is achieved with the second protocol by allowing to reply only those receivers who are candidates for the worst case propagation delay.

We have demonstrated that in the absence of the protocols, the coordinated delivery can not be achieved because the source can not determine the worst case delay and its guess can lead to receivers missing the deadline. If the source can obtain the delay information, the receivers will be able to meet the deadline at the cost of being bandwidth inefficient because all of them will reserve the same maximum necessary bandwidth.

5.2 Future Work

Many of the questions raised in this chapter and throughout the thesis can only be answered through simulations. This thesis does not include any simulation results or performance analysis because mainly of the difficulties we have found to find a network simulator with the support for RSVP. The simulator used to develop the original RSVP protocol back in 1992 is now too old because the protocol has been changed significantly since then[19]. The main research work with RSVP has been done by implementing and testing the protocol in several universities' testbeds.

One network simulator available is ns[20], which includes the support for multicast but not for RSVP, although it is very useful for monitoring and analyzing packet traces and queue behavior. We plan to study the performance using the RSVP module that will be released in early June 1998 for the PARSEC simulator developed at UCLA. PARSEC[21] is a C-based simulation language that can be used for sequential and parallel execution of discrete-event simulation models. In addition, PARSEC currently includes some networking modules such as TCP, IP and IP multicast. We expect to begin our simulations as soon as the RSVP module is released.

References

[1] Banerjea, A. "Multicast routing with QoS Support in the Internet" submitted for publication.

[2] Bolot, J., Turletti, T., Wakeman, I. "Scalable feedback control for Multicast Video distribution in the Internet" ACM SIGCOMM '94 Conference, London September 94.

[3] Huitema, C. "Routing in the Internet" Prentice Hall ISBN 0-13-132192-7, 1995.

[4] Jacobson, V. and Karels, M. "Congestion Avoidance and Control" Proc. ACM SIGCOMM '88, Stanford, CA, August 1998, pp. 314-329

[5] Mills, D "Improved Algorithms for Synchronizing Computer Network Clocks" IEEE/ACM Trans. Networks 3, 3 (June 1995), 245-254.

[6] Mills, D. Network Time Protocol (NTP). Request for Comments: 1305. Network Working Group. University of Delaware. March 1992

[7] Partidge, C. " A Proposed Flow Specification", RFC-1363, July 1992

[8] Partidge, C. "Gigabit Networking" Ed. Addison-Wesley Professional Computing Series ISBN 0-201-56333-9, 1994.

[9] Peng, C. S., Pulido, J. M., Lin, K. J., Blough, D. "The Design of an Internet-based Real-Time Auction System" submitted for publication to the IEEE Workshop on Dependable and Real Time E-Commerce Sytems (DARE'98), June 1998.

[10] Pulido, J. M., "Time Synchronization and Distribution in TCP/IP networks" Technical report, Dept ECE, UCI, 1998.

[11] Saltzer, J. H., Reed, D.P., Clark, D.D. "End-to-End arguments in System Design" MIT Laboratory of Computer Science, Second International Conference on Distributed Computing Systems (April, 1981) pages 509-512

[12] Schulzrine, H., Rosenberg, J. "Timer Reconsideration for Enhanced RTP Scalability", Proc. INFOCOM, San Francisco, CA, March/April 1998.

[13] Schulzrinne, H. et al "RTP: A Transport Protocol for Real-Time Applications" RFC-1889, January 1996

[14] Stevens , R. W. "TCP/IP Illustrated Volume 1, The Protocols" Addison-Wesley ISBN 0-201-63346-9, 1994.

[15] Tanenbaum, A. “Computer Networks” Third Edition. Prentice Hall ISBN 0-13-349445-6, 1996.

[16] Van Renesse, R., Birman, K., Von Eicken, T., Marzullo, K. “New Applications for Group Computing” Theory and Practice in Distributed Systems International Workshop, Dagstuhl Caste, Germany, September 1994.

[17] Zhang, L. et al “RSVP: A New Resource ReSerVation Protocol” IEEE Network Magazine, Vol. 9. No. 5, September 1993

[18] Zhang, L. et al, “Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification “ RFC 2205, September 1997, Proposed Standard.

[19] Zhang, L., private communication

[20] NS web page “http://www-mash.cs.berkeley.edu/ns/”

[21] PARSEC web page “http://may.cs.ucla.edu/projects/parsec/”