Wide Area Multimedia Stream Multicasting Using Active Networks

by

Aaron Lo

B.Sc. University of British Columbia 1998

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

Master of Applied Science

in

THE FACULTY OF GRADUATE STUDIES

(Department of Electrical and Computer Engineering)

We accept this thesis as conforming
to the required standard

The University of British Columbia

October 2001

In presenting this thesis in partial fulfilment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the head of my department or by his or her representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Department of ELECTRICAL AND COMPUTER ENGINEERING

The University of British Columbia
Vancouver, Canada

Date Oct 9, 2001

Abstract

This thesis addresses the difficulties in multicasting multimedia stream over wide area networks. Wide area networks are dynamic in nature. The number of clients is also dynamic. The number of clients can become arbitrary large and can have drastically different abilities and requirements. We identify the four intrinsic requirements for single source multimedia stream multicast in wide area networks. The four requirements are adaptation to dynamic conditions, scalability, light-weight setup, and accommodation of heterogeneous QoS requirements. We propose A-QoS-MM, a data dissemination and group membership service that meets the four requirements. A-QoS-MM is a soft-state reverse shortest path forwarding tree construction algorithm using active networks. The application level processing power provided by active networks gives A-QoS-MM an application level abstraction within the network. A-QoS-MM facilitates deployment of application specific processing within the network. Application specific processing and A-QoS-MM can provide a full set of multimedia multicast services. We tested A-QoS-MM on control, communication and computational costs. We found that control cost and total computational cost operate in an opposite direction with communication cost and average computational cost. We stated that a network configuration should favour communication cost and average computational cost.

Contents

List of Tables

List of Figures

Acknowledgements

I would like to thank my advisor Dr. Mabo Ito. Without his vision and continual financial support, this work would never have been possible. I would also like to show my appreciation to Dr. Son Vuong from whom I receive the initial inspiration in the area.

Finally, I am indebted to my aunt, May, for generously donating her computer on which most of the work is done.

AARON LO

The University of British Columbia
October 2001

To my parents, my family and my love for all their support during my study.

Chapter 1

Introduction

The Internet is growing. The number of hosts connected to the Internet is increasing at an exponential rate. According to the Internet Software Consortium[1], the number of hosts connected nearly doubles every year. This number is likely to continue its increase in the near future.

The Internet already connects a large number of hosts. The large number of hosts connects a large number of users. A large number of service providers are offering increasing number of services to attract business from this enormous market. These services include instant messaging, file transfers, the World Wide Web, interactive gaming, Internet phone, media conferencing, and digital media broadcast. Vast number of services are migrating to the Internet. The increasing number of services attracts even more users to the Internet. This self-reinforcing cycle is likely to continue in the near future and brings more users and services to the Internet.

Multimedia stream distribution over the Internet targets the enormous potential market and connectivity of the Internet. Multimedia stream distribution typically involves a single media source, such as a TV station, and multiple receivers, the TV

[1] http://www.isc.org

channel subscribers. Digital video/audio hardware and software are widely available. Live events and programs are beginning to be distributed over Internet. However, multimedia stream distribution has not been fully migrated to the Internet. The data distribution service for multimedia streams is too demanding for the Internet in its current state.

1.1 Motivation

Multimedia stream distribution over the Internet, and generally over wide area network, has many challenges. An arbitrary number of clients may request the multimedia service. Simultaneously, an arbitrary number of clients may leave the multimedia service. Each of the clients requesting the service may have to pay for the network access and/or the multimedia service. The clients may want some guarantee to the quality of service (QoS) such as data rate, service delay, and reliability. In terms of bandwidth, delay, reliability and topology, wide area networks, in particular the Internet, are heterogeneous in nature. The network load is also dynamic and different across the network. The network heterogeneity may cause the requested QoS to be drastically different. The above challenges break down into three fundamental requirements: dynamic requirement, scalability requirement, and heterogeneity requirement. This thesis later identifies the lightweight requirement. Many researchers have worked on dynamic and scalability aspects of multicast. Fewer researches have worked on the heterogeneity and lightweight aspects. This thesis tackles a subset of the above issues using active networking.

1.2 Our approach

Active networking is an emerging concept to network architecture. Active networks add intelligence inside the network. Network elements within active networks are given the ability to perform user-defined computations. Network problems can be treated within the network rather than at the network endpoints. Active network may also provide application level functionality within the network. The extra computation power appears promising for a wide range of applications such as network management and multicasting.

The thesis proposes a lightweight multicasting data dissemination and membership mechanism. It uses active networking to build application level multicasting trees. Active networking gives the mechanism processing power within the network. Application level abstraction is free from the details of the network. When combined, the raw processing power and application level abstraction can accommodate scalable membership, dynamic membership, and dynamic network conditions. Particularly, the mechanism builds active agents as multimedia gateways at strategic locations. These application level active agents have the ability to handled heterogeneous QoS requests.

The proposed mechanism itself does not provide the QoS guarantee. It does not provide reliability, ordered delivery, or packet repair functionality. The proposed mechanism only provide the underlying QoS aware multicast framework. It gives users the freedom to define more application specific functionality within the active agents.

The proposed mechanism is a generalized framework that does not depend on any particular active network architecture, any particular QoS type, or any particular underlying unicast routing algorithm. We simulate the mechanism using ns-2 from Berkeley. Control, communication and computational simulation data are gathered

to study the performance of the proposed mechanism.

1.3 Synopsis

In Chapter 2, we give a brief overview of active networking concepts. Chapter 2 gives the basic motivation of active networking. It presents general active networking approaches and architecture. It also discusses some of the issues involving active networking.

In Chapter 3, we study current multicasting approaches in detail. The chapter is organized in three parts. The first part presents steiner multicast approaches where multicast is treated as building minimal cost trees. The next part presents multicasting approaches for wide area networks. The last part presents active multicasting services that already make use of active network technology.

Chapter 4 is a short chapter that focuses on the heterogeneous nature of multimedia streaming. This chapter talks about the different types of heterogeneity for multimedia stream applications. The chapter also talks about some of the current approaches to these problems.

Chapter 5 identifies the four intrinsic requirements of a multimedia data dissemination service. The four intrinsic requirements are the dynamic requirement, the lightweight requirement, the scalability requirement, and the heterogeneity requirement. Chapter 5 also shows how the current multicasting approaches fail to meet the requirements.

Chapter 6 proposes an approach that aims to meet the multicast requirements from Chapter 5. This chapter gives a detailed description of the basic operations of the proposed mechanism. The chapter also demonstrates how the proposed mechanism meets the multicast requirements. Some simulation results follow a qualitative analysis.

Chapter 7 is the conclusion.

Chapter 2

Active networks concepts and approaches

The main topic of this research is on multicasting a multimedia stream in wide area networks. The multicasting mechanism to be presented in later chapters requires the active network technology. Active networks is an area of heated debate in itself. The work in this research is not intended to validate or evaluate active networks in its own right. This mechanism simply applies active networks. We investigate the mechanism in relation to some of the issues in active networks. This chapter will provide a background understanding and some of the issues of active networks. Section 2.1 presents the basic motivation of active networking. Section 2.2 describes the active network architecture and its two models. Section 2.3 discusses general issues of active network and some specific issues in relation to multimedia stream multicasting.

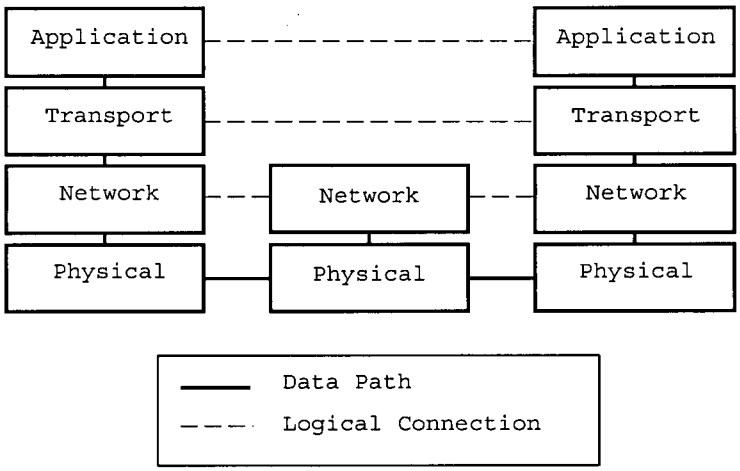


Figure 2.1: A simplified network layer architecture

2.1 Basic Motivation of Active Networking

A basic motivation of active network is the lack of flexibility of layering of protocols. The concept of layering of protocols has become the main principle for network architecture. Layering breaks down communication and abstracts functionality into a hierarchy. A higher layer always depends on the services provided by the immediate layer below. The higher layer accesses the functionality of the lower layer through the service access points. The implementation of each layer is abstracted and hidden from other layers. Each layer only has to communicate with its peer. A simplified network layer architecture is shown in figure 2.1.

Layering of protocols provides the abstraction in favour of network development. Layers can be implemented independently as long as the service access points remain unchanged. However, layering of protocol itself introduces new challenges. The independent implementation may lead to redundant functionality between different

layers. For example, in-order-delivery in transport layer may be redundant on top of an in-order-delivery network layer implementation. This leads to poor performance. Also, lower layer can provide only limited functionality with insufficient higher layer details. For example, the network layer would need representation of higher layer information, such as packet class and priority, to differentiate treatment of different packets. Lower layer performance may actually improve performance if higher layer information is accessible. In the layering approach, such details are encapsulated within each layer and not visible to other layers. One other challenge is the difficulty of introducing new services to the existing network architecture. New technologies and services often have to go through lengthy standardization process. The increasing number of applications would need new services quickly deployed.

Active networking aims to address some of the above issues. Some of the potential applications of active networks are network management, fault tolerance and error correction, QoS support, and multimedia synchronization. Applications that benefit from in network processing can be potential applications for active networks.

2.2 Active network architecture and approaches

The basic idea of active networking is introducing intelligence into the network. Internal network nodes are traditionally viewed as simple forwarding machines with limited capabilities such as routing, and forwarding. The communication end hosts need to take care of network issues such as congestion control. Some of these issues can be better handled inside the network. Problems can be fixed where they occur. The active network architecture gives the internal nodes the power to perform computation.

Computation in internal nodes can be achieved by defining a set of API for active nodes. The API should be accessible from all connected network elements, whether

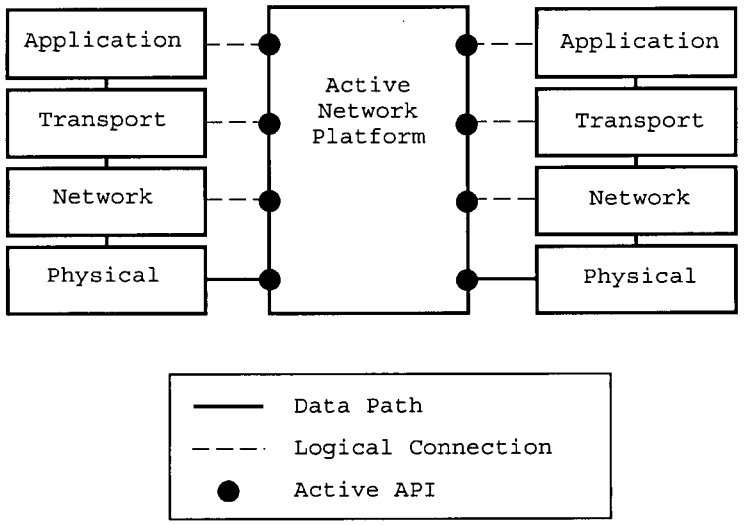


Figure 2.2: An active network architecture

end nodes or internal nodes. Depending on the capabilities of the API, third party organizations can design new network services. Some proposed active networking systems allow network nodes to perform computation as high up as the application layer. Active networks blur the boundaries inside the protocol stack. Information may flow vertically from network layer to application layer and horizontally through transport plane, control plane, and management plane. Figure 2.2 shows an active network architecture.

The active network can be alternatively viewed as the client/server model. Each active node within the network is a server. The API defines the service the active nodes provide. The set of potential clients may include active nodes and end nodes. An active node may request services from other active nodes. This property allows collaboration between multiple internal nodes. The availability of multi-layer resource and ability for multi-node collaboration open new space for network services design.

In [26], Tennenhouse et al. gives a survey of active network research and provides two primary approaches of active networks. The two approaches differ in the way programs and messages are treated.

The discrete approach of active networking separates the messages from the programs. Program code must first be injected into the designated internal active nodes. The appropriate program will handle all messages arriving at an active node after program injection. In other words, program injection modifies states in the active nodes.

The integrated approach of active networking does not distinguish between messages and programs. Messages and programs are abstracted into capsules. Capsules are self-contained communication entities. Each contains a program fragment and embedded data. Active nodes have an execution environment where the contents of the capsules are interpreted. In the integrated approach, a capsule has the ability to modify the state of the active node as well as the state of itself. This characteristic gives the capsules the ability to change the behaviour of active nodes and also the ability to modify their own contents.

Active network architectures are not limited to the two approaches. Individual architecture differs in detail aspects. Four attributes, defined in [8], can characterize active network architecture: underlying network technology, level of programmability, programmable communication abstraction, and architectural domain. Individual architecture and its characteristics are fields of study in their own right; however, they are not the subjects of study in this research. The multimedia multicasting mechanism in chapter 6 applies active networks on the abstraction level of the two approaches. This thesis only studies some of their general issues.

2.3 Issues of active networking

Allowing user processing in internal network nodes raises many issues. Security is one of the most important issues. Opening the API for internal network nodes imposes many potential security threats.

Intruders may want to acquire node resources without authorization. Intruders may inject programs or capsules into the active nodes and acquire resources not belonging to the intruders. Intruders may acquire computational resources. Intruders may acquire communication resources. Intruders may also acquire local active node states. The result of these intrusions includes but does not limit to denial of service attacks, eavesdropping, and theft of user information. A general approach to this issue is to carefully design the API with minimal security risk. Design of the secure API is coupled with the set up of a safe execution environment.

Interoperability is another major issue for active networks. Active network architecture faces two levels of interoperability problem. Within a specific active network architecture, the set of active nodes may use very different underlying technologies. All the different underlying technologies should support a common and technology independent API. This could be done using an architecture independent language such as Java or Tcl to access the API. The higher level interoperability between different active network architecture is more difficult to achieve.

Different architectures have different target applications and design goals. The different architectures have very different capabilities and APIs. It is very difficult to design a universal API suitable for all applications or to have the very different APIs talk with each other.

Efficiency is the third major issue for active networks. Although efficiency is also an issue for most other systems, it is especially important to active network. From the client/server perspective of active networks, the active nodes may need to serve

an arbitrary number of clients simultaneously. In addition, the active nodes should be able to support traditional non-active traffic, as non-active network elements should not distinguish between active elements and non-active elements. The active node API needs to be highly efficient. In general, compiled binaries are more efficient than interpreted language. Some active network architectures use compiled binary or byte code for efficiency.

The three issues are interdependent. Some relations are positive and some are negative. The API is the major factor that determines the performance in all three issues. API design must consider performance trade-off between the three issues.

The security requirement of multimedia multicasting mechanism proposed in chapter 6 is relaxed. Data dissemination of multimedia stream is not a security intensive service. The security of the multicasting mechanism relies on the underlying active network architecture and other third party security services. The interoperability requirement and efficiency requirements of the multicasting mechanism are less relaxed. To make any use out of active network, the elements within the architecture must interoperate. This is a fundamental requirement for all services on top of active networks. The multimedia mechanism also requires interoperability within the active network architecture. The multicasting mechanism also requires the underlying architecture to be efficient to handle the potential number of clients.

In extreme cases, an active network is programmable in all layers. This extreme case of active network architecture will have to solve the above three problems in all levels of abstraction. To enable some of the functionality of active network, and facilitate fast protocol deployment, [2] proposes the active service model, a subset of active networks. The active service model builds on top of the existing Internet network model. It uses the underlying network layer service to provide routing and data delivery. Application layer services can run in the active enabled elements. This active service model does not meet all design goals and does not solve all problems,

but it is useful for a subset of active network target application.

Chapter 3

Multicasting tree construction and group management

When two network nodes communicate, it is called unicast. When more than two network nodes communicate, the mode of communication is called multicast. In multicast, there can be multiple receivers and possibly multiple senders.

Multicast does not only involve data delivery for multiple recipients, but also involve the quality of the delivery. Multicast is a composite service. It is built from a number of sub-services. Each sub-service may provide one or more of the following features[1].

- Data dissemination
- Reliability mechanism
- Repair request
- Feedback control

[1]Reference [22] identified these features and use these features to classify multicast services.

- Retransmission
- Flow and Congestion Control
- Locus of Control
- Ordering
- Group Management

In this chapter, we give an overview of the multicast sub-services that are relevant to multimedia stream applications. Section 3.1 focuses on data dissemination using fundamental tree generation problems. It gives a definition of the steiner tree problem and discusses its approach for multimedia stream applications. Section 3.2 first describes IGMP, a group membership protocol for IP multicast. The section later describes three wide area data dissemination mechanisms: MBone, PIM, and QoSMIC. In Section 3.3, we look at existing active network multicast sub-services.

3.1 The steiner tree approach

Data dissemination in multicast services is often treated as routing data through a routing tree. This section focuses on routing tree construction problems.

3.1.1 Steiner tree problem

A communication network is often modelled as a graph where each vertex is a network element and each edge is a communication link. Many multicast data dissemination mechanisms organize the group members into a routing tree. The root (the source) sends data packets to each of its children and each child sends data packets to their children. This process is recursive and the packets pass through each edge of the

tree once and only once. The network cost is therefore proportional to the tree cost. Minimizing tree cost therefore minimizes the network cost. This reduces the construction of the multicast tree into the steiner tree problem. The steiner tree method aims to minimizes cost of the entire tree in contrast to the shortest path approach which aims to minimizes the cost of each path to the destination. The steiner tree approach is generally more suitable than shortest path approach for the multimedia streaming applications because of the high volume of data involved in the stream applications.

The steiner tree problem can be formally defined as follows. Let K be the number of vertices, $V = \{v_1, v_2, \ldots, v_K\}$ be the set of vertices, $e_{jk} = (v_j, v_k)$ be an edge between two adjacent vertices $v_i, v_j \in V$, $E = \bigcup\{e_{jk}\}$ be the set of existing edges, and $C(e) = c_e$ be a positive cost function defined $\forall e \in E$. A graph G can be defined as $G = (V, E)$. A tree $T = (V', E')$ is a subnetwork of G. The tree cost function is defined as

$$C(T) = \sum_{e' \in E'} C(e')$$

Pick $r \in V$. For a set of vertices $V' \subseteq V - \{s\}$, the steiner tree problems is to find the tree T_{min} such that $C(T_{min})$ is minimum $\forall T$ rooted at r spanning V'. In the context of communication network modelling, V' is multicast group. r is the data source. The steiner tree problem is proven to be NP-complete.

For multimedia stream applications in wide area network, two new requirements arise. Data must reach all members of the multicast group within some specified delay bound. The structure of the tree must also reflect the changes in the dynamic network. The rest of this section discusses various techniques to construct delay bounded multicast trees and various techniques to accommodate dynamic multicast group membership.

3.1.2 Meeting the delay bound

The delay-bounded variant of the steiner tree problem introduces a new positive delay function $D(e) = d_e$ defined $\forall e \in E$. The path between two nodes v_1 and v_k is defined as the sequence $P(v_1, v_k) = \{v_1, e_1, v_2, e_2, \ldots, e_{k-1}, v_k\}$ where e_i is the edge between two adjacent nodes v_i and v_{i+1}. The cost of a path is defined as $C(P(v_1, v_k)) = \sum_{e \in P(v_1, v_k)} C(e)$. The delay of a path is defined as $D(P(v_1, v_k)) = \sum_{e \in P(v_1, v_k)} D(e)$. There may be a number of possible paths $\{P_1(v_1, v_k), \ldots, P_n(v_1, v_k)\}$ for a given pair of vertices v_1 and v_k. The shortest delay path from v_1 to v_k, denoted $SDP(v_1, v_k)$ must satisfy

$$D(SDP(v_1, v_k)) = \min\{D(P_1(v_1, v_k)), D(P_2(v_1, v_k)), \ldots, D(P_n(v_1, v_k))\}$$

The shortest cost path SCP is defined in the analogous fashion. Each node v_i in V' has a positive delay bound Δ_i. A delay-bounded steiner tree problem is to find the minimal cost spanning tree for V' where $\forall v_i \in V', D(P(r, v_i)) \leq \Delta_i$, and $P(r, v_i)$ is a path in the routing tree from the root r to $v_i \in V'$.

Some existing heuristics, such as in Reference [21], are centralized. Some others, such as in Reference [19], are distributed. However, most are variants of the greedy algorithm where the cheapest path from the tree node to the new node gets constructed sequentially. Reference [21] defines two selection functions, f_C and f_{CD}, to provide two heuristics. For w a new node to be added to the tree, v a node already in the routing tree, and v and w satisfies $D(P(r, v)) + D(P(v, w)) < \Delta$,

$$\begin{aligned} f_C &= C(P(v, w)) \\ f_{CD} &= \frac{C(P(v, w))}{\Delta - (D(P(r, v)) + D(P(v, w)))} \end{aligned}$$

The heuristics choose the cheapest path given by the respective selection function from the possible set of v and w. The f_C heuristic optimizes cost while the f_{CD}

heuristic optimizes delay at the expense of added cost. A distributed version of the heuristics in Reference [21] exists but the selection policies remain the same.

The distributed heuristic in Reference [19] also uses the idea of minimizing cost but employs a different method of obtaining the cheapest path. A tree node u is said to be the tree node closest to a non-tree node v if the shortest path from u to v in terms of cost between nodes u and v, denoted by $SCP(u, v)$, satisfies,

$$\text{for any tree node } k: \sum_{e \in SCP(u,v)} C(e) \leq \sum_{e \in SCP(k,v)} C(e)$$

$SCP(u, v)$ is said to be the shortest path from the tree to v. The cost of a tree T to a non-tree node v is defined as

$$C(T, v) = \sum_{e \in SCP(u,v)} C(e)$$

where u is the tree node closest to v. A non-tree node w is said to be closest to a tree T when

$$\text{for all non-tree node } v: C(T, w) \leq C(T, v)$$

The CAO algorithm, proposed in Reference [31], falls in the line of greedy algorithms. The least cost path from r to a new node v is selected with the section of the path already in the tree be 0.

A few non-greedy algorithms exist. Reference [25] proposes a fully distributed algorithm. A member of the multicast group w sends request to one of the tree nodes v_0. v_0 then distributes the request to suitable nodes within the tree $\{v_1, v_2, \ldots, v_n\}$, which satisfies $D(P(r, v_i)) + D(P(v_i, w)) \leq \Delta$. All suitable nodes will send responses back to w.

Another non-greedy approach is bounded shortest multicast algorithm (BSMA) from Reference [32]. The algorithm builds a least delay tree as well as a least cost tree. The least delay tree satisfies the delay requirement for member of the multicast group.

The algorithm iteratively replaces the section of the path between two branching nodes with the section of the path connecting the two nodes in the least cost tree within the delay requirement.

3.1.3 Accommodating dynamic membership

In the dynamic variant of the steiner tree problem, the multicast group V', change dynamically. The dynamic variant can be regarded as a sequence of static steiner tree problem. The problem is formally defined in Reference [17]. Let $R = \{r_0, r_1, \ldots, r_n\}$ be a sequence of requests where each $r_i = (v_i, \rho_i), v_i \in V, \rho_i \in \{join, leave\}$. The ith request adds to or remove from the multicast tree the node v_i. V_i' denotes the resulting multicast group after step i. The dynamic steiner tree problem is to find a sequence of tree $\{T_1, T_2, T_3, \ldots, T_n\}$ where T_i is the minimum spanning tree of V_i'.

One approach to the dynamic steiner tree problem is to apply static steiner tree heuristics at each step. This approach rebuilds the entire multicast tree at each step. Solving the problem in a sequence of static steps is not a solution to the dynamic steiner tree problem for three reasons. For one reason, solving at each step is very expensive. Another reason is that a new request r_{i+1} may arrive before construction for T_i completes. The last reason is that reconstructing the multicast tree may disrupt the flow of continuous data stream. Existing heuristics do not build the entire tree at each step. Existing heuristics deal with the effect of adding and deleting nodes only locally.

"*join*" requests add nodes in the multicast group. Adding new nodes can be handled sequentially. "*leave*" requests delete nodes from the multicast group. If the deleted node, v_i, is a leaf node, then the algorithms can simply delete the branch from the rest of the tree to the node. If v_i is in a path to a subtree, the GREEDY algorithm from Reference [17] will turn v_i into a relay node. The GREEDY algorithms do not

optimize the cost on deletion.

When a "*leave*" request arrives, the restricted dynamic greedy algorithm (R-DGA) proposed in Reference [1] will remove the associated tree node and all its paths to its parent and children . R-DGA will re-add the children nodes back into the multicast group in the same way "*join*" requests. In essence, R-DGA performs partial rebuilding of the multicast tree at each deletion. R-DGA uses IP tunnelling and involves only the members of the multicast group to build the tree, but the idea of partial rebuilding can extend beyond this restriction.

Rebuilding a partial tree in every step may still be a very expensive operation. ARIES from Reference [6], and CRCDM from Reference [23] rearrange regions of the multicast tree only when triggered. When a region of the multicast tree accumulates damages beyond a threshold, the rearrangement event triggers. For ARIES, the damage is measured as the number of nodes recently added or deleted in the region. CRCDM extends the idea of ARIES with the notion of "contribution". The contribution is represented by the number of multicast group members connected to the source through the region. Reference [23] argues that a region should not be disturbed if it serves a large number of members. In CRCDM, a region will not rearrange even when the damage grows beyond the threshold if the contribution of the region is relatively large.

This section does not provide an exhaustive list of steiner tree heuristics. This section provides only the overview and theoretical background of the heuristics relevant to the application domain. The analysis on the classes of heuristics relevant to the application domain will follow in Chapter 5.

3.2 Wide area multicast mechanisms

This section concentrates on the group membership model and data dissemination available for wide area networks.

3.2.1 IP multicast and Internet Group Management Protocol

The IP multicast is an multicast enabling extension to IP. Reference [14] first proposes IP multicast and discusses the detail of the extension. In contrast to the materials in 3.1, Reference [14] does not discuss multicast routing but focuses mainly on the multicast addressing, IP extensions, and group management. IP multicast leaves data dissemination to individual router implementations.

Internet Group Management Protocol (IGMP), also proposed in Reference [14], is the group management protocol for IP multicast. The most important property behind IGMP is that it is receiver-driven. Multicast enabled routers should only forward multicast packets to the attached local area network (LAN) only if there are multicast recipients on the LAN. Multicast enabled routers learn the presence of multicast recipients by broadcasting IGMP queries on the LAN. When a multicast recipient receives a query, it replies with an IGMP report with a join request.

Multicast enabled routers forward multicast packets only to the nodes declared as multicast recipients. Group management in this receiver-driven model involves only the multicast recipients and their local multicast enabled routers. No central node needs to manage all the members.

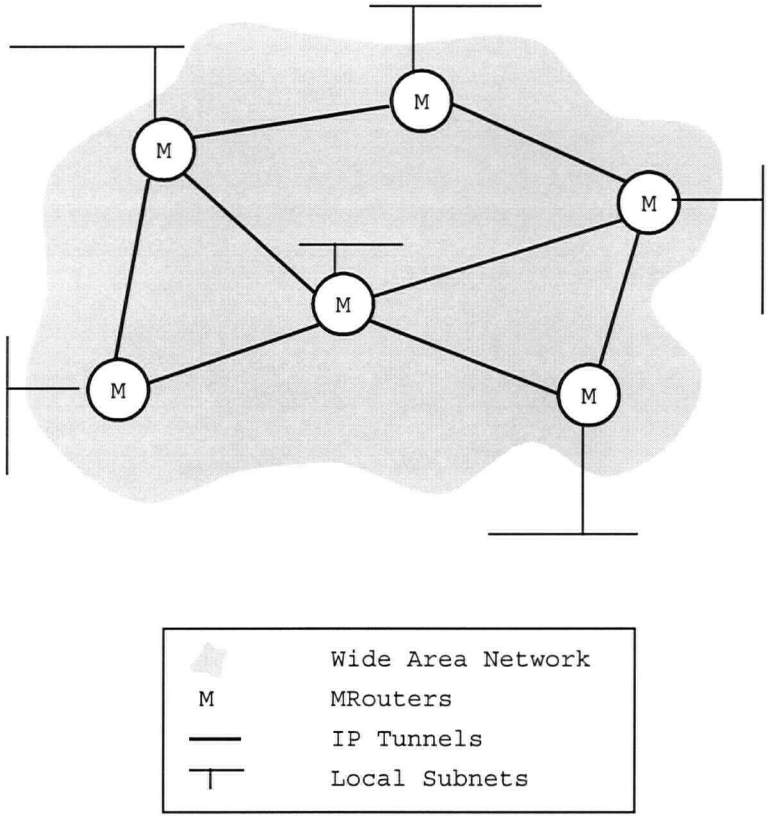


Figure 3.1: An imaginative region of MBone

3.2.2 MBone

MBone from Reference [9] is the short name for the Multicast Backbone. MBone is a virtual network on top of the Internet that provides the multicast connectivity between subnets. All multicast groups using the MBone share this virtual network. Figure 3.1 illustrates the topology of MBone. MBone currently connects more than one thousand subnets.

MBone connects islands of multicast enabled subnets with IP tunnels proposed in

Reference [24]. IP tunnels are unicast routes. MBone runs distance vector multicast routing protocol (DVMRP), a shortest path multicast routing protocol from Reference [28], on the islands. Data carried by the tunnel are delivered but not processed between the end points of the unicast route. In MBone, the multicast state carried in the payload bypasses the intermediate non-multicast routers. Only the MBone capable routers (MRouters) need to process the multicast state. Tunnelling allows MBone to scale into the wide area network.

MBone is available only to the subnets where MRouters are installed. For hosts residing outside the MBone service area, the hosts would need to establish tunnels to their nearest MRouters. MBone also have a relatively fixed topology. MRouters do not change their tunnels connecting to the MBone very often.

3.2.3 Protocol Independent Multicast

The protocol independent multicast (PIM) is proposed in Reference [13]. The sparse mode of PIM (PIM-SM) aims to provide good scaling property for sparse groups. A sparse group is a multicast group where most network elements do not participate in the multicast and the multicast group members span a wide area. Shortest path routing algorithms such as DVMRP fails to scale for sparse groups because all network elements must participate. PIM-SM scales well for sparse membership.

Another fundamental design goal of PIM-SM is to support high quality data distribution. PIM achieves this goal by providing flexibility for each group to build the distribution tree according to the target application. A low data rate application with many senders would benefit from the scalability of a shared tree such as CBT from Reference [4] whereas a high data rate low delay application with a small number of senders would benefit from the low delay of a source-based tree.

PIM-SM is independent of any particular unicast protocol but uses the service of

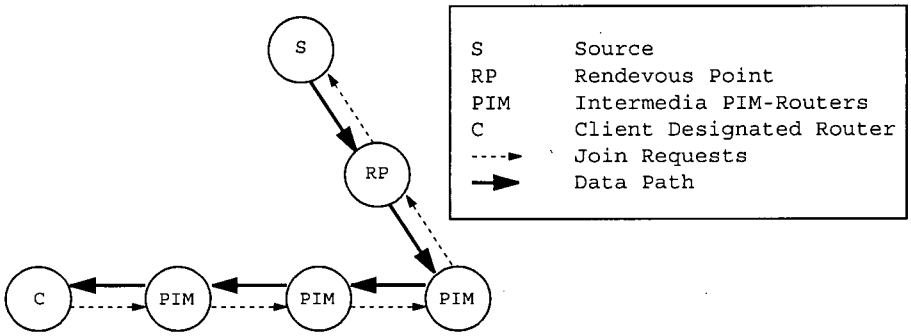


Figure 3.2: Single client subscribing in PIM-SM

the unicast protocol to adapt to topology changes. PIM-SM is also a robust protocol as it uses a soft-state mechanism. PIM-SM avoids a single point of failure and the constructed tree adapts gracefully with dynamic membership.

Each multicast group sets up one or more *rendezvous point(s)* (RPs). When a receiver wish to join a multicast group, the local PIM capable router sends a PIM join request to one of the RPs of the group. The intermediate PIM-routers between the local PIM-router and the RP sets up the distribution tree. When the join request reaches the RP, the RP sends a join request toward the source of the group. The PIM-router in the source subnet will respond with a PIM register request. The intermediate PIM-routers between the RP and the source then sets up a data path. The data path from the source and the data path to the receivers meet in the RPs. Figure 3.2 gives an example of PIM-SM construction from a single client.

This construction ends up in a shared tree. When the receivers learn about the source, the receivers then can establish source-based tree if one is required. The receivers can send PIM join request to the source. The intermediate routers will then prune the path from the receivers to the RPs. Both the shared tree construction and

source-based tree construction uses reverse path forwarding from Reference [12].

3.2.4 Quality of Service sensitive Multicast Internet protoCol

Quality of Service sensitive Multicast Internet protoCol (QoSMIC), proposed in Reference [16], is a multicast dissemination mechanism that considers the quality of service (QoS) in its design. Like PIM-SM, QoSMIC is also independent of any particular underlying unicast protocol but uses the underlying protocol to adapt to topology change. QoSMIC also can construct different tree types according to different applications. QoSMIC finds the path using reverse path forwarding. Only the necessary nodes are involved in the tree construction thus QoSMIC scales well in sparse group. Most importantly, QoSMIC provides multiple routes for a single join request. The receiver could select the route best-suited QoS requirement.

The QoSMIC join operation consists of two procedures: the local search procedure and the multicast tree search procedure. The local router designated for the joining member initiates the local search procedure by flooding a probe message within its neighbourhood specified by the probe message time-to-live value. All routers in the tree that receives the probe message are considered candidates. The candidates unicast BID messages back to the local designated router.

The multicast tree search requires a manager router. The management router does not have to be the root of the tree. This design choice limits the effect of topology on the performance of the protocol. When the manager fails, the protocol can easily recover because the manager is not in the critical path of data. The local designated router sends a join message to the manager router. For each group, the manager router may choose to perform a centralized selection or a distributed selection. For centralized selection, the manager router must have detail knowledge of the topology.

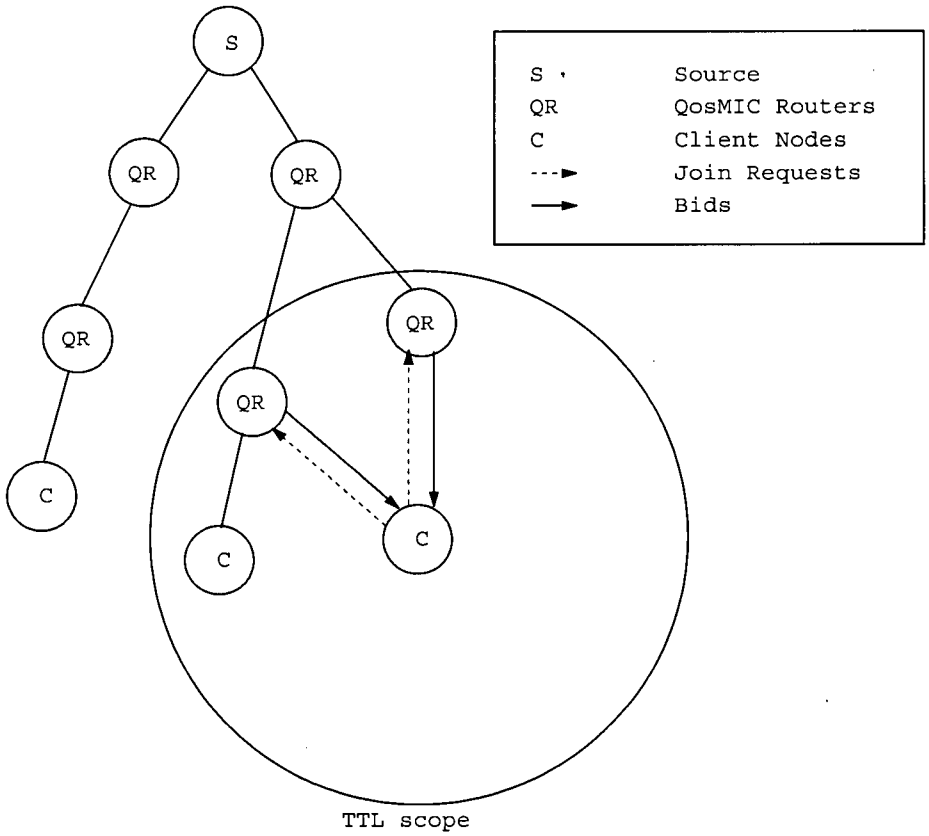
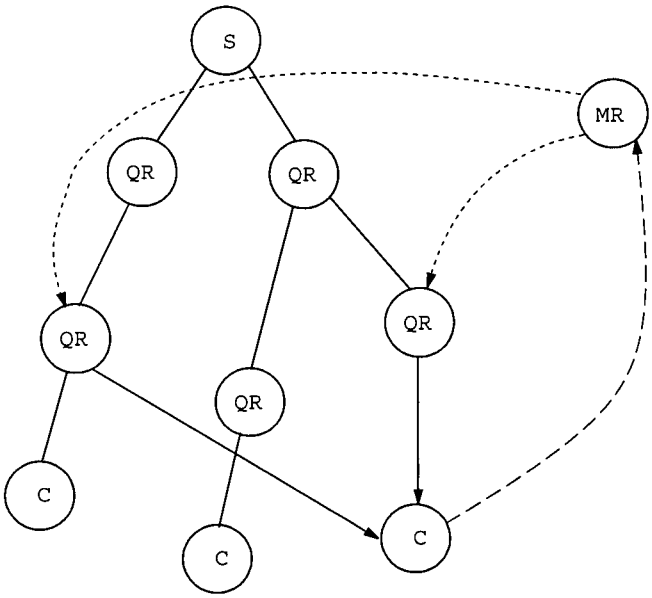


Figure 3.3: The QoSMIC local search



S	Source
MR	Manager Router
QR	QosMIC Routers
C	Client Designated Routers
—►	Tree Search Requests
---►	Bid-orders
——►	Bids

Figure 3.4: The QoSMIC tree search

With its internal knowledge, the manager finds and unicasts BID-ORDER messages to suitable tree routers. The tree nodes that receive the BID-ORDER are considered candidates and unicast BID messages to the designated router similar to the local search. In the distributed case, the manager distributes the BID-ORDER along the tree. The tree nodes can use a distributed selection mechanism to determine the set of candidates. The choice of distributed selection mechanism should consider the network topology and traffic behaviour. Figure 3.3 and Figure 3.4 demonstrate the local search and the tree search respectively.

For both the local search and the multicast tree search, a tree router may receive BIDs from within the tree. This situation implies that the tree router must locate on the route between the BIDder and the receiver. The tree router is a closer tree node to the receiver. The tree router will drop the received BIDs and issues its own BID.

After the local search procedure and the multicast tree search procedure. The designated router can select the best route according to the BIDs. The leave request is handled using the same operation as the join request. The tree nodes then modify the tree according to the membership change.

3.3 Multicast using active networks

One of the design goals of active networking is to provide computation within the network. Multicast, as delivery service for multiple recipients, can benefit from the extra in-network computation active networks provide. A number of multicast services are already proposed for active networks. More specifically, these multicast services are value-added services on top of the network layer. These multicast services can be implemented in the active service model. Furthermore, these services degrade gracefully for lower number of active elements. The first active multicast service is a basic data dissemination mechanism. A reliable multicast service follows. This section ends with

a reliable and congestion control service.

3.3.1 Basic data dissemination mechanism

First, we examine a basic data dissemination mechanism. This mechanism first appears in Reference [29] as a sample application for active networks. Similar to the source-based tree in PIM-SM and QoSMIC, the basic data dissemination mechanism constructs the tree using reverse path forwarding. Each receiver sends *subscribe* active packets toward the source. The active *subscribe* packets travels in the reverse path.

Intermediate active nodes that receive the active *subscribe* packets set up forwarding pointers in the direction of the subscriber and forward *subscribe* packet upstream. Data packets can later distribute through the forwarding pointers. This mechanism is soft-state and the forwarding pointers need periodic refreshment. The mechanism has the property of gracefully tree pruning.

The original mechanism does not deal with the redundant active *subscribe* packets arriving at the source. Reference [5] later extends the mechanism to avoid this *subscribe* implosion problems. Intermediate active nodes set up the forwarding points but do not forward redundant *subscribe* packets upstream.

3.3.2 Active Reliable Multicast

Active Reliable Multicast (ARM) does not deal with the issue of data dissemination. ARM makes use of local storage and computational power available in the active service model. ARM provides best-effort caching, scoped retransmission, and NACK fusion/suppression, to support reliable multicast service.

ARM caches data packets for possible retransmissions. When the data packets pass through intermediate active nodes, the intermediate active nodes stores the data

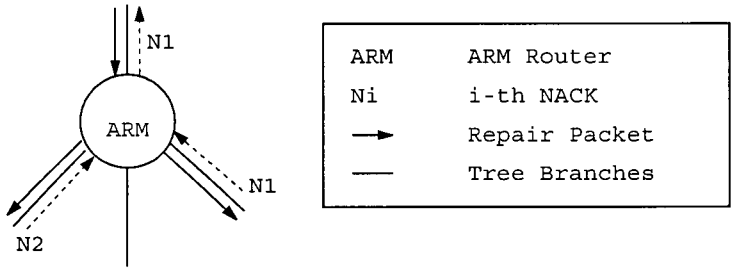


Figure 3.5: ARM NACK Suppression

packets, including lost packet repairs, in their local storage. The lifetime of the cache depends on the data rate and the maximum round trip time. When a receiver detects a packet lost, it sends a negative acknowledgement (NACK) for that packet upstream. An intermediate active router that receives the NACK retransmits the packet if the intermediate router has the lost packet in cache. Retransmission requests no longer go all the way back to the source. In network retransmission significantly reduces the repair latency for lost packets.

The intermediate active routers also provide NACK suppression. The active intermediate active routers maintain states for NACKs received and repairs retransmitted . When an intermediate active router receives a NACK for a packet that the router has not cached, the intermediate active router goes through its state and determines if a NACK for the same lost packet has already been received. The router also remembers where the NACK came from. The intermediate active routers always send the first NACKs upstream and discard all duplicated NACKs. All repairs are forwarded only down the link where the respective NACKs has been received. This selective repair forwarding constitutes a local, scoped retransmission scheme. Figure 3.5 illustrates the NACK suppression operation for ARM. For both NACK suppression and scoped retransmission, no communication takes place unless necessary.

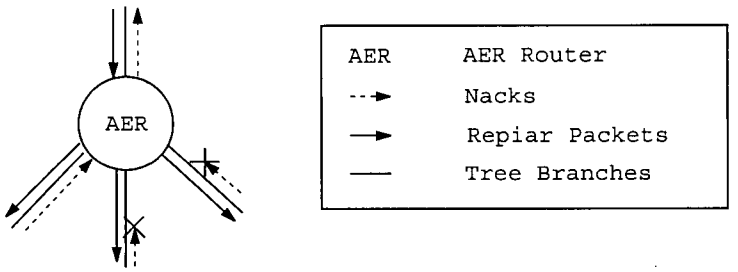


Figure 3.6: AER NACK suppression

3.3.3 Active Error Recovery and Nominee-based Congestion Avoidance

Active Error Recovery (AER) and Nominee-based Congestion Avoidance (NCA) are both proposed in Reference [20]. Both use the storage and computation ability of the active service model. Together, AER and NCA provide a scalable and reliable multicast service.

AER has similar design goal as ARM: to provide best-effort caching, to provide scoped retransmission, and to provide NACK suppression. Caching and scoped retransmission in AER are similar to their counterpart in ARM. AER, however, introduces random timers for NACK suppression. When a receiver or an intermediate active router detects a packet lost, it waits a random amount of time, issues a NACK upstream, then starts a NACK retransmission timer. If a NACK from upstream arrives before the NACK is issued in the downstream, the downstream node suppresses its NACK. When an intermediate active router receives a NACK for a packet it is unable to repair, the intermediate router subcasts the NACK one level downstream and start requesting a repair for itself. Figure 3.6 shows a scenario of AER NACK suppression. The difference in NACK suppression between ARM and AER is that

ARM does not propagate duplicated upstream NACKs but AER prevents upstream NACKs from issuing altogether. This downward suppression helps prevent the non-active part of the tree from NACK implosion problem.

NCA is a protocol that regulates a uniform congestion window for the entire multicast group. In particular, NCA is "TCP friendly" that the multicast session bandwidth is no bigger than competing TCP sessions. Periodically, receivers issue congestion status messages (CSMs) upstream. CSM includes some receiver details that get evaluated according to criteria given in Reference [20]. The CSMs go through a tournament-like selection upstream and the greatest denominating receiver (the worst receiver) becomes the nominee for congestion control. A TCP like rate adjustment algorithm takes over and starts regulating a uniform congestion window based on the level of congestion between nominee and the source.

Chapter 4

Multimedia in wide area networks

Besides data distribution, multimedia stream applications has another major difficulty to deal with. Receivers may request data at different rates. Receivers may also request data in different formats. This chapter dedicates to the heterogeneous requirements and their solutions. Section 4.1 gives a general approach to multimedia applications in wide area network. Reference [27] identifies and discusses two approaches to the specific heterogeneous problems. Section 4.2 gives an overview of layered video, a solution to the heterogeneous rate problem. Section 4.3 talks about multimedia gateways that deals with the heterogeneous format problem.

4.1 Application level framing and light weight session

Like active networks, application level framing (ALF), introduced in Reference [10], is an alternative architectural approach to the structured protocol layering. Layered structuring is not targeted to any particular application domain, and, therefore, layered structuring may constrain the performance of some application specific systems.

On the other hand, application level framing defers the engineering decisions later to avoid unnecessary constraint.

ALF requires application specific semantics in the application level protocol design. Application data are segmented and delivered in application data units (ADUs). These ADUs are stand-alone data units. They are the smallest unit manipulation. The application then is given the power to deal with error recovery, packet ordering, and other details.

Light-weight session (LWS) from Reference [18] is an extension of application level framing in the real-time multimedia domain. In LWS, senders just send data to the multicast groups. Receivers interested in a particular multicast session announce their interest using IGMP and IP multicast takes care of data delivery. It is up to the receivers to adapt to the different data rates and handle packet lost. Applications can implement rate adaptation and loss tolerance policies according to their specific requirements.

ALF and LWS do not solve the heterogeneity problem explicitly but they have an important implication. Heterogeneous requests may be better handled in a higher and possibly more efficient layer.

4.2 Heterogeneous rates and layered encoding

In wide area networks, receivers of a single multimedia session may have very different capabilities (in terms of bandwidth, buffer spaces, etc.) The heterogeneous rate problem is the problem of accommodating different data rates within a single multicast session. A multimedia source can adjust the output rate to an appropriate rate. If the output rate is set at a level that no congestion occurs at all branches, then the receiver of lowest capability limits quality received by higher capability receivers. On the other hand, if the output rate is set to satisfy the requirement of a portion of

receivers, then some lower capability receivers will experience congestion. Monotonic rate transmission is not a good solution to the heterogeneous rate problem.

Layered encoding is an alternative approach to the heterogeneous rate problem. Layered encoding splits a single multimedia flow into a base flow and hierarchical enhancement flows. The base flow can be decoded into a low quality video or audio. The higher layer flows add enhancement to the quality of lower flows. This implies that higher layer flows is only meaningful if lower layer flows are correctly received. Each flow has different bandwidth requirement. The source can send each flow over a separate multicast session and a receiver with a particular capability can subscribe to the suitable set of multicast sessions. Reference [27] points out that corresponding packets in the higher layers are useless when a lower layer packet is lost.

Layered encoding would benefit from two network functions: selective discards and prioritizing flows. The intermediate nodes should selectively discard meaningless corresponding higher layer packets when a lower layer packet is lost. The intermediate nodes should also give higher priorities to lower layer flows to support minimal quality delivery.

4.3 Heterogeneous formats and multimedia gateways

For a single multimedia flow, different receivers may request multimedia service in different formats. Multimedia gateway addresses the heterogeneous format requirement for wide area network receivers.

Multimedia gateways typically take flows of one encoding and forward flows in another encoding. Multimedia gateways also transcode streams with the same encoding but with different parameters. A procedure for video transcoding involves

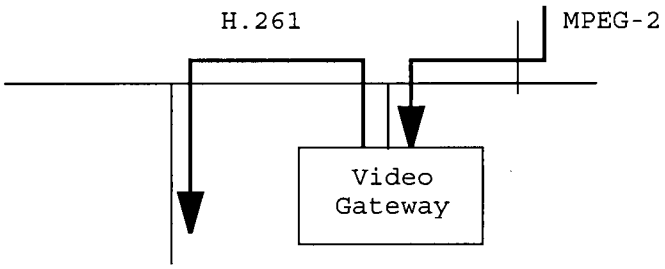


Figure 4.1: Video gateway transcoding from MPEG-2 to H.261

performing decoding, de-quantizing, and reverse discrete cosine transform (reverse DCT) of the original flow. The procedure then redoes the discrete cosine transform (DCT), quantization, and encoding in the new encoding scheme. For optimization, this transcoding procedure may sometime omit the reverse DCT and the DCT as in Reference [3]. Figure 4.1 shows a video gateway at work.

In addition to performing transcoding, a multimedia gateway can connect two multimedia sessions (one from upstream and one from downstream) into a single logical conference transparently. To achieve this, the multimedia gateway converts and aligns for synchronization and control information across the two sessions.

The difficulty of applying a multimedia gateway is the offline manual setup. Multimedia gateways do not service local receivers on demand. They are manually configured at fixed locations independent of individual sessions. This manual offline setup does not facilitate efficient strategic placement of gateways for all sessions.

Chapter 5

The wide area multimedia heterogeneous multicast problem

In a wide area network, there are intrinsic requirements that make multicasting difficult. In this chapter, we will look at the wide area multimedia heterogeneous multicast problem and discuss how the approaches in Chapter 3 failed.

5.1 The intrinsic requirements of wide area network multicast

There are four main requirements for wide area network multimedia multicast: dynamic, scalable, light-weight, and heterogeneity. These are the four intrinsic requirements. A particular stream may have additional QoS requirements such as delay and data rate. QoS requirements are application and stream specific and they are often restricted by physical properties of the network. The above four requirements are fundamental for all wide area network multimedia services.

A wide area network multimedia multicast service should adapt to changing net-

work conditions and changing membership. The dynamic requirement gives rise to two restrictions. A wide area network is dynamic in nature. The load of the network changes. The topology of the network changes. A wide area multicasting mechanism must accommodate dynamic network conditions. Multimedia application is also dynamic in nature. Subscribers to a multimedia service may join and leave the session at will. The multicast mechanism should also take care of dynamic membership.

A wide area network multimedia multicast service should have reasonable performance for an arbitrary number of clients. The scalability requirement gives rise to two other restrictions. For a single multimedia session, there can be an arbitrary number of subscribers. It is important for a multimedia multicasting scheme to scale to large number of receivers. The multicasting mechanism should be receiver driven. i.e. the receiver initiates the multicast rendezvous procedure. The multicasting mechanism should also be distributed. The source node is intrinsically the single point of failure, but the mechanism should not have a performance bottleneck at a central processing node.

A wide area network multimedia multicast service should require minimal setup control. Two additional restrictions arise from the light-weight requirement. According to the light-weight session model, receivers simply announce their interest to join the group. Receivers have no prior knowledge of group membership. Furthermore, receivers have no prior knowledge of network condition. The light-weight requirement conforms to the dynamic requirement as receivers should not need to handle dynamic membership and dynamic network conditions. The light-weight requirement also conforms to the scalability requirement as the active nodes maintain relatively few control state in comparison with group membership and network size.

Finally, a wide area network multimedia multicast service should be able to manage drastically different QoS requirements. The heterogeneity requirement is analogous to the heterogeneity problem in Chapter 4. The multicast scheme must

simultaneously support different levels of QoS required by different receivers. QoS may include reliability and data rate. Although generally accepted as a requirement for multimedia transport, bounded delay between receivers and the source is relaxed. Because real-time multimedia streams are continuous, the receivers begin to receive streams of data when they attach to the branches of the dissemination tree. The receivers are unaware of source-to-receiver delay. The receivers are more interested in the promptness of multimedia service and can tolerate some delay from the source. For real-time 1-to-n multimedia multicast, bounded delay between receivers and the tree is sufficient.

The wide area multimedia multicast problem is to find a 1-to-n multicast mechanism that satisfies all the above requirements.

5.2 Some inadequate solutions

This section demonstrates how each multicasting scheme from Chapter 3 fails to meet the wide area multimedia multicasting requirements.

The steiner tree approaches from Section 3.1 are promising for dynamic membership, scalability, and delay constraint requirements. Distributed delay constrained steiner tree heuristics with dynamic extension is capable of delivering the requirement. However, all steiner tree heuristics depends on the knowledge of membership. Also, none addresses dynamic network conditions and heterogeneous QoS.

For MBone, dynamic membership is inherent in its design. Receivers can easily tune into light-weight multicast sessions. MBone also handles some degree of dynamic network conditions as it uses IP tunnelling between islands of MRouters. Packets route around troubled or congested regions between two MRouters, but the islands of MRouters have fixed topology. MBone also uses DVMRP for internal routing. It does not scale well for sparse membership as in multimedia stream multicast. MBone

	Dyanmic	Scalable	Light-weight	Heterogeneous
Steiner Tree Algorithms	YES	YES	NO	NO
MBone	YES	NO	YES	NO
PIM-SM	YES	NO	NO	NO
QoSMIC	YES	NO	NO	YES

Table 5.1: Existing multicast mechanisms against multimedia multicasting requirements

provides no solution to the heterogeneous QoS problem.

PIM-SM also fails to meet all requirements. PIM-SM also uses IP tunnels between PIM capable routers. In addition, PIM-SM does not impose a fix topology for PIM capable routers. PIM-SM can deal with dynamic network conditions much better. PIM-SM also supports dynamic membership. PIM-SM is receiver driven but not fully distributed. It is locally centralized at the rendezvous points. The RPs are the local points of failure; however, failures do not propagate to other part of the tree. PIM-SM also fails to address heterogeneous QoS.

QoSMIC concentrates on heterogeneous QoS requirements. QoSMIC also solves dynamic membership and dynamic network conditions. QoSMIC does not solve the scalability issue. QoSMIC requires a central management node for tree search. For a large multicast group, the manager node will become a performance bottleneck as the manager node keeps all the details for group membership. In addition, the QoS requirement specified in QoSMIC does not handle multi-rate stream distribution.

No approach discussed meets all requirements for multimedia stream multicast applications; therefore, no approach is ideal for multimedia stream applications. Table 5.1 shows the four groups of multicasting mechanisms against the four requirements.

Chapter 6

Active QoS aware multimedia stream multicast service

This chapter proposes Active QoS Multimedia stream Multicast service (A-QoS-MM) that combines group membership and data dissemination subservices and includes a QoS aware semantic. The target application of A-QoS-MM is multimedia stream multicast such as web TV or web radio. A-QoS-MM meets all requirements identified in Chapter 5. A description of A-QoS-MM is presented in 6.1. Discussion and analysis of A-QoS-MM follow in Section 6.2.

6.1 A description of A-QoS-MM

Before discussing the details of A-QoS-MM, we must discuss some basic requirements of A-QoS-MM.

The mechanism uses active network technology. More specifically, the multicast service requires only application level processing. The active service model is a sufficient requirement for the multicast service. A-QoS-MM is also independent of active

network approach. A-QoS-MM requires only that the architecture is capable of intercepting incoming active packets, and, when requested, creating agents inside active nodes.

A-QoS-MM assumes that there is an active router inside the multimedia server's subnet. We denote this active router the source node. The source node subscribes to the multimedia service. It is a delegate of the multimedia server for A-QoS-MM.

To support heterogeneous QoS, A-QoS-MM requires QoS be characterized and quantized in an ordered manner. For example, a QoS level n demands a better QoS than QoS level $n-1$. Data rates are quantized and they can map into QoS levels easily. Reliability can be characterized by the threshold of percentage packet lost. A-QoS-MM further assumes that details of the different levels of QoS and the address of the source are available to all potential clients.

A-QoS-MM is receiver-driven. For the basic join operation, A-QoS-MM behaves similarly to the basic active data dissemination in Section 3.3. When a client is interested in joining a multicast session, it sends an active *subscribe* packet towards the source using the existing unicast service. The *subscribe* packet includes program information to create active agents in active nodes.

A-QoS-MM assumes no knowledge of network conditions. Because no delay information is available, we approximate delay constraint with scope constraint. Reference [19] argues that less number of hops *usually* has shorter delay (exception with satellite links). Each *subscribe* packet includes a user specified time-to-live (TTL) value in the underlying unicast header. The *subscribe* packet travels only as far as the TTL limits.

A-QoS-MM *subscribe* packets should be compatible with existing networks. The *subscribe* packets should transparently route through non-active nodes. A-QoS-MM *subscribe* packets should be embedded within underlying packets. Reference [30] proposes the Active IP option as an active extension to traditional IP. When *sub-*

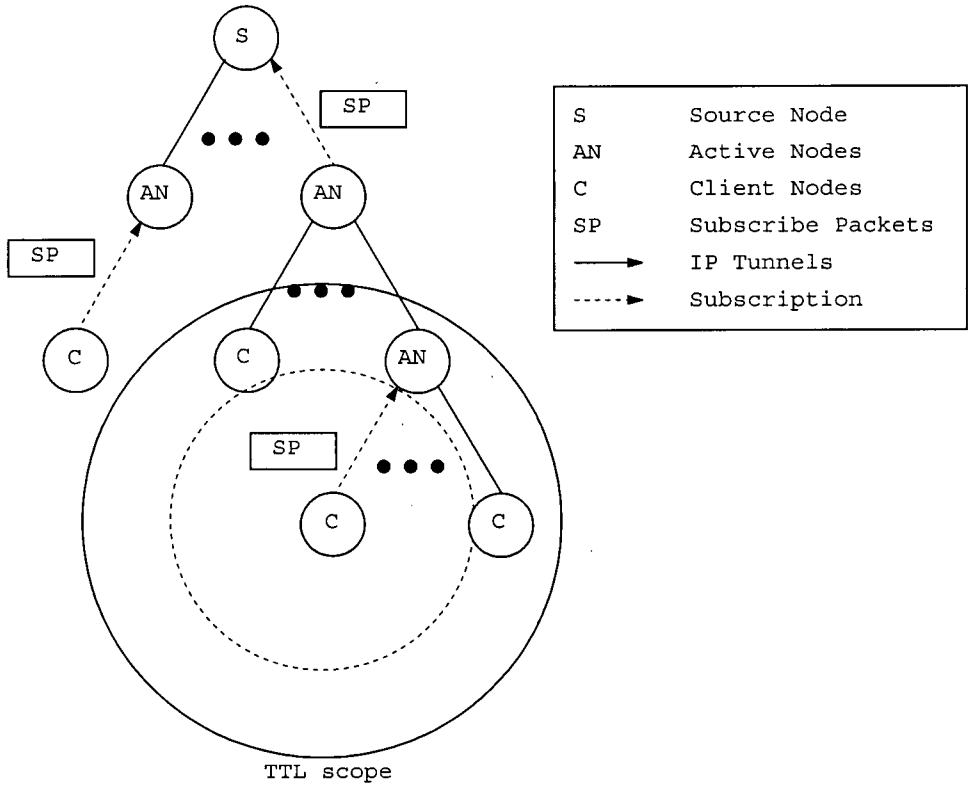


Figure 6.1: Basic joining operation of A-QoS-MM

scribe packets reach traditional non-active network nodes, the nodes can forward the *subscribe* like regular non-active packets. In essence, A-QoS-MM tunnels through traditional networks.

When a *subscribe* packet reaches an active node, the active node should recognize and pick up the active *subscribe* packet for further processing. If the active node has not been visited by *subscribe* packets from the same session, the active node creates an active agent within the execution environment in the active node. The active agent then registers the sender address of the *subscribe* packet as one of its children.

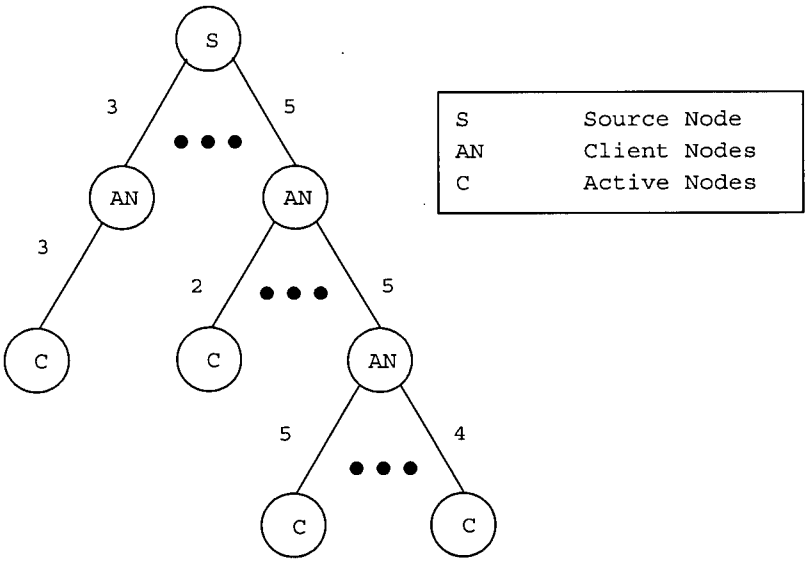


Figure 6.2: A-QoS-MM tree with heterogeneous QoS levels

After registering the child, the agent replaces the sender address with its own address and continues to forward the *subscribe* packet upstream.

This join operation is recursive. Active nodes intercept *subscribe* packets, create branches, and forward the *subscribe* packets to the source node. The *subscribe* packets can be intercepted by active nodes further up the path. When a *subscribe* packet reaches a visited active node, the active node does not need to create a duplicate agent for the same multicast session (common source node). Instead, the existing agent registers the sender as one of its children. Alternatively speaking, branches of the same multicast session merge. A-QoS-MM builds a reverse shortest path source-based tree. This A-QoS-MM extends basic active data dissemination by tunnelling *subscribe* packets through non-active elements. Figure 6.1 gives a snapshot of the basic joining operation in action.

A-QoS-MM is soft-state. The parents (intermediate active agents) set registration expiry timers for their children. The children (clients and also intermediate active agents) periodically send *subscribe* packets to refresh registration. If a parent node receive the *subscribe* packets from a child before the associated timer expires, it keeps respective logical tunnel alive. Otherwise, the parent assumes the child has left the group. The parent eliminates the internal state associated with that particular child and stops forwarding future data packets down that direction.

A-QoS-MM also adds a QoS semantic on top of basic active data dissemination. A *subscribe* packet will contain the QoS level required by the issuing client. When the *subscribe* packet reaches an active router, the active router only forward the *subscribe* packet upstream when the QoS level in the *subscribe* packet is greater than the registered QoS level in the active agent. In other words, the active agent only subscribes upstream when its current subscription can no longer cater to the demand to the registering child. The operation is analogous to a tournament. Only the strongest in the same stage proceed to the next stage (upstream node). In A-QoS-MM, the strongest is the *subscribe* packet with the highest QoS requirement. If the QoS requirement throughout the multicast group is uniform, A-QoS-MM behaves exactly like basic data dissemination. Figure 6.2 display a tree with heterogeneous QoS.

At this point, a tree with branches of different QoS levels is constructed. When an active agent receives a data packet from upstream, the active agent duplicates the data packet and processes each duplicate packet according to the different QoS levels. The application level computation is specified by program included in the *subscribe* packet that creates the agent. The application level computation may include but not limited to selective forwarding/discard as described in Section 4.2, multimedia transcoding as described in Section 4.3, and active reliability services as describe in 3.3. Application level computation can also provide local cache and retransmission

for streams with tight deadlines. A-QoS-MM involves only the active node lying on the shortest path from the client to the source. We can view the active agents as general programmable multimedia gateways that are built on demand at strategic locations. After the packets have been processed, the agent uses best-effort unicast to forward the processed packets to the appropriate children.

6.2 Discussion and analysis

In this section, we first give a discussion A-QoS-MM in relation to the requirements identified in Section 5.1. Then, we present some qualitative analysis for A-QoS-MM.

6.2.1 Meeting the requirements

A-QoS-MM meets all requirements identified in Section 5.1. First, A-QoS-MM does not require knowledge of network condition. Like PIM-SM and QoSMIC, A-QoS-MM uses the underlying unicast service for packet delivery. The underlying unicast service takes care of routing details and changing network conditions. As the network changes, the unicast service adapts to the change. A-QoS-MM, residing on top of the unicast service, also adapts to changing network conditions.

Also, A-QoS-MM accommodates dynamic membership. In contrast to the steiner tree approaches in 3.1, the reverse shortest path forwarding algorithm used in basic active data dissemination and A-QoS-MM does not require knowledge of membership. Each client establishes a reverse path toward the source. The reverse path of a client does not depend on any other client. This reverse shortest path forwarding algorithm will yield the same tree regardless of the order of join operations. Thus, A-QoS-MM does not need a pre-determined join order of clients. Clients can dynamically join the group. Clients can also dynamically leave the group. The soft-state model automati-

cally prunes branches toward inactive clients. In addition, A-QoS-MM is much more flexible to dynamic network conditions and dynamic membership because the tree can restructure using the soft-state model. This meets the dynamic requirements.

For the light-weight requirements, logical tunnels and reverse shortest path forwarding algorithm solve the problem. The logical tunnels give abstraction over the network layer. The clients of A-QoS-MM need to know only the address of the source. All details are handled by the underlying unicast service. A-QoS-MM does not need to maintain the network state. A-QoS-MM uses the reverse shortest path forwarding algorithm to build the distribution tree. As described above, this algorithm requires no knowledge of membership. A-QoS-MM meets the light-weight requirements.

For the scalability requirement, A-QoS-MM is receiver driven. A-QoS-MM requires no central processing node. The Membership State is distributed across the tree. Processing is also distributed across the tree. A-QoS-MM conforms to the scalability requirement.

For heterogeneous QoS requirement, A-QoS-MM explicitly includes a QoS semantic in its tree construction. Active networks technology enables A-QoS-MM to accommodate heterogeneous QoS levels. Data rate is one of the QoS requirements. Figure 6.3 shows an example of a multi-rate tree. Every client is able to subscribe a desired data rate without additional communication burden.

A-QoS-MM is a much more generalized subscription model than multi-rate delivery. A-QoS-MM can accommodate any QoS requirement that can be quantized into multiple levels. Processing of the different QoS levels can pass on to the specific handling services, such as using AER/NCA for reliable service.

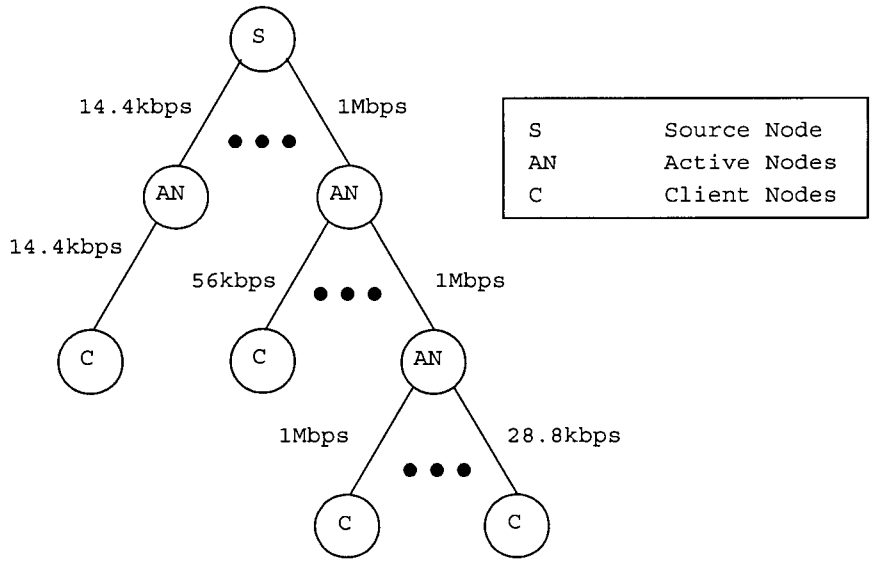


Figure 6.3: An example of a multi-rate dissemination tree

6.2.2 A qualitative analysis of A-QoS-MM

In addition to meeting the requirements in Section 5.1, A-QoS-MM must meet fundamental requirements such as correctness and robustness. Multicasting services are also prone to implosion problems. We also show how A-QoS-MM avoids this problem.

There are three correctness requirements for A-QoS-MM. We must ensure that A-QoS-MM builds a reverse shortest path tree. We must also ensure that tree construction terminates. We must also ensure that data is properly distributed to the clients. Because A-QoS-MM uses underlying unicast for delivery, the reverse shortest path requirement falls through to the underlying unicast layer. If the unicast routing routes packets through the shortest path, then unicast packets in the reverse direction are also travelling in their shortest paths. Forwarding in the shortest path is one of the fundamental design goals of unicast routing. The above condition is always

met. The *subscribe* packet stops when the *subscribe* packet reaches an active router or when the *subscribe* packet reaches the source. When a *subscribe* packet reaches an active router, the *subscribe* packet was travelling through its shortest path to the source. The *subscribe* packet either get discarded after registration if the active router is already in the tree, or continues on otherwise. In the case where the active router is in the tree, the path from the source to the active router always follows the reverse shortest path. In the case where the active router is not yet in the tree, the packet continues on going in its shortest path. The resulting paths from the source to every client always follow the tunnels established in the reverse shortest paths.

Join operation in A-QoS-MM always terminates. The underlying unicast routing is assumed to be free of loops. The result path followed in A-QoS-MM join operation is therefore free of loops. The destination of all *subscribe* packets is the source. The *subscribe* packets either stops at the source, at an in-tree active router, or at a non-in-tree active router. The operation continues only when the *subscribe* packet reaches a non-in-tree active router. In that case, the join operation goes on and the *subscribe* packet is sent toward the source again. The outgoing packet will stop again when it meets one of the three choices. This situation is recursive. Because the underlying unicast is free of loops, the *subscribe* packet has to reach an in-tree active router or the source at a point. Thus, the join operation eventually terminates in all cases.

To distribute data to all clients, we must ensure the data forwarded following the branches eventually reaches the clients. In a symmetric network where the cost and delay of a link in both directions are equivalent, the data packets from the source node to the clients will follow the loop free reverse shortest path. All clients will be able to receive the data packet. In an asymmetric network where the cost and delay of a link in the two directions differ, packets may take different routes between two hops. Before reaching the next hop, a data packet can possibly reach a downstream tree node or another tree node not belonging to the same reverse path. A loop may form in

this situation as the same packet may later forward back to the same node. Because A-QoS-MM uses IP tunnelling between parent and children nodes, the destination address of the data packets are the children's addresses. Each segment of the tunnel is loop free because it is a unicast path. When a data packet reaches an in-tree active router, the router processes the packet only if the router is the destination of the packet. When a data packet reaches another tree node in its path to its destination, the in-the-way tree node can simply forward the data packet without manipulation. The data packet will always travel in the loop free tunnels between the parent and its children. Because the tree does not have logical loops and segments of tunnels between parents and children contain no loops, data packet will flow downstream and will terminate at the client nodes. A-QoS-MM can be modified to take advantage of short-cuts using in-the-way nodes. The in-the-way nodes can process and forward the data to their children and forward the original packet to the destination. Extra state information is needed at the in-the-way nodes to memorize the packets that used the short cuts. When the same packets revisit in-the-way nodes when they are forwarded down the tree, the in-the-way nodes may simply discard the revisiting packets.

A-QoS-MM is robust. It recovers automatically and gracefully to local link failures. A-QoS-MM uses IP tunnelling and packets can route through troubled area without major service disruption. A-QoS-MM also automatically and gracefully recovers to local active node failures. If an active node without any children fails, there will be no service disruption. In the case that a participating active node fails, the children will get a minimal service disruption of the refreshment period. When the refreshment timer expires, the children will subscribe to different regions of the tree and continue to receive service.

Multicast services are prone to potential implosion problems because the multicast tree can accommodate an arbitrary number of members and all members can potentially send packets to a single source simultaneously. In A-QoS-MM, all *sub-*

scribe packets is directed to the source. If there are no active routers between the clients and the source, A-QoS-MM reduces to multiple unicasts and the source can suffer from subscription implosions. However, this is not a useful case of A-QoS-MM. For general cases of A-QoS-MM where active routers are abundant, no node in A-QoS-MM suffers from implosion. A-QoS-MM includes an explicit QoS semantic. Only one *subscribe* packet from all siblings travel upstream. Each node will only receive at most the number of children of *subscribe* packets.

6.3 Simulation results

To judge whether a multicasting data dissemination service such as A-QoS-MM is efficient, communication cost and control cost are particularly important. The communication cost and control cost determines the effective data rate of a multicast delivery service. Because A-QoS-MM involves application level processing, computational cost is also a major concern. Using simulation, we can assess the performance of A-QoS-MM in relation to these three costs. The design of the simulation can also shed some light on the design of possible A-QoS-MM implementations. We first describe the simulation system used. Then, we look at the results obtained in the three areas.

6.3.1 Simulation System

Here, we will show the architecture to the relevant parts in *ns-2*. PANAMA's active network package for *ns-2*, on which we extend into A-QoS-MM simulation, will also be shown. Then, we will give a description of our extension and the set up of the simulation runs.

6.3.1.1 The ns-2 architecture

We used *ns-2* for our multicast simulation. It is a part of the Virtual InterNetwork Testbed (VINT) joint project from USC/ISI, Xerox PARC, LBNL, and USB. *ns-2* is an object oriented discrete event simulation system tailored to network simulations. Each type of network elements, from physical layer links to network layer routers and packets to transport layer TCP connections, are defined as classes. Individual elements are the instantiations of their corresponding classes. These objects are configurable, but they retain the general properties and behaviours of the actual network elements. *ns-2* is able to work on many levels of abstraction. In this research, *ns-2* builds a model of wide area network.

Reference [15] is a comprehensive user documentation for *ns-2*. Reference [11] provides a detail hierarchical break down of *ns-2* classes, and the list of methods and fields for each class. The most relevant classes to A-QoS-MM are the packet class, the node class, the classifier class, and the agent class.

Packets in *ns-2* are regarded as events. The packet class defines the type of information to be passed to the each node. It contains different headers. Each header has its own set of fields.

A node represents its counter part in the real world. It is an abstraction of a collection of network entities for a single address. Agents, interfaces, and other network entities attach to their resident nodes.

When a packet reaches a node, the attached address classifier receives the packet event. A classifier is a dispatch for handling specific type of packets. Classifiers may attach on top of other classifiers. A node may have a hierarchy of classifiers attached to it. For example, an address classifier determines if the packet should be forwarded to the next hop. A port classifier dispatches a packet destined for the local node to the specific handling agent. According to the information provided, the packets could

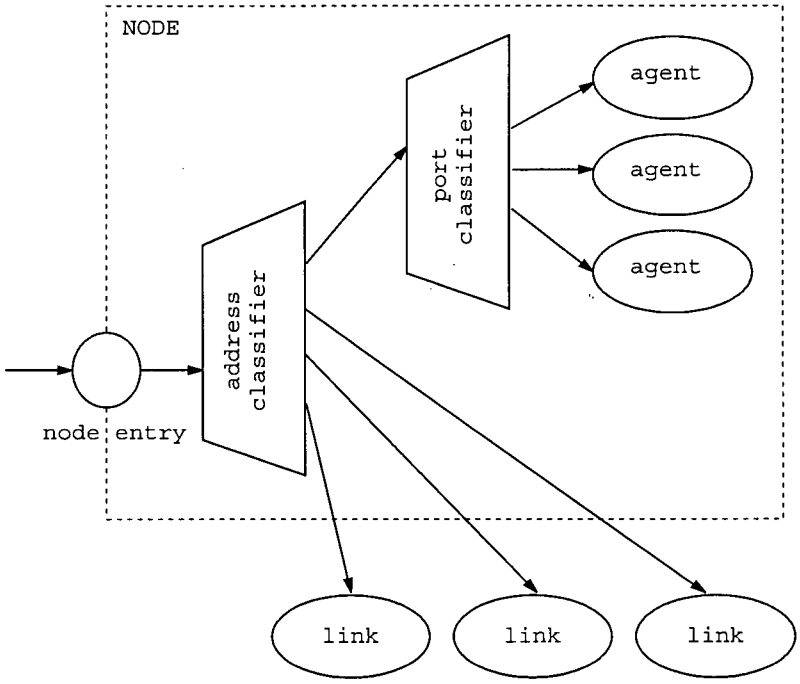


Figure 6.4: Internal structure of a ns node

then be forwarded to the next hop or an attached agent.

The agent class is an abstraction of higher layer entities and agents are packet handling objects. Each type of agents handles a specific type of packets. An agent can be a source or receiver of a packet or stream. Some examples are a TCP agent, a RTP agent, or a sink. Figure 6.4, a simplified reproduction from Reference [15], shows the internal structure of a *ns* node and the relationship between the objects.

6.3.1.2 PANAMA active network simulation architecture

To simulate active networking with *ns-2*, *ns-2* must have the ability to create active agents. The Protocols for Active Networking with Adaptive Multicast Applications

(PANAMA) project from TASC and UMass provides an active network package for *ns-2* that enables users to create pre-programmed active agents inside network nodes in simulation driver scripts. Users can call active agent methods in the simulation script to enable or disable agents.

The PANAMA active package adds several new classes to *ns-2* and re-calibrates the simulation core for the new classes. First, PANAMA defines a new type of packets, the active packet. The active packets can be further classified as active data packets, active initialize packets, active control packets, and other types of active packets. PANAMA derived ANAgent (Active Network Agent) class from the agent class. The ANActive class defines the skeleton of agents that are capable of handling the new active packet header. Based on the ANAgent class, PANAMA further derived the ANSndr class and the ANActive class. The ANSndr class defines agents that produce and send active data packets. The ANActive class defines agents that handle received active packets. In this package, the active agents simply change the payload size of the active packets. PANAMA also defines the ActiveNodeClassifier class as a derived class of classifier that dispatches active packets to active agents. PANAMA also modified the part of the *ns-2* core that is makes the new active packets and classes compatible with the other parts of the core.

To enable active agents in the PANAMA package, users must invoke the initialize method of the agents in the simulation script. The agents then send themselves each an active initialize packet. The active initialize packets remain in their respective local node, trace the data path from the lowest node entry to the ActiveNodeClassifier, and, along the way, inform the internal node components about the active agents.

The capability of enabling agents only in the script. A-QoS-MM requires active agents to be enabled when an active packet is received. Also, only one ANActive agent can attach to an ActiveNodeClassifier. This design decision is not suited for multiple QoS simulation. To simulate A-QoS-MM, we have to extend both *ns-2* and

the PANAMA package.

6.3.1.3 Architecture of A-QoS-MM simulation

The active agents are pre-programmed to perform the A-QoS-MM specific tasks. This is different from actual active network approaches. In actual active networks, active packets should contain all necessary information to create active agents. In our simulated approach, agents are pre-programmed. They are enabled when active packets are received. Although there is architectural difference between the actual active network and the simulated network, there are no functional differences in this simulation because A-QoS-MM is the only existing active service in the simulation. An active packet in the simulation is able to specify the only type of active agents to be created.

The very first thing we extended on top of the PANAMA package is the subtype of active packets. A-QoS-MM subscribes upstream by sending *subscribe* packets. We extended the PANAMA package with this new subtype of active packets.

In A-QoS-MM, a data packet travelling downstream is passed to the active agent for processing. The agent then duplicates and processes the packet for different QoS and forwards the processed packets to the appropriate downstream node. The processing of the different QoS is done among different processing entities with each dedicated to one QoS level.

If the above design is to be implemented in *ns-2*, it would look like Figure 6.5. The ActiveNodeClassifier would pass the packet up to an intermediate active agent. Then the agent would dispatch the packet to the number of processing entities. The architecture from Figure 6.5 would be functionally correct, but the intermediate active agents would behave very much like ActiveNodeClassifiers and dispatch packets. The active agents and the ActiveNodeClassifiers are functionally redundant.

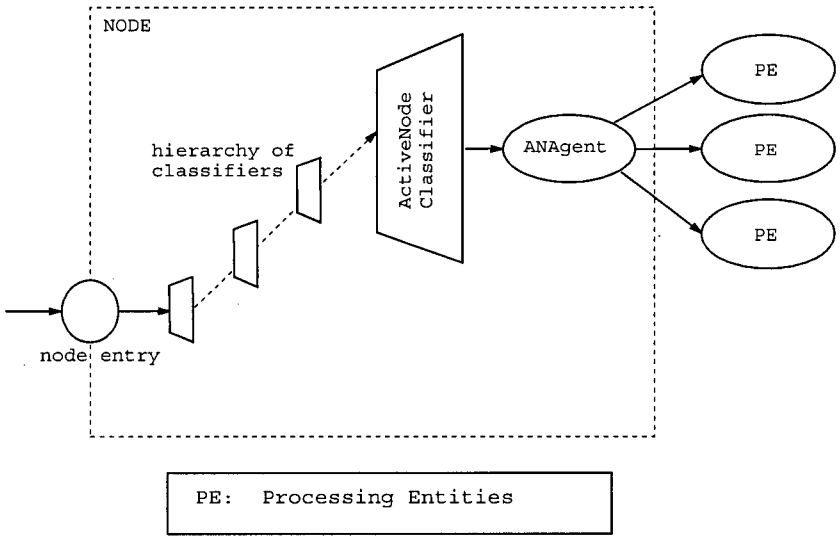


Figure 6.5: A possible internal node architecture for A-QoS-MM

In fact, the processing entities handle active packets. They are more suited as active agents in the *ns-2* architecture. In our own implementation of the A-QoS-MM simulation, we merged functionality of active agents from the previous design with the functionality of ActiveNodeClassifiers. We bring down the processing entities as active agents. The new version of node architecture would look like Figure 6.6. Instead of having to build a new interface to access processing entities, we can use the available interface from *ns-2* to access active agents.

A new problem arises from this new node architecture. The PANAMA package was originally designed for a maximum of one active agent per node. The agent enabling process is very straightforward. The agent is enabled when a method is invoked. In this new architecture, agents have to be enabled when the node receives a *subscribe* packet.

They are first initialized much the same way as the original PANAMA active

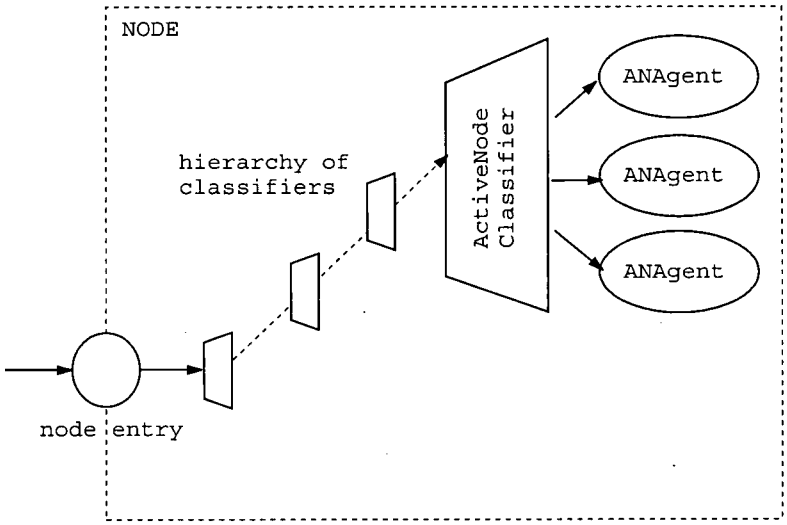


Figure 6.6: Our node architecture for A-QoS-MM

agents. Data path is set u along the hierarchy of classifiers. Then, the agents are disabled. Disabling an agent is done with invoking the agent's disable method and sending itself a disable packet. The classifiers on the way would switch off the path toward the active agents and treat future active packets (except enable packets) as ordinary packets. To re-enable the active agents, the paths to the active agents are switched on when enabled packets are received. The disabled active agents themselves need not send enable packets. Other active agents or clients can send the enable packets. On-demand agent enabling is achieved. This models the on-demand active agent creation in A-QoS-MM.

Membership management involves all active agents within an active node. To correctly manage their children, active agents must interact with each other. In addition, no active agents can deal with only their own children. For example, child A originally subscribes QoS level n. When child A changes its requirement to m, the

active agent dedicated to QoS level n should give up the parenthood to child A to active agent dedicated to QoS level m. This complex handoff mechanism cannot be efficiently implemented within each agent. We use a mediator.

We define the SMFMembership class. The SMFMembership object is the access point of membership information for all active agents within an active node. Each SMFMembership object resides within one ActiveNodeClassifier object. At initial agent set up, active agents remember the pointer to the SMFMembership object and register themselves in the SMFMembership object. Active agents can later access the SMFMembership object and vice versa. SMFMembership has both a hash table and a number of priority queues.

The hash table keeps track of all QoS level subscribed by all children. Each priority queue dedicate to each active agent. The priority queues manage the working sets of children for the active agents with priority of earliest expiry time.

When a downstream node subscribes QoS level n at an active node, hash table inserts the child's address and a list[1] of QoS level subscribed in its entry. In the case that the member node has already subscribed QoS level m, the hash table simply updates the corresponding entry and inserts n in the QoS list. Looking at the list of subscribed QoS level, the SMFMembership object is able to find out all and the highest level of subscribed QoS for the child node. SMFMembership is then able to use this piece of information to handle the complex handoff. SMFMembership is able to remove the member node from the lower QoS level priority queues, and insert the member node into the highest QoS level priority queue. If the member is only subscribed to one QoS level, SMFMembership simply insert the member node into the corresponding priority queue. If the current subscription is only a refreshment, SMFMembership rearranges the member node in the priority queue.

[1]In the actually simulation, we use an array of flags with size of the total number of QoS levels. The flag at the subscribed QoS level position is set.

Each priority queue contains the children nodes for the active agent dedicated to the QoS level. When an active agent finished processing an active data packet, it looks at the corresponding priority queue, duplicates the processed packet, and sends the duplicated packets to all of its children.

Priority queues provide simple mechanism for removing expired children. When the active agent's timer expires, the active agent removes the first child node of its priority queue, and re-adjusts the timer to the next expiry time. This expiry mechanism is very efficient. Also, the active agent only needs one timer for all children.

Priority queues also come in handy when the active node subscribes upstream. An active node will always want to subscribe the highest QoS level of its children. With priority queues, the highest QoS level can be easily determined by finding the non-empty priority queue with the highest dedicated QoS level. A central membership management object, SMFMembership, is useful for the simulation. Perhaps a similar membership management entity may be equally useful in an actual active network environment.

membership management. The original design does not provide this

6.3.1.4 Simulation setup

We generated numerous graph samples for the simulation. We used Georgia Tech Internetwork Topology Models (GT-ITM) for topology generation. The topology of choice is transit-stub graphs from Reference [7]. Transit-stub graphs are three level hierarchical graphs that characterize typical internetworking topologies. We used graphs of size 50, 100, 200, 300, 400, and 500. Ten graphs for each size are generated.

We run tests that give the level of control, communication, and computation cost for a given graph. We test with hybrid networks. Active and non-active elements co-exist in the generated networks. We test with active/traditional hybrid networks.

We vary the number of clients and density of active nodes. The number of clients represents the level of population inside a network. The density of active nodes is an indirect representation of the number of active nodes participating in a multicast session. The higher the density, the higher the likelihood that a non-client node is an active node. All tests run all of the generated graphs but the results of only the most illustrative ones are shown.

A-QoS-MM uses unicast tunnels. We run Dijkstra Shortest Path First (SPF) algorithm with hop metrics as the underlying unicast routing algorithm. The simulation tests only measure the costs at a snap shot of the mechanism. We simulate only one period of the soft-state refreshment cycle. No network changes occur. Dijkstra SPF is sufficient for the simulations. Because the simulation test deal with logical tree hops and uses unicast only as a delivery service, any unicast routing algorithms should give similar results.

6.3.2 Control cost

Control cost is one of the three major concerns in A-QoS-MM. As a matter of fact, control cost is important to any large-scale communication for scalability reasons. Control cost is particularly important for soft-state systems because it affects the effective throughput. A-QoS-MM is a soft-state, large-scale multicasting mechanism. A-QoS-MM should minimize its control cost.

In A-QoS-MM, leave operation can be handled with little control effort with timeouts and implicit pruning. The only operation that requires control effort is the join operation. When a client wish to join, it sends a *subscribe* packet. When the *subscribe* packet reaches an active router, it may create another *subscribe* packet. The process goes on until the *subscribe* packet reaches an active node capable of delivering the require QoS. Every join operation will create a number of *subscribe* packets. The

control effort of A-QoS-MM is the additional control processing needed to add the client into the group. The control effort is directly proportional to the number of *subscribe* packets created.

In our test runs, we used the number of *subscribe* packets created as the control cost. We kept a counter of the number of *subscribe* packets in the simulation script. The counter is initialized to zero. When a client or an active node subscribes, it invokes the subscription method which then increments the counter. Because *ns-2* is a single-threaded event-driven simulation package, the counter does not experience critical section synchronization problems.

From Theorem A.1, the probability that the immediate upstream active node generates a new *subscribe* packet is $\frac{1}{2}$. From Corollary A.2, it follows that a single join operation is likely to generate less than two *subscribe* packets. The total number of *subscribe* packets generated for n clients is likely to be less than $2n$. Figure 6.7 demonstrates this relationship.

Figure 6.7 plots the number of *subscribe* packets generated for a 200 nodes network with four QoS levels. We vary the number of clients. We also vary the number of active nodes as a percentage of the remaining nodes. The percentage of active nodes is a measure of the density of active resource available.

For a fixed number of clients, the number of *subscribe* packets increases in the same direction as the percentage of active nodes. This result conforms to the theoretical model in Corollary A.2. A lower percentage of active nodes is likely to produce a shallow tree. The small number of stages generates less *subscribe* packets. For a fixed percentage of active nodes, the number of *subscribe* packets increases in the same direction of the number of clients. This too conforms to Corollary A.2. The expected number of *subscribe* packets is directly proportional to the number of clients. The results of all simulation runs follow the same trend.

A sub-linear correlation between the number of clients and the number of *sub-*

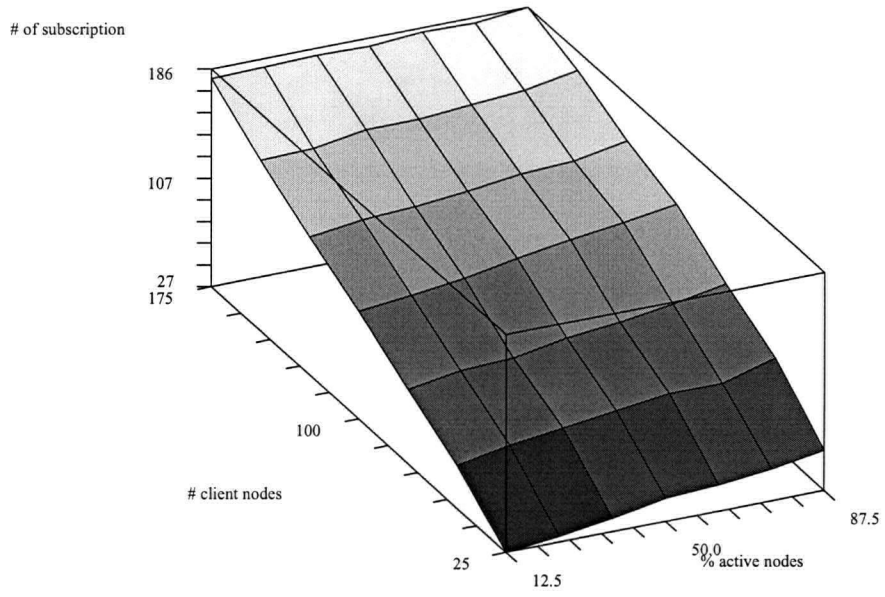


Figure 6.7: Total number of subscribe packets

scribe packet as in Corollary A.2 is acceptable in terms of scalability. Although A-QoS-MM is a soft-state mechanism, the number of *subscribe* packets does not affect the effective throughput. All control packets travel upstream, and all data packets travel downstream. The control packets do not take hold of any downstream bandwidth.

The actual control bandwidth required is also determined by the particular active architecture that is used. For an integrated active architecture, the *subscribe* packets include the entire setup and functional description of the active agents. The size of the included codes and therefore the size of the packets may become quite big depending

on the application. For a discrete active architecture, the *subscribe* packets have to include only some identification or abstracted description of the active agents. If an active node finds no associated code for the active agent within the node's registry, the node can then request the code from downstream nodes. The offline program transfer incurs a longer initial delay. The integrated approach requires higher control bandwidth, but the discrete approach requires longer initial setup delay.

6.3.3 Communication cost

Communication cost is another major concern for A-QoS-MM. For any multicasting routing service, large volume of data is going to stream through the constructed tree. For this reason, multicast routing services should minimizes the cost of the constructed tree.

From the light-weight requirement, A-QoS-MM has no prior knowledge of membership and network condition. A-QoS-MM is restricted to build the source-based tree from the client direction using lower layer unicasting service. A-QoS-MM uses reverse shortest path forwarding. The tree cost of A-QoS-MM is comparable to other reverse shortest path services such as CBT and PIM. Some multicast routing services such as QoSMIC can outperform reverse shortest path forwarding; however, such services require source/tree initiated construction or centralized processing which contradicts other requirements from Section 5.1.

For a graph with unit cost edges, the total number of tree hops reduces to total tree cost. The number of tree hops is easiest to be determined when all clients subscribe to the same QoS level. The sum of all paths from client to the tree becomes the number of tree hops.

To measure the path to the tree, we have to modify the simulation core. When a *subscribe* packet reaches a non-visited agent, the non-visited agent simply forwards

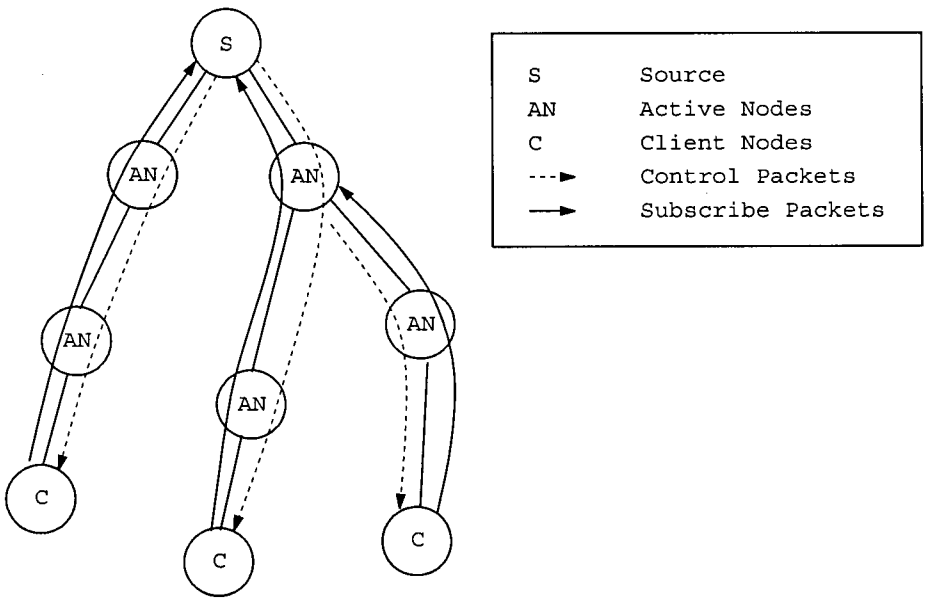


Figure 6.8: Path measurement using subscription/control echo

the subscription packet upstream. We modify the behaviour of active agents when the *subscribe* packet reaches a visited agent. The visited agent sends a *control* packet echo back to the *subscribe* sender. The source also always echoes with control packets. This operation is shown in Figure 6.8. We fix the delay of all links to half a second. To each client, the delay from sending a *subscribe* packet to receiving a *control* packet is the number of hops to the tree. We find the total number of tree hops by summing all path hops.

Figure 6.9 shows the tree cost of a 200 nodes network with various number of clients and active nodes. When the number of clients increases, more nodes and more hops are added to the tree. The tree hops increases. When the percentage of active nodes increases, some partially redundant network paths of logical tunnels

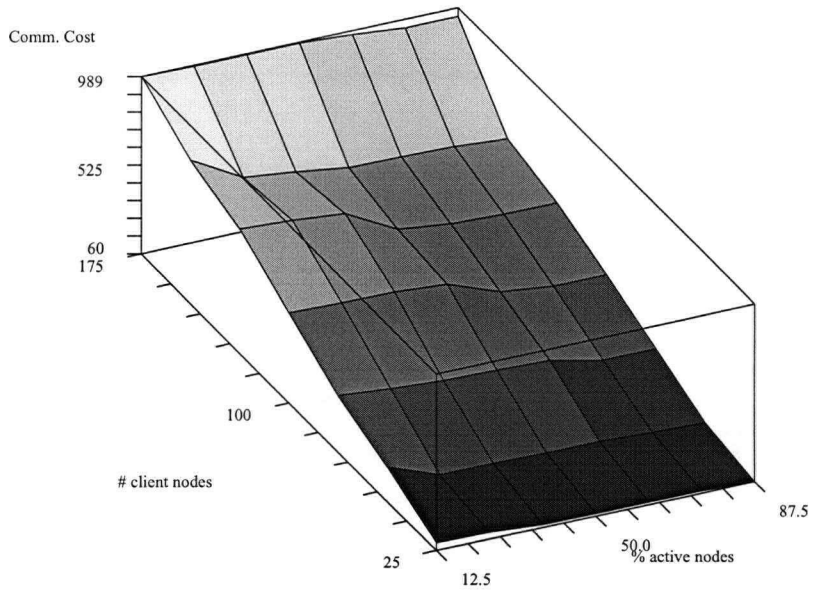


Figure 6.9: Total hops of tree

from children to parents are eliminated with additional intermediate nodes. Figure 6.10 shows the theoretical total hops of a tree with depth of 3 and degree of 5 with various numbers of uniformly distributed clients and active nodes. The total number of nodes is different between the simulation runs and theoretical calculations, but all follow the same pattern.

Tree cost does not entirely represent the communication cost of A-QoS-MM because tree cost ignores the ability of A-QoS-MM to deliver multiple data rate. A-QoS-MM is inherently more network efficient than the other multicast routing services as A-QoS-MM includes data rate as one of its explicit QoS semantics. Provide that

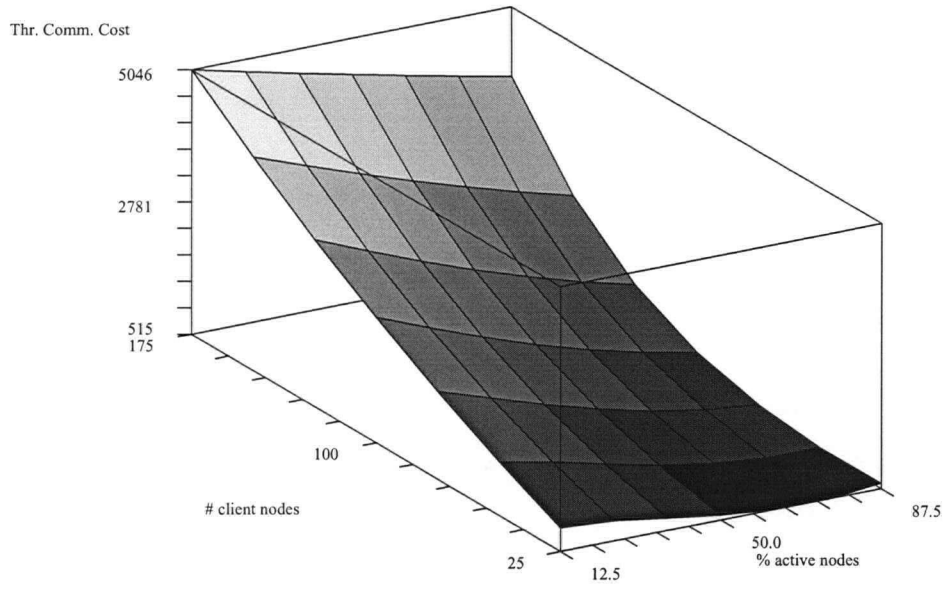


Figure 6.10: Theoretical total hops of tree

clients do not request data rates more than the bottleneck bandwidth in their respective unicast paths to the source, all clients receive the full requested data rates. A-QoS-MM branches the reverse shortest path forwarding tree at the latest possible points to reduce additional bandwidth consumption.

6.3.4 Computational cost

A-QoS-MM uses the computational power from active networking to provide the QoS aware multicast service. Computation in active routers requires computational resources. Computational resources are often limited. Computation is not free. A-

QoS-MM has computational cost.

Computation in each active router mostly deals with the processing for each of the different QoS levels requested immediate downstream. The more different levels of QoS are requested in the immediate downstream, the more computation is required. The total computational cost is directly proportional to the sum of number of different QoS levels of all participating active nodes.

To measure the computational cost, we use the original simulation core. The number of active agents enabled within a node equals the number of different QoS level subscribed. Summing the number of enabled active agents on all active nodes gives the total number of different QoS level subscribed. In the simulation script, we declare two global variables. One for the number of enabled nodes, and one for the total number of enabled agents. At the end of each simulation run, we collect both the number of active nodes involved and the number of agents created. The simulation script goes through each active node and gather the information. A flag is set if an active node is enabled. For each enabled node, the script invokes an update_children method in one of the children to access and retrieve the number of different QoS level in SMFMembership.

Figure 6.11 display the total cost for a 200 node network with four QoS levels. We vary the numbers of clients and active nodes to obtain a surface plot in Figure 6.11. We also obtained Figure 6.12 for the cost taken on average of ten simulation runs. Figure 6.13 and Figure 6.14 show the number of active nodes involved in the mechanism for one randomized network graph, and the average number of active nodes involved for ten simulation runs, respectively.

In both the result of a randomized graph, Figure 6.11, and the average result from ten graphs, Figure 6.12, the number of different QoS levels generally increases when the percentage of active nodes increases. At higher density of active nodes, *subscribe* packets are more likely to encounter active nodes, as shown in Figure 6.13 and Figure

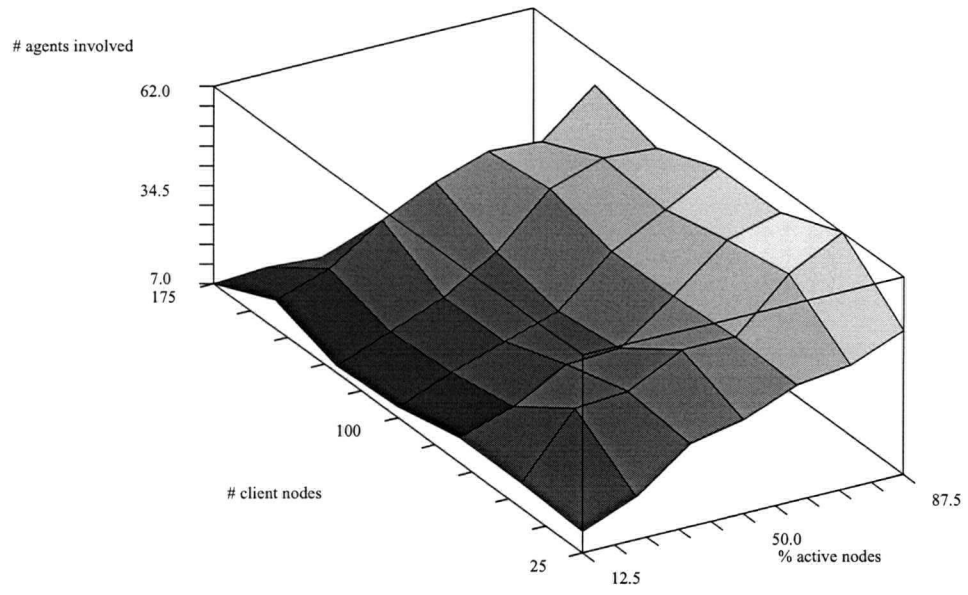


Figure 6.11: Total computational cost

6.14, and therefore creating additional computational entities (active agents) within the network. Each additional active agent contributes to the total different QoS levels.

Figure 6.11 and Figure 6.13 show less of a pattern with respect to the different number of clients. Figure 6.12 and Figure 6.14 can tell us more about the computational performance. At low numbers of clients, the number of active nodes involved and the number of agents created are small. Few clients are subscribing, and the clients subscribe with smaller number of differing QoS levels. As a result, less active nodes get involved and less agents are created. As the number of clients increases toward the mid-range, the number of active nodes involved and the number of agents

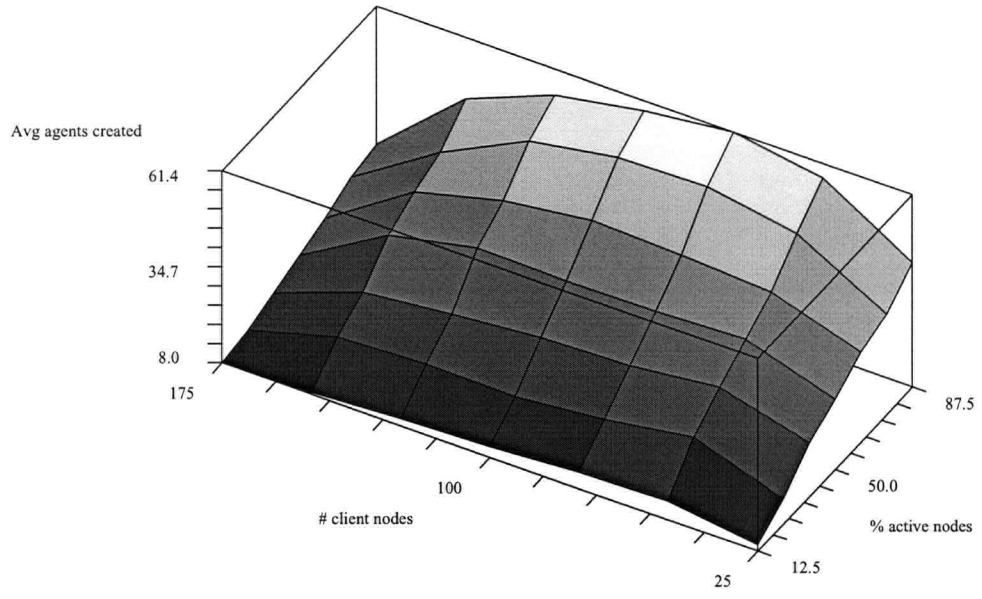


Figure 6.12: Average computational cost across ten runs

created increases. More clients are subscribing, therefore there is a bigger number of different QoS level subscriptions.

As the number of clients goes from mid-range to high-range, the number of agents created decreases. Higher number of clients leaves less room for active nodes. With a fixed number of QoS levels, the clients are going to have a large number of subscription subscribing to a limited number of different QoS levels. As a result, fewer active nodes create fewer agents that handles a large number of subscriptions.

Figure 6.15 shows the number of agents created per active node involved. This is equivalent to the computational cost per active node involved. The computational

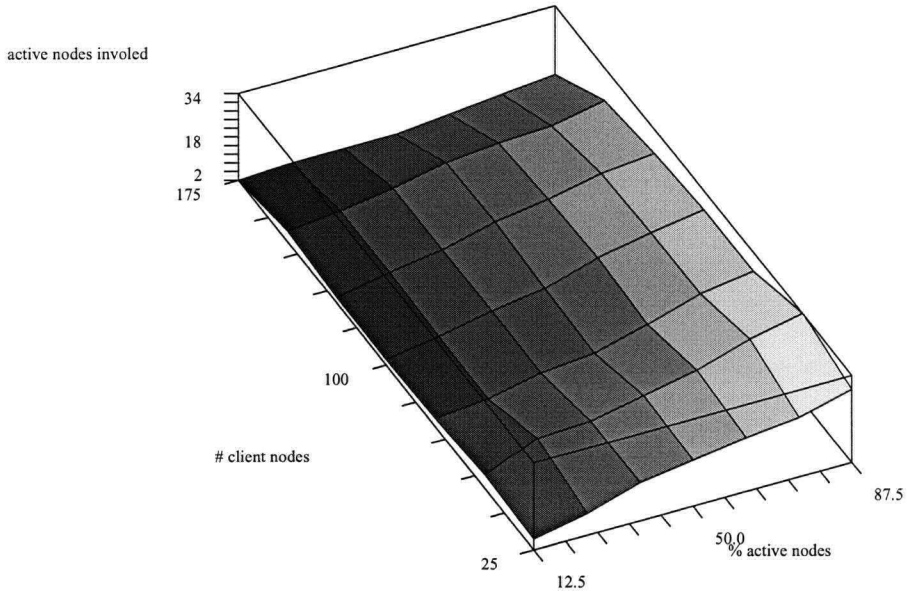


Figure 6.13: Active nodes involved in the mechanism

cost per node generally increases as the number of clients increases. Each active node is receiving more subscriptions and it has to create more agents to handle the larger number of QoS levels.

The computational cost per node generally decreases as the density of active nodes increases. At higher active node density, more active nodes are participating in the multicast session. Processing of different QoS levels are spread out. Each active agent only needs to handle a small number of QoS levels.

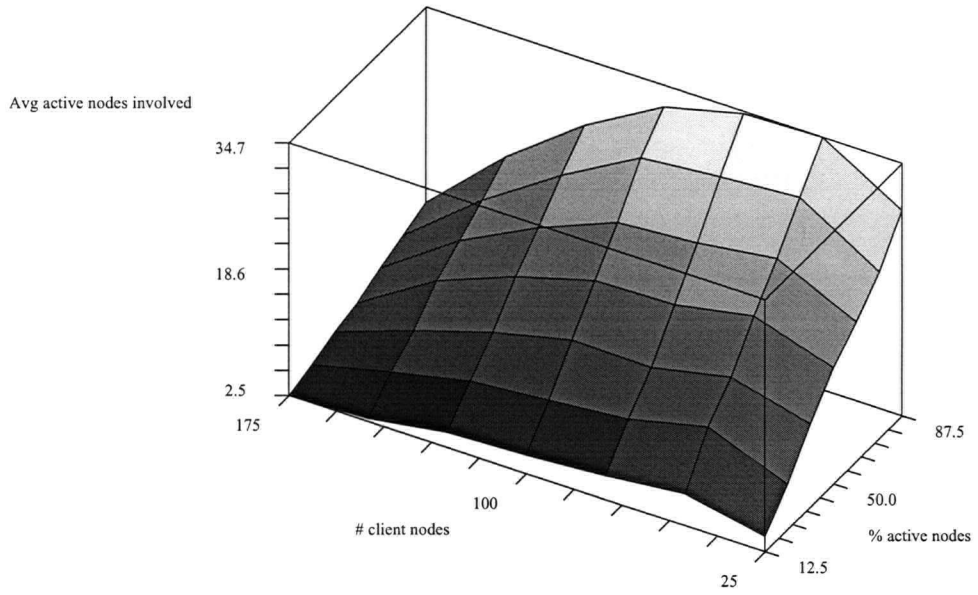


Figure 6.14: Computational cost per node

6.3.5 Discussion of results

Although the tests run at various number of clients, the results are most relevant at low number of clients. High number of clients resembles the conditions of a dense multicast session. Multicast in wide area networks are more likely be sparse where the session has a relatively low number of clients.

At low number of clients, all costs, including total computational cost, increase as the number of clients increases. The behaviour with respect to the density of active nodes is less uniform across the four costs. When the density of active nodes increases, the total computational cost and control cost increase while the other costs

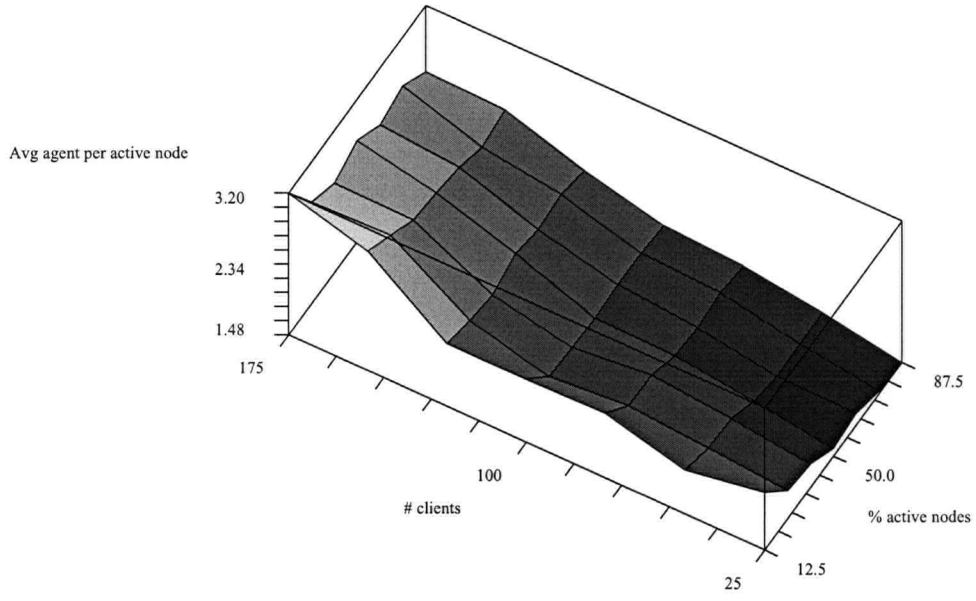


Figure 6.15: Average computational cost per node

decreases. As we place replace more traditional network elements with active nodes, we improve the communication. We also spread the computational load. However, performance improves at the cost of increasing total computation and control. At high density of active nodes, additional increase in active nodes yields little marginal improvement on the other costs. For example, an additional active node within a local area network will bring little improvement on communication. The additional active node will only induce extra control and computation.

6.4 An analytical comparison with QoSMIC

The mechanism most comparable with A-QoS-MM is QoSMIC. They are similar in aim and in scope. Both focus to solve the heterogeneous QoS multicast problem in the wide area domain. The application level computational cost is not applicable to QoSMIC. A-QoS-MM communication cost and control cost can be compared against QoSMIC.

In the local search flavour of QoSMIC, a join operation involves two phases. In the searching phase, the joiner floods its neighbours with time-to-live of t. From Reference [16], the flooding requires $O((\omega-1)^{t-1})$ message complexity where ω denotes the average degree of the network. In the bidding phase, at most N messages are generated for a tree of size N. The control message cost for QoSMIC per join operation per node becomes $O((\omega-1)^{t-1}+N)$.

For the multicast tree search version of QoSMIC, the joiner sends the request only to the manager router. The manager router chooses c suitable candidates as the joiner's upstream router, and sends each a BID-ORDER request. Each candidate then sends a BID to the joiner. A total of $2c$ messages are created for each join operation. If c is proportional to the size of the tree N, the message complexity becomes $O(N)$.

As shown in Corollary A.2, A-QoS-MM generates only 2 messages per join operation. This is $O(1)$ message complexity. For a tree size of N, the control message complexity becomes $O(N)$. For local and multicast tree QoSMIC, the control message complexity becomes $O((\omega-1)^{t-1}N+N^2)$ and $O(N^2)$ respectively.

In terms of communication cost, local search QoSMIC and A-QoS-MM should have similar performance as both utilize reverse path forwarding. Each joiner also utilizes the available unicast routing table and needs no local computation. This requires only $O(1)$ time complexity. For multicast tree search QoSMIC, the manager

router can improve communication with extra processing. Running steiner tree type algorithm typically requires at least $O(N^2)$ time complexity where N is the size of the tree.

The above analysis shows that A-QoS-MM outperforms QoSMIC in terms of control cost for each join operation. Multicast tree QoSMIC, on the other hand, has potential communication improvement over A-QoS-MM. This improvement would be most observable if network condition is extremely asymmetric, but it will provide only marginal benefit for regular network conditions. Multicast tree QoSMIC requires centralized computation, which violates the distributed requirement, and does so at the cost of time complexity.

Chapter 7

Conclusions

With the increase in the connectivity and user population of the Internet, a new set of applications and services is migrating to home network-enabled PCs. For full deployment of multimedia stream distribution however, the Internet is not ready. A few technical issues need to be resolved.

This thesis identifies the wide area multimedia multicast problem as the combination of four sub-problems. A single source multimedia multicast service should be able to adapt to dynamic membership and dynamic network conditions. The multimedia multicast service should also be able to simultaneously accommodate an arbitrary number of clients. The multimedia multicast service should require minimal control states. The multimedia multicast service should also be able to manage the drastically different QoS requirements within its group members.

We proposed the A-QoS-MM multimedia data dissemination and membership service that solves the four sub-problems. A-QoS-MM uses active network for the computational power within the network. A-QoS-MM is soft-state and A-QoS-MM can adapt to dynamic conditions. A-QoS-MM builds a reverse shortest path forwarding tree for scalability. Using reverse shortest path forwarding and logical tunnels,

A-QoS-MM is free from the network and membership details. With application level abstraction and reverse shortest path forwarding, A-QoS-MM is also able to achieve light-weight setup. An explicit QoS semantic makes A-QoS-MM a heterogeneous QoS aware data dissemination and membership service.

With A-QoS-MM satisfying the four basic requirements, applications can build agents within active nodes to provide addition functionality. A-QoS-MM can combine with a reliability service, an order delivery service, a packet repair service, and/or other services to deliver a full set of application specific functionality.

The simulation results show that total computation cost and control cost improve at the cost of the average computation cost and communication cost. The converse is also true. The performance is unacceptable at either end of the spectrum. An inherent goal of multicasting mechanisms is to reduce communication cost. In the case of high average computational cost, active nodes would perform very poorly with high computational load. For A-QoS-MM, the network should be set up in favour of the communication cost and average computational cost.

To improve the total computational cost, future effort is needed for flexible agent invocation. For A-QoS-MM, active nodes within a small region are functionally redundant. The upstream tree nodes within the small region will only process one QoS level because the lowest tree nodes process all the QoS levels and subscribes only the highest QoS levels. A-QoS-MM would benefit from not creating the active agents in the upstream active nodes and bypassing the upstream active nodes.

Conversely, A-QoS-MM may benefit in terms of communication cost and average computational cost if long tunnels detours to sub-optimal shorter tunnels. Activating and setting up active nodes near long, optimal tunnels is functionally comparable with additional active nodes in the optimal path.

A-QoS-MM may benefit in all aspects with flexible agent invocation. The challenge is to decide whether to suppress or to reinforce agent invocation in different

situations. Local active node population and the application may influence the decision. Flexible agent invocation may span the field of distributed agent coordination and distributed agent communication.

One other area of improvement is in the underlying active architecture. More code efficient active API may reduce the size of the subscribe packet thus reducing the control cost. Another area of improvement is a tree restructure algorithm. Reverse shortest path used in A-QoS-MM does not provide the best communication performance in asymmetric networks. A-QoS-MM should be able to construct or restructure the tree for performance in the downstream direction. A further goal of A-QoS-MM is to provide flexible extension to the rigid MBone. A-QoS-MM can provide a short-cut between distant MBone regions. It can also provide multicast service to regions without MBone. Future work in A-QoS-MM should address the integration details and the other above issues.

Bibliography

[1] Ehud Aharoni and Reuven Cohen. Restricted dynamic Steiner trees for scalable multicast in datagram networks. *IEEE/ACM Transactions on Networking*, 6(3):286–297, June 1998.

[2] E. Amir, S. McCanne, and R. Katz. An active service framework and its application to real-time multimedia transcoding. In *Conference on applications, technologies, architectures, and protocols for computer communication (SIGCOMM '99)*, pages 178–189, Cambridge, August 31st - September 4th 1998.

[3] E. Amir, S. McCanne, and H. Zhang. An application level video gateway. In *The Third ACM International Multimedia Conference and Exhibition (MULTIMEDIA '95)*, pages 255–266, New York, November 1996. ACM Press.

[4] A. Ballardie, P. Francis, and J. Crowcroft. Core based trees. In *Conference on applications, technologies, architectures, and protocols for computer communication (SIGCOMM '93)*, San Francisco, September 1993.

[5] A. Banchs, W. Effelsberg, C. Tschudin, and V. Turau. Multicasting multimedia streams with active networks. In IEEE Computer Society. Technical Committee on Computer Communications, editor, *LCN'98: proceedings: 23rd Annual Conference on Local Computer Networks*, volume 23, pages 150–161, 1109 Spring

Street, Suite 300, Silver Spring, MD 20910, USA, October 11–14 1998. IEEE Computer Society Press.

[6] Fred Bauer and Anujan Varma. ARIES: A Rearrangeable Inexpensive Edge-based on-line Steiner Algorithm. Technical Report UCSC-CRL-95-36, University of California, Santa Cruz, Jack Baskin School of Engineering, July 1995.

[7] K. Calvert, M. Doar, and E. Zegura. Modeling internet topology. *IEEE Communications Magazine*, June 1997.

[8] A. Campbell, M. Kounavis, J. Vicente, D. Villela, and K. Mikiand H. De Meer. A survey of programmable networks. *ACM SIGCOMM Computer Communication Review*, 29(2):7–24, April 1999.

[9] S. Casner. Are you on the MBone? *ACM SIGCOMM Computer Communications Review*, 1(2), 1994.

[10] D. D. Clark and D. L. Tennenhouse. Architectural considerations for a new generation of protocols. *Sigcom 90*, pages 200–208, 1990.

[11] A. Clerget. http://www-sop.inria.fr/rodeo/personnel/antoine.clerget/ns/ns/ns-2.1b5/hier.html.

[12] Yogen K. Dalal and Robert M. Metcalfe. Reverse path forwarding of broadcast packets. *Communications of the ACM*, 21(12):1040–1048, December 1978.

[13] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C.-G. Liu, and L. Wei. An architecture for wide-area multicast routing. In *Proceedings, 1994 SIGCOMM Conference*, pages 126–135, London, UK, August 31st - September 2nd 1994.

[14] S. E. Deering. RFC 1112: Host extensions for IP multicasting, August 1989. Obsoletes RFC0988, RFC1054. Updated by RFC2236. Status: STANDARD.

[15] K. Fall and K. Varadhan. *ns Notes and Documentation.* 1999.

[16] M. Faloustsos, A. Banerjea, and R. Pankaj. QoSMIC: Quality of Service sensitive Multicast Internet protoCol. In *Conference on applications, technologies, architectures, and protocols for computer communication (SIGCOMM '99)*, pages 144–153, Cambridge, August 31st - September 4th 1998.

[17] M. Imase and B. Waxman. Dynamic Steiner tree problem. *SIAM Journal on Discrete Mathematics*, 4(3):369–384, August 1991.

[18] V. Jacobson, S. McCanne, and S. Floyd. Lightweight sessions - a new architecture for realtime application and protocols, November 1993.

[19] Xiaohua Jia. A distributed algorithm of delay-bounded multicast routing for multimedia applications in wide area networks. *IEEE/ACM Transactions on Networking*, 6(6):828–837, December 1998.

[20] S.K. Kasera, S. Bhattacharyya, M. Keaton, D. Kiwior, J. Kurose, D. Towsley, and S. Zabele. Scalable fair reliable multicast using active services. *IEEE Networks Magazine*, To appear.

[21] V. Kompella, J. Pasquale, and G. Polyzos. Multicast routing for multimedia communication. *IEEE/ACM Transactions on Networking*, 1(3):286–292, June 1993.

[22] Katia Obraczka. Multicast transport protocols: A survey and taxonomy. *IEEE Communications Magazine*, pages 1–15, January 1998.

[23] S. Raghavan, G. Manimaran, and C. Murthy. A rearrangeable algorithm for the construction of delay-constrained dynamic multicast trees. *IEEETNWKG: IEEE/ACM Transactions on NetworkingIEEE Communications Society, IEEE*

Computer Society and the ACM with its Special Interest Group on Data Communication (SIGCOMM), ACM Press, 7(4), 1999.

[24] W. Simpson. RFC 1853: IP in IP tunneling, October 1995. Status: INFORMATIONAL.

[25] Q. Sun and H. Langendoerfer. A distributed delay-constrained dynamic multicast routing algorithm. In *Lecture Notes in Computer Sciences in European Workshop on Interactive Distributed Multimedia Systems and Telecommunication Services (IDMS'97)*, volume 1309, pages 97–106, Springer-Verlag, 1997.

[26] D. Tennenhouse, J. Smith, W. Sincoskie, D. Wetherall, and G. Minden. A surevy of active network research. *IEEE Communication Magazine*, 35(1):80–86, January 1997.

[27] T. Turletti and J-C. Bolot. Issues with multicast video distribution in heterogeneous networks. In *6th Packet Video Workshop*, Portland, October 1994.

[28] D. Waitzman, C. Partridge, and S. E. Deering. RFC 1075: Distance vector multicast routing protocol, November 1988. Status: EXPERIMENTAL.

[29] D. Wetherall, J. Guttag, and D. Tennenhouse. Ants: A toolkit for building and dynamically deploying network protocols. In *IEEE OPENARCH'98*, 1998.

[30] David J. Wetherall and David L. Tennenhouse. The ACTIVE IP option. In *Proceedings of the Seventh ACM SIGOPS European Workshop*, Connemara, Ireland, September 1996.

[31] R. Widyono. The design and evaluation of routing algorithms for real-time channels. Technical report, University of California at Berkeley International Computer Science Institute, June 1994.

[32] Q. Zhu, M. Parsa, and J. J. Garcia-Luna-Aceves. A source-based algorithm for delay-constrained minimum-cost multicasting. In *Proceedings of the 14th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'95)*, pages 377–385, Los Alamitos, CA, USA, April 1995. IEEE Computer Society Press.

Appendix A

Theorems and proofs

Theorem A.1 *The probability that the immediate upstream active router will generate a new subscribe packet has an upper bound of* $\frac{1}{2}$.

Proof Let P_s be the probability that the immediate upstream active router will generate a new subscribe packet. Each active node has a state that correspond to its highest level of QoS. For a system with N levels of QoS, the nodes can take one of N states. The upstream node will generate a new subscribe packet only if the subscribing packet has a higher state. Let $P_{sub}(k)$ be the probability that the subscribing node is in state k. Let $P_{upstream}(i < k)$ be the probability that the upstream has a state lower than k. Then P_s is the sum of the probability that upstream has a lower state than the subscribing node for all states of subscribing node.

$$\begin{aligned} P_s &= \sum_{k=1}^{N} P_{sub}(k) \times P_{upstream}(i < k) \\ &= \sum_{k=1}^{N} \frac{1}{N} \times \frac{k-1}{N} \\ &= \frac{1}{N^2} \sum_{k=1}^{N} (k-1) \\ &= \frac{1}{N^2}(1+2+3+\ldots N-2+N-1) \end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{N^2}\frac{N(N-1)}{2} \\
P_s &= \frac{N-1}{2N} \\
&< \frac{N}{2N} = \frac{1}{2}
\end{aligned}
$$

Corollary A.2 *The expected number of subscribe packets generated in a single join operation has an upper bound of 2.*

Proof The packets generated includes the original *subscribe* packet from the receiver, and all additional packets generated on the way upstream. The expected number of additional packets, $E(N_{additional})$, is the sum of the product of the number of packets generated and the probability that *subscribe* packet is needed at each stage upstream. If necessary, only one *subscribe* packet is generated at each stage. $E(N_{additional})$ reduces into the sum of the probability of generating *subscribe* packet at each stage. For a network with depth D, $E(N_{additional}) = \sum_{k=1}^{D} {P_s}^k$. From Theorem A.1, $E(N_{additional}) < \sum_{k=0}^{D} (\frac{1}{2})^k < 1$. The expected number of packets generated, $E(N_{sub})$, is the sum of the original packet and the expected additional packets. $E(N_{sub}) = 1 + E(N_{additional}) < 2$.