Université d'Ottawa • University of Ottawa

HLA Streaming and Real-Time Extensions

By

Hui Zhao

Thesis submitted to
the Faculty of Graduate and Postdoctoral Studies of the University of Ottawa
in partial fulfillment of the requirements of the degree of
Doctor of Philosophy in Electrical Engineering

Ottawa-Carleton Institute of Electrical and Computer Engineering
School of Information Technology Engineering
University of Ottawa
Ottawa, Ontario, Canada

Abstract

The High Level Architecture (HLA) is IEEE's 1516 standard [1] which defines a software architecture for the development and execution of Distributed Interactive Simulation (DIS). It also provides a general-purpose network communication mechanism for DIS through its implementation – Run-Time Infrastructure (RTI). However, HLA does not address two critical features: stream (audio/video) and real-time transmission. This thesis focuses on solving these two issues.

The first part of this thesis reviews the basic concepts of DIS and its evolving history. Then it illustrates how the HLA supports the requirements of the DIS via its service groups, using a virtual shopping mall application as an example.

The second part introduces an HLA-compliant solution for stream transmission and control. An audio playback application is illustrated to show how to transfer stream data, using HLA defined APIs and control transfer schedule, using the operating system interval timer. Then, a presentational audio-video continuous-media retrieval application is illustrated to show how to control stream retrieval.

In the third part, a real-time extension proposal to HLA and an architecture of real-time RTI is presented. The Internet is moving to Quality of Servive (QoS) age, which will provide delay and jitter bounded services. With the IP QoS and real-time operating systems involved, it is shown that HLA can support real-time DIS by taking advantages of these new technologies. After analyzing the limitation of current HLA and RTI, a proposal of real-time extension to HLA and the architecture of real-time RTI is illustrated.

In the fourth part, our experiences of building QoS networks and a performance analysis, including end-to-end RSVP for unicast and multicast, end-to-end DiffServ, RSVP over differentiated service network, etc, are presented.

We hope that the thesis' contributions would create a discussion in the DIS community on the topic of Real-Time DIS (RT-DIS). With new real-time technologies in operating systems and networks blooming, RT-DIS has the possibility of coming true, while its final realization depends on the effort of the whole DIS community.

Publications

The following publications by the authors are relevant to the work in this thesis:

Papers in refereed Journals

- H. Zhao and N. D. Georganas, "Collaborative Virtual Environments: Managing the Shared Spaces", Networking and Information Systems Journal, Vol.3, No.2, 2001. ISSN 1290-2926. Selected by the book: Innovative Technology: Information Systems and Networks, Spring and Summer 2002, Publications: Joint Venture of Hermes Publications in Paris and Kogan Page Publishers in London.
- H. Zhao and N. D. Georganas, "HLA Real-Time Extension", submitted to Concurrency and Computation: Practice and Experience. Journal. (Selected by the IEEE DS-RT 2001 Conf). ISSN: 1532-0626.
- H. Zhao and N. D. Georganas and M. Hosseini, "Building QoS Networks", submitted to Computer Communication Journal, ISSN: 0140-3664.
- H. Zhao and N. D. Georganas, "HLA Stream Solutions", submitted to ACM Network Multimedia, ISSN: 0942-4962.

Papers in refereed conference proceedings

- H. Zhao and N. D. Georganas, "An Approach for Stream Transmission over HLA-RTI in Distributed Virtual Environments", Proc. IEEE/ACM Third International Workshop on Distributed Interactive Simulation and Real Time Applications (D I S - R T ' 99), Greenbelt MD, Oct. 1999
- H. Zhao and N. D. Georganas, "An Approach for Stream Retrieval over HLA-RTI in Distributed Virtual Environments", Proc 4th IEEE D S - R T ' 2000 Fourth IEEE International Workshop on Distributed Simulation and Real Time Applications, San Francisco, Aug 2000
- H. Zhao and N. D. Georganas, "HLA Real-Time Extension", Proc 5th IEEE D S – R T ' 2001 Fifth IEEE International Workshop on Distributed Simulation and Real Time

Applications, Cincinnati, Ohio, Aug.2001

- H. Zhao and N. D. Georganas, "Enabling Technologies for Real-Time Distributed Simulation", Proc. 2001 Fall Simulation Interoperability Workshop, Orlando, Florida, USA, Sept. 2001
- H. Zhao and N. D. Georganas, "HLA Real-Time Extension", Proc. 2001 Fall Simulation Interoperability Workshop, Orlando, Florida, USA, Sept. 2001
- H. Zhao and N. D. Georganas, "Architecture Proposal of Real-Time RTI", Proc. 2001 Fall Simulation Interoperability Workshop, Orlando, Florida, USA, Sept. 2001

Acknowledgements

A number of people have, directly or indirectly, contributed to this thesis. It is my pleasure to have the opportunity to thank them.

First of all, I would like to thank my thesis supervisor, Dr. Georganas. His encouragement, support and guidance have been very instrumental during my Ph.D studies. I also extend my thanks to my committee, Dr. Emil M. Petriu, Dr. Dorina C. Petriu and Dr. Luis Orozco-Barbosa, and my examiners for reviewing and commenting this thesis: Dr. A. Boukerche (external examiner from University of North Texas, USA), Dr. D. Ionescu, Dr. D. Makrakis, and Dr. C. Huang.

I would like to express my eternal gratitude and love to my wife, Hong Zhang. Her love and support gave me enough strength to complete my Ph.D study.

Many thanks to ladies and gentlemen who helped me during my research project – Garth Scully, Kimberly D. Shorb, Nat Landry, and Paula Cobb. Mr. Garth Scully, the Capital and Atlantic Regional Manager of Cisco Systems Canada, sponsored my research with Cisco's advanced routers. Ms. Kimberly D. Shorb, the Chariot Product Manager of NetIQ Corporation, sponsored my research with NetIQ's advanced performance measurement tools. Ms. Nat Landry and Mr. Mike Halhed, senior engineer and manager of Cisco, helped me a lot for preparing and configuring the routers.

Then, I would like to express my gratitude to the members of the HLA standard committee and Simulation Interoperability Standards Committee - Phil M. Zimmerman, Steve Bachinsky, Reed Little, Bill Waite, Fred Wieland, Mike Lightner, Katherine L. Morse, Tom Lake, Susan Symington, Arlund Ronald, Mark Crooks, for the numerous discussions that I had with them through the Internet. I would like to thank them for sharing their experience on HLA standard.

I am also indebted towards my friends and mentors Yanhe Fan, Francis Hung, Ivan Chow. Their encouragement made me persevere in my study for years.

Last but not least, I would like to thank my buddies in MCRLab, especially Francois Malric, Mojtaba Hosseini, Nawel Chefai, Ziad Sakr, for their support and help.

Table of Contents

Part IV. Building Network for RT-DIS

List of Figures

List of Tables

Abbreviations

AAA	Authentication, Authorization and Accounting
ALSP	Aggregate Level Simulation Protocol
ADS	Advanced Distributed Simulation
AS	Autonomous System
CLS	Controlled-Load Service
DAC	Dial Access Client
DARPA	Defence Advanced Research Projects Agency
DDM	Data Distribution Management
DiffServ	Differentiated Services
DIS	Distributed Interactive Simulation
DM	Declaration Management
DMF	Dynamic Multicast Filtering
DMSO	Defense Modelling and Simulation Office
DPC	Deferred Procedural Call
DSBM	Designated SBM
DVE	Distributed Virtual Environment
FIFO	First-In-First-Out
FM	Federation Management
FOM	Federation Object Models
GPOS	General Purpose Operating System
GQOS	Generic Quality of Service
GS	Guaranteed Service
HLA	High Level Architecture
HMI	Human Machine Interface
IEEE	Institute of Electrical and Electronic Engineers
IETF	Internet Engineering Task Force
IntServ	Integrated Services
ISC	Internet Service Client
ISP	Internet Service Provider
ISR	Interrupt Service Routine
ISTP	Interactive Sharing Transfer Protocol
M&S	Modeling and Simulation
MPLS	Multi-Protocol Label Switching
NAC	Network Access Client
NAP	Network Access Point
OM	Object Management

OMG	Object Management Group
OMT	Object Model Template
OWM	OWnership Management
POSIX	Portable Operating System Interface
QoS	Quality of Servive
RED	Random Early Detection
RMS	Rate Monotonic Scheduling
RR	Round-Robin
RSVP	Resource Reservation Protocol
RT-DIS	Real-Time Distributed Interactive Simulation
RTI	Run-Time Infrastructure
RTOS	Real-Time Operating System
SBM	Subnet Bandwidth Manager
SIMNET	Simulation Networking
SLA	Service Level Agreement
SOM	Simulation Object Models
SPLINE	Scalable Platform for Large Interactive Networked Environments
STP	Spanning Tree Protocol
TM	Time Management
TOS	Type of Service
VPN	Virtual Private Network
WFQ	Weighted Fair Queue

Part I

Distributed Virtual Environments: Managing the Shared Spaces

Chapter 1
Introduction

1.1. Distributed Interactive Simulation to Distributed Virtual Environments

[1]The venerable problem solving technique of simulation finds itself in the midst of a revolution. As an economical, timesaving and safety solution, simulation is widely applied today to support a myriad of purposes, including: training, interaction, visualization, hardware testing, and real-time decision support.

Distributed Interactive Simulation (DIS) is a newer concept, accompanying the development of computer network technology. DIS refers to the technology of executing simulation entities over multiple computers connected via a network through which simulation entities' messages are exchanged.

DIS adds advanced features to simulation, including: reduced model execution time; scalable performance; geographically distributed users and/or resources; integrated simulations running on different platforms; fault tolerance.

Due to the extraordinary development of computer networks, the distributed interactive simulation (DIS) plays a more and more important role in the simulation community. In general, there are three categories of DIS applications. The first category is pure simulation, which means that all simulation entities are computer programs, for example, weather forecast simulation, celestial mechanics simulation, etc. The second category is the people-in-the-loop simulation that the human acting as user or trainee closes the control loop. This is widely used in virtual shopping malls, collaborative design and engineering, collaborative augmented reality for sharing spaces, multi-user virtual conferencing, shared virtual environment for training (pilot training), network interactive 3D games, etc [2]. The third category is equipment-in-the-loop simulation in which at least one actual equipment is connected to the simulation network and can interact with other computer program simulation entities. Example applications include airplane design and testing, weapon systems design and testing, etc.

As a branch of DIS, the Real-Time Distributed Interactive Simulation (RT-DIS) refers to a DIS with special time requirements. The term "real-time", as it relates to simulation, requires that the computer program execution of a modelled, dynamic process must occur within specified time deadlines (i.e., not faster or slower) [3]. It represents or simulates a real, dynamic phenomenon as it occurs. Real-time simulation allows a more realistic representation of the physical system being studied (as opposed to pure mathematical analysis or standard computer analysis) and permits both quantitative and qualitative (human analysis) evaluation [3]. Obviously, all people-in-the-loop simulations and many equipment-in-the-loop simulations are some kind of RT-DIS application with different time critical level.

As a specific application of RT-DIS, a Distributed Virtual Environment (DVE) is a software system in which multiple users interact with each other in real-time, even though those users may be physically located in different places around the world. The DVE enabled solution will remarkably affect many applications that demand a high quality Human Machine Interface (HMI), such as e-Commerce, network entertainment, distance learning, etc. The DVE systems usually aim at providing users with a sense of realism by incorporating realistic 3-D graphics and vivid sound to create an immersive experience.

1.2. State of the Art in Advanced Simulation Technology and Standards

1.2.1. Simulation Networking (SIMNET)

In the mid 1980's the U.S. Department of Defence, Defence Advanced Research Projects Agency (DARPA), launched the SIMNET project and generated a real-time vehicle-level distributed interactive (virtual) simulation system. In SIMNET [4], individual vehicle simulators are connected via a computer network, permitting them to coexist in a common, shared simulation environment and to interact (e.g. engage in combat) through the exchange of information packets on the network that connects them. SIMNET simulators usually each represent a single tank or vehicle. SIMNET is used to train tank and vehicle crews in cooperative team tactics.

SIMNET follows four design principles:

- Distributed computation based on combat entity ownership,
- Avoidance of single critical resources,

- Reliance on broadcast communications, and
- Replication of a limited set of combat entity attributes among all simulations.

SIMNET was the first step for distributed interactive simulation.

1.2.2. Aggregate Level Simulation Protocol (ALSP)

In early 1990, DARPA sponsored MITRE to investigate the design of a general interface between large, existing, aggregate-level combat simulations. MITRE presented the ALSP[5] system and adopted all the principles from SIMNET. In addition, aggregate-level simulations have unique requirements for time and data management that are addressed by services specific to ALSP. ALSP provides time management services to coordinate simulation times and preserve event causality across simulations. The data management scheme allows each simulation to use its own representation of data. Simulations share information in a commonly understood manner independent of each simulation's internal data representation. The typical application based on ALSP is the Joint Training Confederation (JTC) [5].

The main difficulty of ALSP has been its inflexibility. In order to incorporate a new simulation in the federation, considerable re-writing of the existing members is needed.

1.2.3. Distributed Interactive Simulation (DIS)

The DIS protocol is intended to replace its precursor – the SIMNET protocol. DIS had larger ambitions than SIMNET. SIMNET devices used the same technology, whereas DIS was aimed at simulators using dissimilar technology [6]. In 1993, DIS was approved as the IEEE 1278 series standards.

Through the use of the DIS protocol standard, DIS integrates traditional simulator technologies with computer communication technologies to create a system that provides a common field on which the various simulators can interact in active, real-time situations.

DIS was founded on a few basic concepts: multiple entities simulation; no central node; autonomous simulation nodes; standard communications protocol; receiving nodes perception; local maintenance of other nodes' states.

Although it was a great improvement to SIMNET, several major problems associated with scaling the current suite of DIS protocols illustrate the difficulty of building large virtual environments. First, enormous bandwidth and computational requirements were required for large-scale simulation; Second, multiplexing of different media at the application layer was needed; Third, DIS lacked an efficient method for handling static objects; Lastly, models and world databases must be replicated at each simulator.

1.2.4. Advanced Distributed Simulation (ADS)

There are areas of simulation where DIS may not be appropriate or meet the timing or data transmission rates required. Founded on DIS, ADS [7] uses similar principles that DIS is founded upon, but allows for use of protocols and methodologies outside of the DIS standards [8]. The term ADS includes DIS as a subset and is intended to support a mixture of virtual, live, and constructive entities.

ADS is more conceptual and therefore more flexible than DIS. DIS can be thought of as one specific implementation of the ADS concept. The High Level Architecture (HLA) is another example of an implementation of the ADS concept [8].

1.2.5. Scalable Platform for Large Interactive Networked Environments (SPLINE)

To support their work on social virtual reality, Mitsubishi Electric Research Laboratories (MERL) researchers designed and implemented the SPLINE middleware system with many features [9]: multiple users; spoken interaction; 3D graphics and sound; run-time modifiability to the environment; open Interfaces at both the network and application programmer levels.

SPLINE provides development APIs and libraries. Such libraries provide very detailed and essential services for real-time multi-user cooperative applications. For its communication, SPLINE uses the Interactive Sharing Transfer Protocol (ISTP), which is a hybrid protocol supporting many modes of transportation for VR data and information through five sub-protocols, namely: 1-1 Connection; Object State Transmission; Streaming Audio; Locale-Based Communications and Content-Based Communication sub-protocols.

SPLINE partitions the World Model in *locales* which may have any shape. Once a user joins a given *locale* everything which is located in that *locale* as well as the *locales* with immediate neighbourhood would be visible. It is possible to be "present" in more than one *locale* by using spObserver objects. Internally a spObserver object makes the local SPLINE engine to listen to the multicast group of that *locale* (and its neighbourhood).

SPLINE has the advantage of providing audio communication and audio/visual rendering modules. But since it derives from SIMNET and DIS, the major problems of the DIS system also exist in SPLINE.

1.2.6. High Level Architecture (HLA)

HLA is the next generation DIS, adopting new architecture and bandwidth reduction techniques. HLA is inspired by all previous architectures and protocols and designed to supplant both DIS and ALSP [10].

HLA was proposed by DARPA/DMSO for distributed simulation in 1996. It represents an attempt to build on ALSP and DIS and to apply the lessons learned to provide a much more permanent foundation and framework for distributed simulation in the future. Every simulation developed for the US Department of Defense is expected to conform to HLA from Oct.1, 2000. The Object Management Group (OMG) adopted HLA as the Facility for Distributed Simulation Systems 1.0 in November 1998 [11]. More recently (Nov. 2000), HLA was approved as the IEEE 1516 series standards for distributed interactive simulations [1]. Obviously, the HLA will be the infrastructure for future distributed simulations.

1.3. High Level Architecture Introduction

1.3.1. High Level Architecture Standards

The HLA standard for modeling and simulation (M&S) consists of three specifications:

- IEEE1516 – Framework and Rules [12]

This is the top level base document of a family of related High Level Architecture (HLA) documents. It defines the HLA, its components, and the responsibilities of federates and federations.

- IEEE1516.1 – Federate Interface Specification [13]

  Defines the functional interface between federates and the HLA run time infrastructure (RTI), including six service groups: federation management, declaration management, object management, ownership management, time management and data distribution management.

- IEEE1516.2 – Object Model Template (OMT) Specification [14]

  Provides a specification for documenting key information about simulations and federations. OMT is used to describe Simulation and Federation Object Models (SOMs and FOMs).

Interoperability and reusability are major drivers for the HLA. Interoperability is the ability of system components to exchange data and the ability of those components to interpret the data in those components to interpret the data in a consistent way. Reusability is facilitated by having modular components with commonly understood behaviours and well-defined interfaces through which a variety of client applications can access the components.

1.3.2. Architecture and Mechanisms

Federation and federates are two important concepts in HLA. Federation is a named set of interacting simulation entities that are used as a whole to achieve some specific objective. A Federate is a simulation entity running as a member of a HLA Federation

that provides specific function to other federates. All applications participating in a Federation are called Federates.

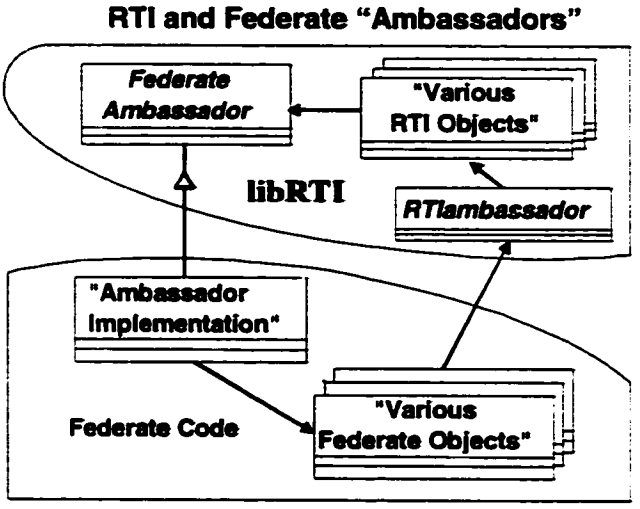


Figure 1.1 Federate and LibRTI [15]

The information is exchanged among federates through a kind of group communication mechanism. Data source federates publish a variable to a federation and data consumer federates subscribe to the same variable. For each variable, there is a group containing all data consumer federates. When the source federate updates the variable, all the consumer federates will be notified and get the updated value of the variable. The variable update is implemented using a multicast mechanism.

Run-Time Infrastructure (RTI) is a middleware that implements the interface specification of the HLA standard. It provides services in a manner that is analogous to the way a distributed operating system provides services to applications [15]. RTI provides a C++ library, libRTI, through which a federate developer can use the services specified in the HLA Interface Specification (refer to Figure 1.1).

Within libRTI, the class *RTIambassador* bundles the services provided by the RTI. All requests made by a federate on the RTI take the form of a *RTIambassador* method call. The abstract class *FederateAmbassador* identifies the callback functions each federate is obliged to provide.

More than twenty companies and institutes are manufacturing and supporting RTI and OMT software. A vendors list could be found at DMSO HLA homepage [16].

1.3.3. HLA Applications

The objective for military application of HLA is to construct a rapidly configured mix of computer simulations, actual war-fighting systems, and weapons systems simulators geographically distributed and networked, involving tens of thousands of entities to support training, analysis, and acquisition. Such simulations would be used both to train individuals to perform particular tasks, to interpret data, and to make decisions, and to help groups of individuals (tank crews, fighter squadrons) work together as a team.

Besides new developed HLA based simulations, all legacy military simulations are being ported to HLA since every simulation developed for the US Department of Defense is expected to conform to HLA from Oct.1, 2000.

For example, the U.S. Army's Tank-Automotive Research Development and Engineering Center's (TARDEC) VETRONICS Simulation Facility (VSF) and Physical Simulation Laboratory (PSL) are integrating man-in-the-loop distributed simulation with two six-degree of freedom motion based systems [17] (refer to Figure 1.2). The VSF is capable of networking with other simulators via the Distributed Interactive Simulation

(DIS) Protocol. They declared that the VSF would migrate from a DIS based simulation facility to a fully compliant High Level Architecture (HLA) facility [17].

Figure 1.2 TARDEC's VSF and PSL [17]

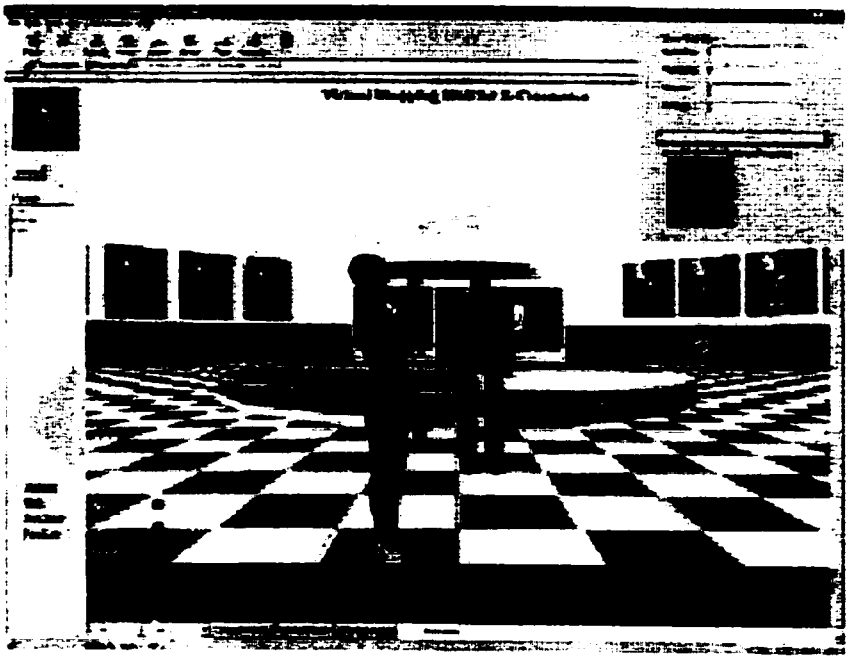


Figure 1.3 MCRLab's Virtual Shopping Mall [21]

The civil industry would also benefit from HLA for many applications. Roger Smith, the technical director of BTG inc, USA [18] emphasised in the IEEE DS-RT2000 workshop [19] that the distributed simulation would go beyond military applications. It will be widely used in many civil fields, including: massively multi-user games, distributed theme parks, commercial distributed computing, weather analysis and prediction, Internet traffic analysis, stock market prediction, real-time control, traffic/air traffic control, distributed engineering, movie production, virtual conference, e-commerce, etc.

The virtual shopping mall [20,21] developed at the Multimedia Communication Research Labortary (MCRLab), the University of Ottawa, is a good example (refer to Figure 1.3 [21]). With the creation of a virtual shopping mall, simulations of most of the actual shopping environments and user interactions can be achieved. The virtual mall brings together the services and inventories of various vendors and provides customers with the same shopping experience, as they would have in an actual store or shopping mall. Real-time interactions among entities in the virtual environment, such as Distributed-shopping and shopper-vendor avatar interactions, are implemented over the HLA-RTI.

More HLA based simulations and applications could be found in the papers of the Simulation Interoperability Workshop [22].

1.4. This Thesis

1.4.1. Thesis Organization

The first part of this thesis (Chapters 1-2) reviews the basic concepts of distributed interactive simulation and its evolving history from SIMNET to HLA. Then it illustrates how the HLA supports the requirements of the DIS via its service groups, using a virtual shopping mall application as an example.

The second part (Chapters 3-5) introduces an HLA-compliant solution for stream transmission and control. A RTI-based audio playback application is illustrated to show how to transfer stream data, using RTI-APIs and control transfer schedule, using the operating system interval timer. Then, a presentational audio-video continuous-media retrieval application is illustrated to show how to control stream retrieval. Finally, an architecture of an HLA-compliant videoconference application is presented.

In the third part (Chapters 6-8), a real-time extension proposal to HLA and an architecture of real-time RTI is presented. The Internet is moving to Quality of Servive (QoS) age, which will provide delay and jitter bounded services. With the IP QoS and real-time operating systems involved, it is shown that HLA can support real-time DIS by taking advantages of these new technologies. After analyzing the limitation of current HLA and RTI, a proposal of real-time extension to HLA and the architecture of real-time RTI is illustrated.

The network is a critical part of DIS. The DIS network is moving from an isolated proprietary network to the public Internet. How to guarantee quality requirements of DIS is important. In the fourth part (Chapters 9-10), our experiences of building QoS

networks and a performance analysis, including end-to-end RSVP for unicast and multicast, end-to-end DiffServ, RSVP over differentiated service network, etc, are presented. Chapter 11 presents our conclusions.

1.4.2. Original thesis contributions

- Illustrate the usages of HLA features for managing a shared space of DVE
- Invent a stream transmission and control mechanism using HLA compliant features.
- Propose a real-time extension to the HLA standard
- Propose an architecture of real-time RTI
- Present verification results of various QoS mechanisms of IP network.
- Present practical experience of building QoS network for real-time DIS.

Chapter 2
Managing the Shared Spaces

2.1. DVE Requirements

Singhal and Zyda summarized the common features of networked virtual environments [23], which is also suitable for Collaborated Virtual Environments:

- A shared sense of space: All participants are presented with the illusion of being located in the same place, such as in the same room, building, or terrain.
- A shared sense of presence: When entering the shared place, each participant takes on a virtual persona, called an avatar, which includes a graphical representation, body structure model, motion model, physical model, and other characteristics.
- A shared sense of time: Participants should be able to see each other's behavior as it occurs. In other words, the DVE should enable real-time interaction to occur.
- A way to communicate: Though visualization forms the basis for an effective DVE, most DVEs also strive to enable some sort of communication to occur among the participants.
- A way to share: The user's ability to interact with the virtual environment itself and other users.

For a large scale, shared space of DVE, the scalability issues should also be considered. Scalability is the ability of a distributed simulation to maintain time and spatial consistency as the number of entities and accompanying interactions increases [24]:

- Awareness/Visibility Control: The ability of transferring only required data and avoiding transferring non-required data.
- Dynamic Simulation Entity Management: The ability of adding/deleting a simulator into the run-time simulation without negative effects.
- Simulation backup and restore: The ability of recording and retrieving simulation status.

HLA provides a sophisticated infrastructure to satisfy these requirements through the six service groups defined in the HLA Interface Specification:

- Federation Management
- Declaration Management
- Object Management
- Ownership Management
- Time Management
- Data Distribution Management

We are going to illustrate how these services are used to manage the shared space of DVE through the virtual shopping mall example.

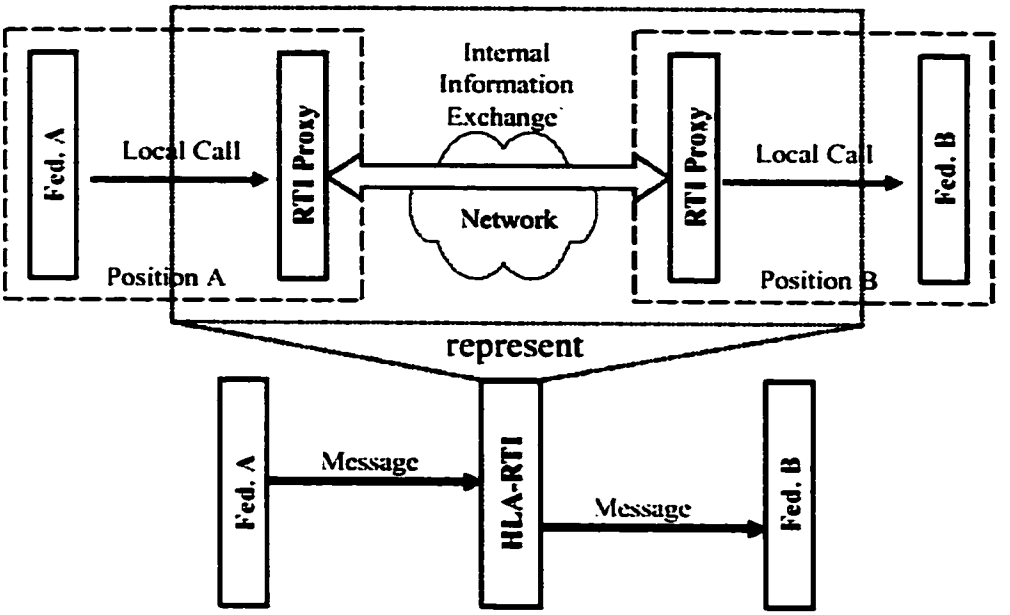


Figure 2.1 HLA-RTI Representation

HLA is a distributed framework. A federation consists of many federates. Each federate acts a role in the federation. There is a HLA proxy located locally with each federate. The data exchange among federates is completed by HLA proxies and federates do not care about how HLA accomplishes that (refer to Figure 2.1). From the perspective of the programmer, all HLA services can be accessed through a local call, and all network involved activities are encapsulated in the HLA-RTI.

2.2. Federation Management (FM)

"Federation Management" refers to the creation, dynamic control, modification, and deletion of a federation execution [13]. The basic federation management activities are shown in Figure 2.2. Before a federate may join a federation execution, the federation must be created. The first federate is in charge of creating the federation and then other federates may join and resign from it in any sequence.

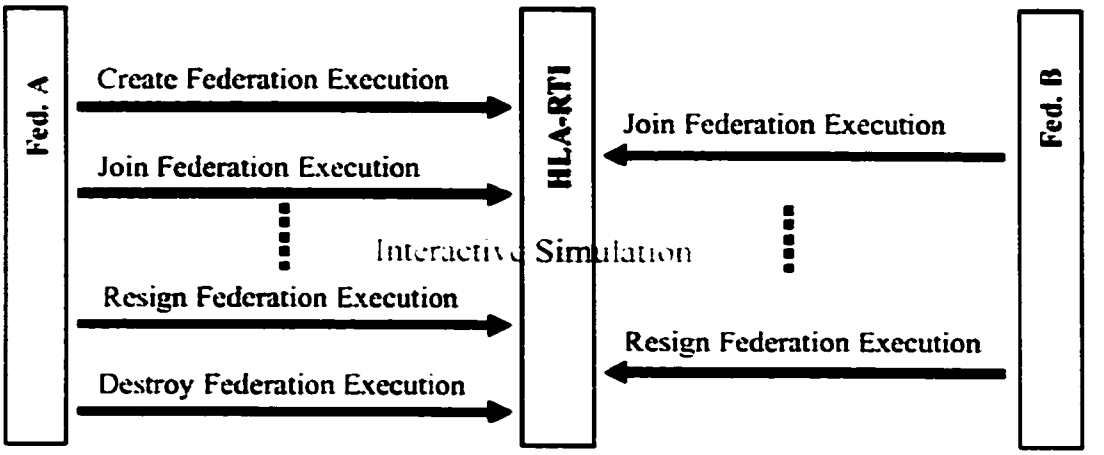


Figure 2.2 Federation Management – Create, Join, Resign and Destroy

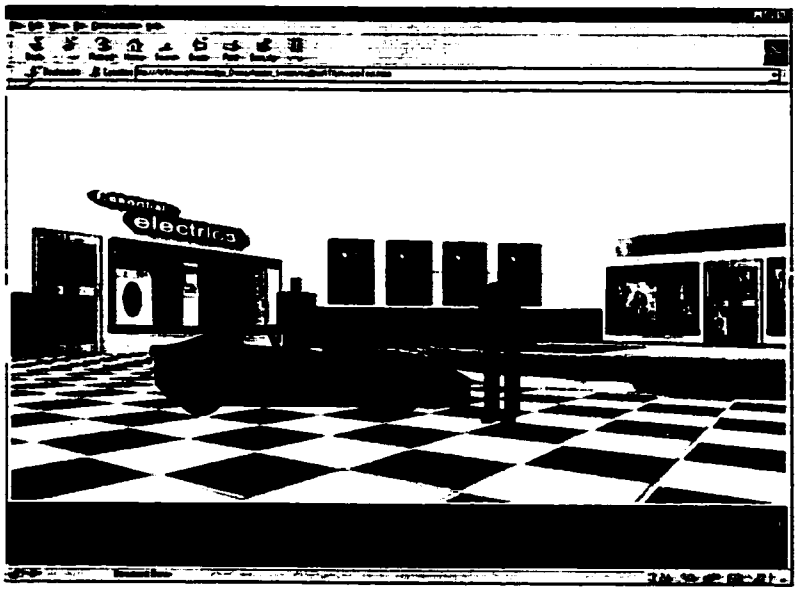


Figure 2.3 Federation Management – Virtual Shopping Mall

In the virtual shopping mall application (refer to Figure 2.3), the shopping mall is a federation, and all customers, dealers, products may be represented as federates. From a technical perspective, customers entering the shopping mall are considered as federates joining the federation. When they leave the shopping mall, the related federates resign from the federation.

2.3. Declaration Management (DM) and Object Management (OM)

Before federates in a federation can see each other and start exchanging information, they must tell the federation what data they can provide to others and what data they need from others. These requirements are handled through Declaration Management and Object Management.

The HLA demands two mechanisms for sharing information between federates: objects and interactions. *Object classes* are comprised of attributes. Object classes describe *types* of things that can *persist. Interaction classes* are comprised of parameters. Interaction classes describe *types* of events [15]. The primary difference between objects and interactions is *persistence*. Objects persist, interactions do not. In addition, object attributes can be identified and updated individually while interaction parameters can not be identified individually and must be updated based on whole interaction unit. In other words, programmers can update specific object attributes of an object and leave other attributes unaffected but they can not do the same thing to interactions. When they update interaction, all parameters of the interaction will be affected. Obviously, objects are good at presenting the long live status, while interactions are good for messages.

Declaration Management provides the function to federates to declare their intention to generate and consume information. Object Management deals with the registration, modification, and deletion of object instances and the sending and receiving of interactions [13].

The basic activities of DM and OM services for object classes operations are shown in Figure 2.4.

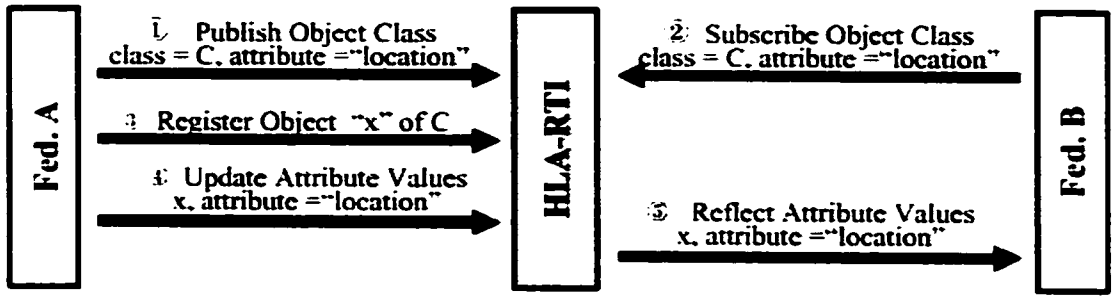


Figure 2.4 Declaration Management – Object Class

Step (1) indicates that the federate A may subsequently register object instances of object class "C" and provides the information of its attribute "location". Step (2) indicates that the federate B needs the "location" information of any object instances of object class "C". Similar to the instantiation concept of Object Oriented Programming (OOP), published object classes in HLA also need instantiation before using it and multiple objects can be instantiated from one published object class. In step (3), federate A notifies RTI that it creates a unique object instance "x" from the object class "C". When federate A updates the attribute "location" of the object instance "x" in step (4), RTI will notify federate B of the new value of the "location" in step (5).

The basic activities of DM and OM services for interaction classes operations are shown in Figure 2.5.

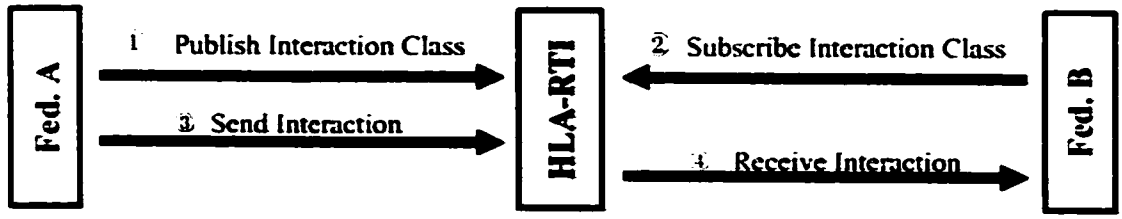


Figure 2.5 Declaration Management – Interaction Class

The interaction class operation is similar to object class operation. Federates must publish and subscribe their interested interaction class before using it. But an interaction

unit does not need registration. The sender federate constructs the interaction, fills all its parameter fields and then sends it out. The receiver federate gets the whole interaction through RTI notification.

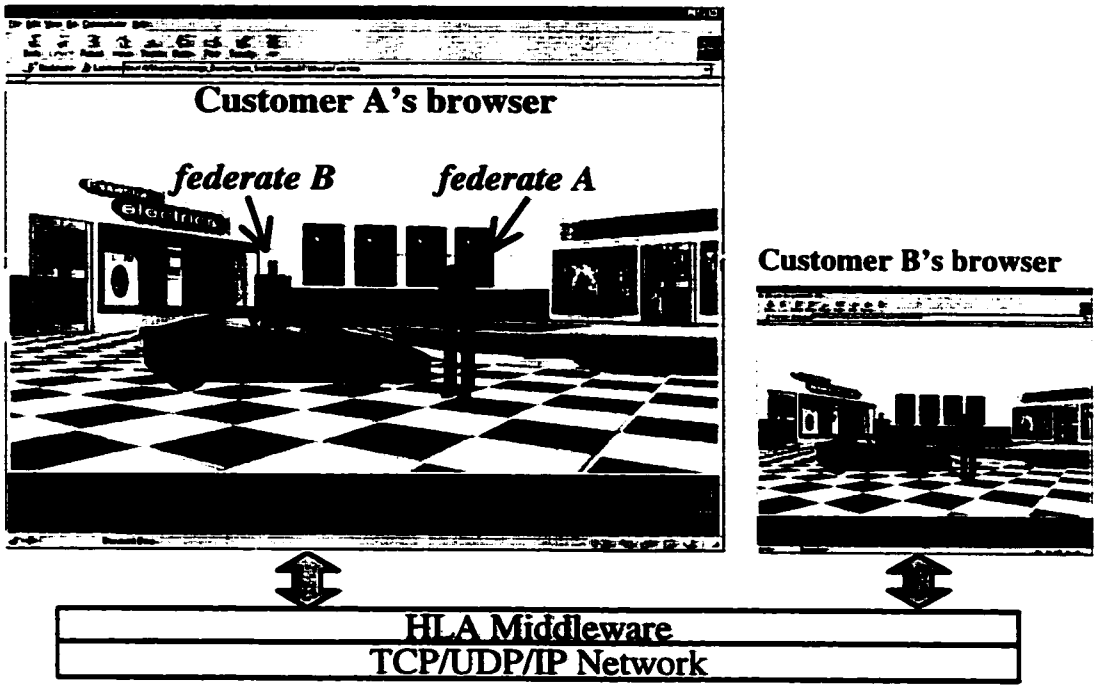


Figure 2.6 Declaration Management Example

In the virtual shopping mall example, all avatars declare their position so that they can see each other. Their position data are updated when they are moving, so other federates can get the new position value and update the customers' browsers respectively (refer to Figure 2.6).

With the same mechanism, avatars can also declare their other behaviors so that they can interact with each other. For example, the interaction mechanism can be used to transfer short messages among federates. All avatar federates publish and subscribe to the same interaction named "BBS", so any BBS interaction sent out from one avatar federate can be received by all other avatar federates. Then a BBS system is generated.

2.4. Ownership Management (OWM)

Ownership management is used by federates and the RTI to transfer ownership of instance attributes among federates [13]. Only the federate that owns an instance attribute can update the attribute values. The ownership management methods provide a facility for exchanging attribute ownership among federates in a federation execution using a "push" and/or a "pull" model [15]. The push model refers to that a federate can try to give away responsibility for one or more attributes of an object instance, refer to Figure 2.7. Alternatively, the pull model refers to that a federate can try to acquire responsibility for one or more attributes of an object instance (refer to Figure 2.8).

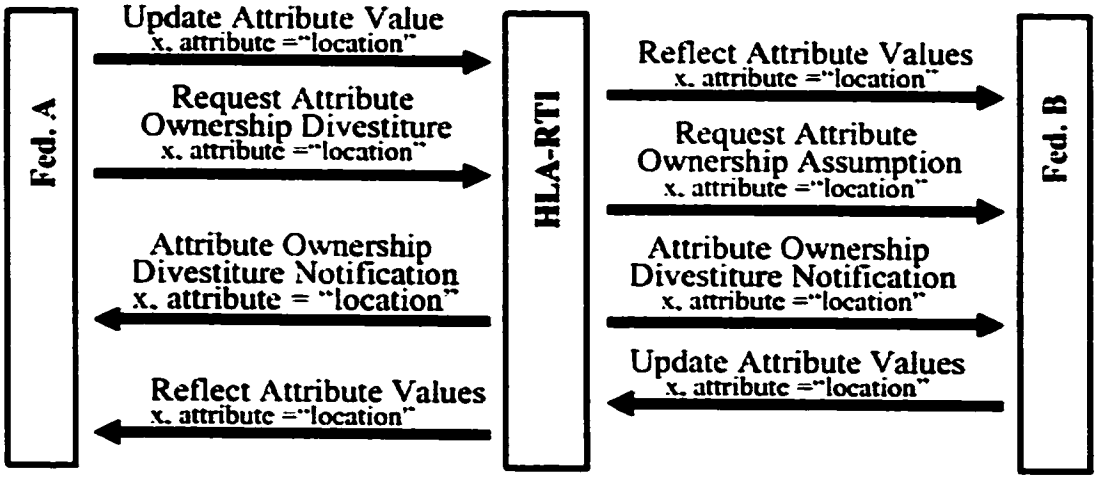


Figure 2.7 Ownership Transfer – Push

As push model of ownership transfer is shown in Figure 2.7. Federate A owns attribute "location" for object x and publishes updates for the "location". The Federate A requests attribute ownership divestiture of attribute "location" for object x. Then the RTI issues a request for ownership assumption to federate B. Then RTI notifies both A and B that the attribute ownership is exchanged. From this point on, the federate B begins issuing updates for attribute "location" for object x.

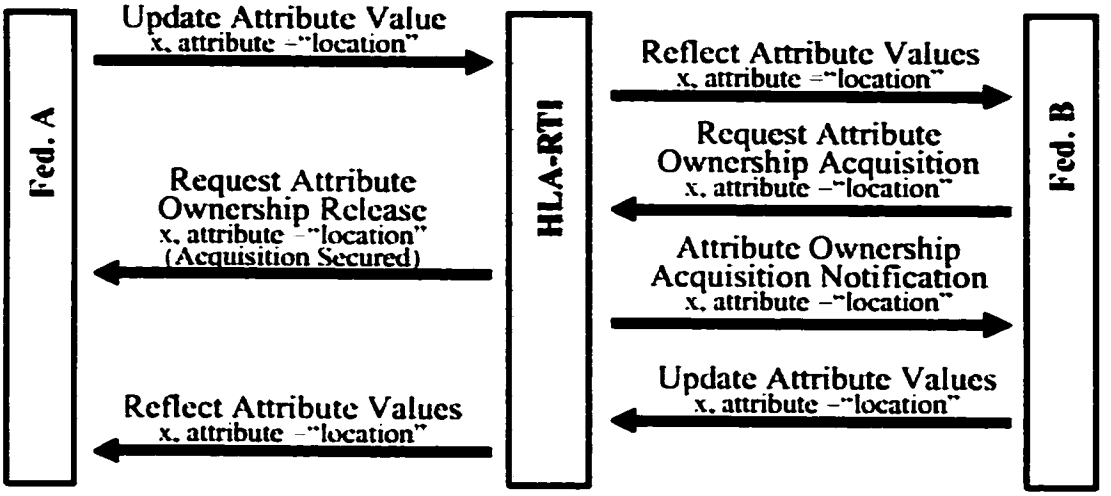


Figure 2.8 Ownership Transfer – Pull

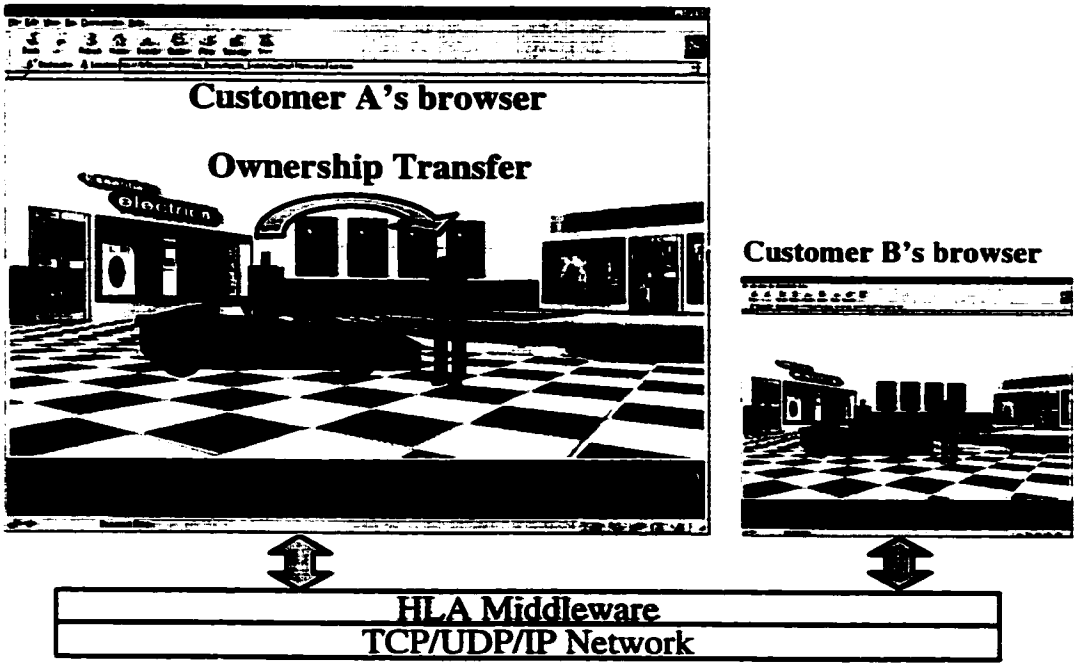


Figure 2.9 Ownership Transfer in the Virtual Shopping Mall

A pull model of ownership transfer is shown in Figure 2.8. Federate A owns attribute "location" for object x and publishes updates for the "location" at the beginning. Federate B requests attribute ownership acquisition of attribute "location" for object x. Then the RTI issues a request for attribute ownership Release to the owner of object x, that is

federate A. Then the RTI receives the response that federate A will release the ownership of the attribute. Then the RTI notifies that B gets the ownership. From this point, the federate B begins issuing updates for attribute "location" for object x.

Let's see the utility of ownership exchange in the virtual shopping mall application (refer to Figure 2.9). Assume the ownership of the car object belongs to the salesman federate at the beginning. It is very possible that the customer wants to try its options by himself, such as change of color, change of body style, etc. At this situation, the salesman can transfer the ownership of the car to the customer and then the customer can try it. Of course, if the salesman does not give up the ownership, the customer cannot get it.

2.5. Data Distribution Management (DDM)

The HLA effectively serves as an intelligent switch - matching up producers and consumers of data, based on declared interests and without knowing details about the data format or content being transported. Furthermore, DDM provides a flexible and extensive mechanism for further isolating publication and subscription interests - effectively extending the HLA's switching capabilities.

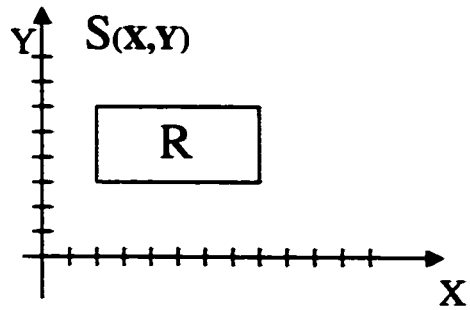


Figure 2.10 Concept of Routing Space and Region

In DDM, a federation "routing space" is defined. A Routing Space is defined as a multidimensional coordinate system and a Region is defined as a scope in a routing space. As the example shown in Figure 2.10, S(X,Y) is a routing space; X, Y are two dimensions of S; R(x:2-8,y:3-6) is a region.

Figure 2.11 is an example given in the RTI Programmer's Guide [15] about the Routing Space of DDM which defined by the three dimensions "longitude", "latitude", and "altitude".

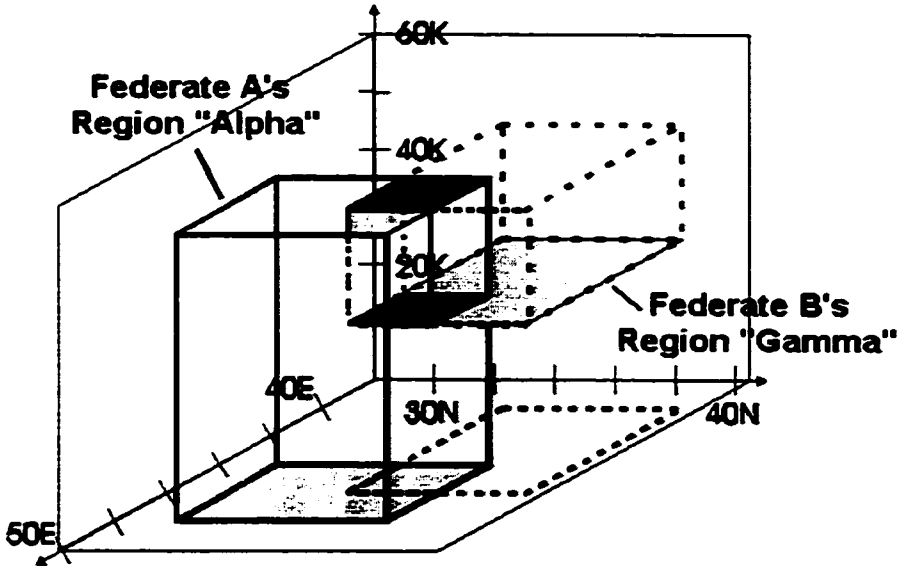


Figure 2.11 Routing Space and Region in DDM [12]

Federates can fine-tune their subscription declarations and data updates in terms of regions within the routing space. Federate A might publish its objects class X within the region RAlpha{longitude: 44°E - 48°E, latitude: 30°N - 37°N, altitude: 0 - 50,000 ft}, and federate B might subscribe to same objects class X within the region RGamma{longitude: 40°E - 46°E, latitude: 34°N - 40°N, altitude: 30,000 ft - 50,000 ft}. We can see that the overlap between the two regions RAlpha∩Gamma{longitude: 44°E - 46°E, latitude: 34°N - 37°N, altitude: 30,000 ft - 50,000 ft} is relatively small. DDM is in

charge of judge the overlap and make sure that only updates fall into the overlap region be notified to subscribed federates. For example, the federate A's update to object instance x with value {45°E, 36°N, 40,000ft} will be notified to federate B, while {***43***°E, 36°N, 40,000ft}, {45°E, ***39***°N, 40,000ft} and {45°E, 36°N, ***10,000***ft} will not be notified to federate B.

Figure 2.12 shows the declaration object management usage associated with DDM. Federate A offers to publish object class C with attribute "location" with region Alpha. Federate B subscribes to class C and attribute "location" with region Gamma. Federate B can discover the overlap of Alpha and Gamma. When federate A update the attributes within the overlap, simulator B can get the reflection. When federate A update the attributes out of the overlap, simulator B does not be reflected.

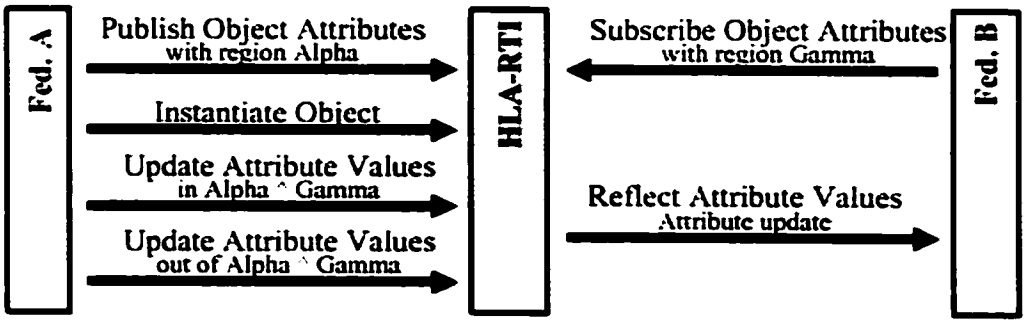


Figure 2.12 Data Distribution Management

Back to the virtual shopping mall application (refer to Figure 2.13), customers are interested in the products in the store that they visit. They are interested in what the salesman is doing, even more, they are interested in what the other customers are doing in the same room. But definitely, they don't want HLA to send to their computer all events and status of the whole shopping mall. Obviously, any personal computer cannot handle

that. More importantly, customers are not interested in what happens in other stores. Thus the data distribution management service is very important and necessary.

The virtual shopping mall is partitioned into several separate shops, which were defined as different regions in the RTI routing space. Object instances are registered with associate update regions. Then RTI only dispatches update messages to the objects within the same region.

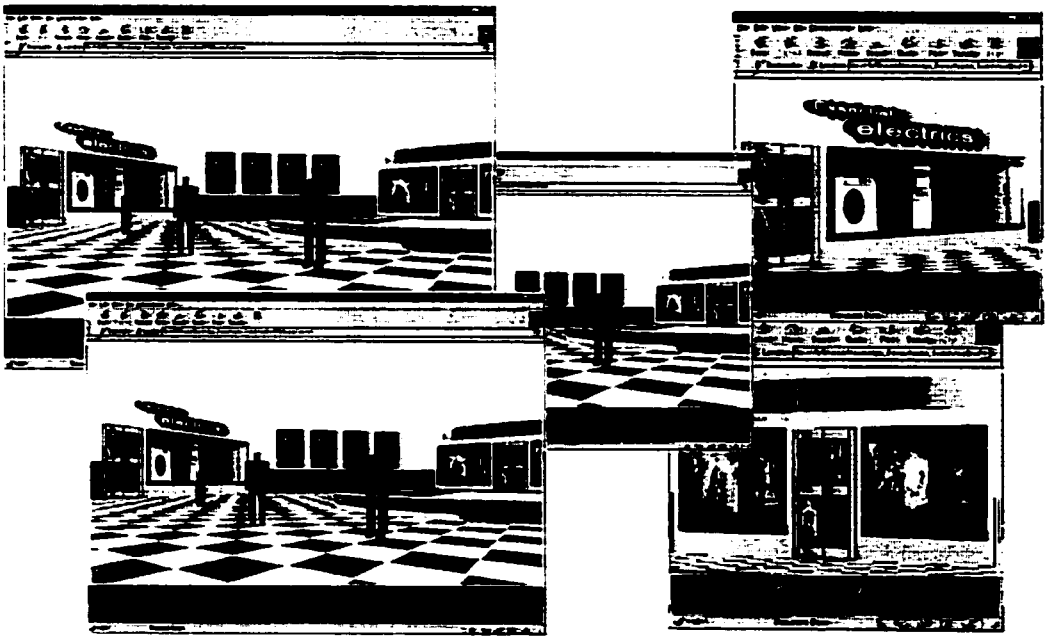


Figure 2.13 DDM in Virtual Shopping Mall

2.6. Time Management (TM)

Time Management is concerned with the mechanisms for controlling the advancement of each federate along the federation time axis [13]. It should be emphasized that the purpose of TM is not to provide a real-world time in the federation. Instead, TM provides a synchronization mechanism among federates though time-stamped-ordered (TSO) events and time advances rules.

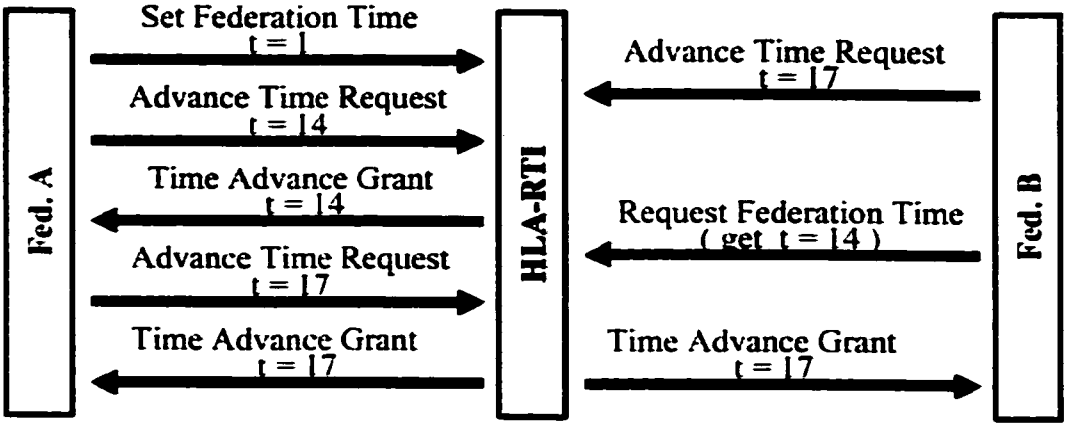


Figure 2.14 Time Management

TM is usually used in the distributed simulations that involve heavy duty processing but need real time results or synchronization points. Each federate consumes different simulation time that is far longer than the actual time in the real system. But with the TM, federates can advance time following the actual time in the real system, ignoring the actual simulation time.

Different federates may consume different simulation time. The federates that finish their processing earlier may ask for advancing federation time earlier, but the federation may not grant the time advance and the federates may have to wait. The federation only grants the smallest time request, so earlier finishing federates will wait until the slowest federate finished and requests further time advance (refer to Figure 2.14).

In the virtual shopping mall application, all federates are fast enough for real-time response, so no further synchronization mechanism provided by TM is required. But as an example, we can consider a TM based real-world clock federate in the virtual mall.

The responsibility of the clock federate is to advance the federation time following the real-world time: keeping the current real-world time updated by GPS or other devices; advance the federation time following the real-world time. Any other federates can

depend on federation time to update their status or actions respectively, for example, the different light/music/advertisements during different time periods of a day.

2.7. Matches between HLA and DVE Requirements

Reviewing the requirements of DVE, we can see how the HLA matches them.

- The shared sense of space is represented by the federation concept of HLA. The Federation of HLA can provide an illusive space where any simulator can easily join in and resign from.
- The shared sense of presence is supported by the federate concept of HLA with assistance of the Object Management service. Each avatar is implemented as a federate with status and behaviors represented by the attributes of HLA objects.
- The shared sense of time – Our virtual shopping mall application verified that HLA-RTI has acceptable performance to support DVE real time interaction over a LAN environment. Furthermore, the Data Distribution Management service of HLA effectively limits unnecessary events and reduces the response time of a federate. On the other hand, HLA has some limitations for supporting large scale real time distributed simulations over the Internet (refer to 2.8.2).
- The way to communicate is supported by the Object Management service of HLA. Objects may be used to represent status updates and interactions may be used to represent messages. Not only simple text based messages can be exchanged, but also more natural audio communication can be supported.
- The way to share is supported in HLA through the Objects Management service and the Ownership Management service. The user interacts with the virtual environment itself and with other users by updating object attributes, sending interactions and

exchanging ownership of object attributes.

- Awareness/Visibility Control is handled by the Data Distribution Management service. DDM divides the DVE space into smaller regions; the data generated in one region will not be sent to other regions. As a result, awareness and visibility are well controlled.
- Dynamic simulation entity management is supported by the Federation Management service. The Simulation entity is represented as a federate. Any federate can join in and resign from a federation easily at any time by implementing simple steps. One limitation is that the federate must be predefined in FOM, refer to 2.8.1.
- Simulation backup and restore are handled by the Federation Management(FM) and Time Management(TM) services. FM provides functions for coordinating federation-wide saves and restores. One federate may initiate a federation save or restore and RTI will notify this request to all other federates in the federation. Then all federates save or restore their status. The TM service is used when the optional function parameter – federation time is provided when the first federate initiates the request. The given federation time will be communicated to all federates and all federates will save or restore their status at specific federation time points.

2.8. HLA Limitations for Shared Space Management in DVE

As a new DIS standard that was inspired from all old ones, HLA has powerful features to support a very wide range of DIS applications, but there are still two limitations for shared space management.

2.8.1. Federation Object Model (FOM) Issues

HLA demands FOM for each federation. FOM is a profile for concise and rigorous description of the data exchange among federates within a federation, including an enumeration of all object and interaction classes pertinent to the federation. FOM definition is always required to build a federation. All exchangeable data must be declared in FOM. All federates in a federation can only utilize the attributes and parameters defined statically by the FOM.

FOM causes some limitation for DVE. DVE has potential requirements of "Openess" which means that a new featured federate should be able to join in after the federation has started. In the virtual shopping mall example, new salesmen and new products may be added to the virtual mall anytime. The new stuffs may have new behaviors or features not defined in the old FOM. To activate this new behaviors and features, the operator has to stop the virtual mall, update the FOM and then startup the virtual mall again. This is inconvenient for a small virtual mall since periodic maintenance becomes necessary; it is also unacceptable for a large scale virtual mall which may be over several time-zones and may have no time for maintenance.

Some researches address this issue [25], which is not discussed in this thesis.

2.8.2. Stream Transmission and Control

A stream is a continuous sequence of data transferred from one node to others, such as audio and video in a computer network.

Since audio and video are a most natural way for humans to perceive the environment and exchange information, it is not acceptable that a DVE does not support audio and video.

The HLA standard does not support native stream transmission and control functions. A HLA compliant stream transmission and control scheme is proposed in part II of this thesis.

2.8.3. Quality of Service Issues

While there are many soft real-time applications in simulation, there are also numerous applications where a hard real-time feature is an absolute requirement. Research shows that in a flying simulator not only *delays* but also *badly timed cues* can adversely affect pilot perception, control behavior, and performance [26]. Hard real-time response is not an option or a luxury in this kind of system, but rather an absolute requirement. For example, in a virtual surgery operation that is operated by several surgeons around the world, real-time responses from all the end devices and network are critical.

The real-time requirement is not difficult to be satisfied at end nodes by using a real-time operating system (RTOS). More importantly, the network communication quality is also critical for real-time response. The Internet Engineering Task Force (IETF) is working on QoS solutions for IP networks.

HLA has not considered QoS issues. If HLA wants to support real-time DIS, it must extend itself to use real-time features of RTOS and QoS features of the underlying network.

An HLA real-time extension and related real-time RTI is proposed in part III of this thesis to address this problem.

Part II

HLA Stream Transmission and Control

Chapter 3
HLA Stream Transmission

3.1. Stream Requirements of DVE

As we introduced in 1.1, A *Distributed Virtual Environment* (DVE) is a software system in which multiple users interact with each other in real-time, even though those users may be physically located around the world [23]. Typically, each user accesses his/her own computer workstation or console, using it to provide a user interface to the content of a virtual environment. These environments usually aim to provide users with a sense of realism by incorporating realistic 3-D graphics and stereo sound to create an immersive experience [23].

DVE are being used for education and training, for engineering and design, for commence and entertainment. Indeed, the range of applications is growing rapidly as developers find new ways to leverage the technology to save costs, reduce time-to-market, and enhance human safety [23].

A stream is a continuous sequence of data transferred from one node to others. Audio and video are a most natural way for humans to perceive the environment and exchange information. In a DVE system, audio and video are used for the following purpose:

- Immersive experience objective of virtual environment (requires background sound playback);
- Product or Information introduction (requires audio/video retrieve function);
- Communication between people (requires interactive audio (even video) phone function).

Based on these requirements, DVE systems demand both a stream retrieve mechanism and a stream interactive communication mechanism.

HLA-RTI is becoming the standard of DVE. It only defines the APIs for information exchange. It does not provide APIs for stream transmission directly. Stream transmission has special requirements compared to a general variable value exchange, such as stable data rate, elimination of delay jitter, etc. Developers should work out a stream transmission mechanism by themselves based on HLA-RTI providing APIs.

In the following several chapters, we are going to illustrate our HLA compliant stream transmission and control scheme.

3.2. Stream Transmission Architecture

Stream transmission involves sender and receiver. As a requirement of HLA-RTI, both sender and receiver are federates and they must follow the HLA rules in order to be RTI compliant:

> *HLA Rule 3: During a federation execution, all exchange of FOM data among federates shall occur via the RTI* [12].

This rule demands to use RTI provided API [13] to implement stream transmission.

The most natural APIs that could be used for stream transmission is the pair of *sendInteraction()* and *receiveInteraction()*. "Each interaction is constructed, sent and forgotten. Interaction recipients receive, decode and apply the interaction." [15]. It is just the way that a stream sender/receiver does.

sendInteraction() and *receiveInteraction()* belong to the object management category of RTI service. Referring to [15], *sendInteraction()* is a function method of class *RTIambassador* and *receiveInteraction()* is an abstract function method declared in abstract class *FederateAmbassador*. The sender federate should call *sendInteraction()* to send data out to network and the receiver federate should call *receiveInteraction()* to receive data from network. As mentioned above, the federate must implement the functionality declared in *FederateAmbassador*. Thus a receiver federate should implement *receiveInteraction()* and add operations of receiving data in this function.

Thus the proposed architecture for stream transmission is shown in Figure 3.1.

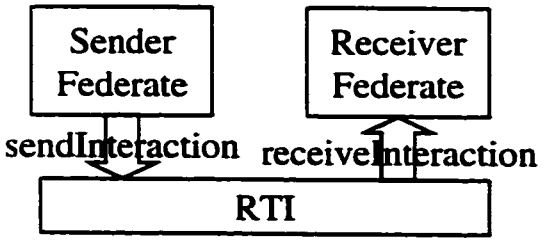


Figure 3.1 Architecture for Stream Transmission

RTI supports both "reliable" mode and "best effort" mode communication. Data loss and error are not very critical for audio/video transmission, so it is not necessary to use the "reliable" mode. In order to improve throughput and reduce latency, we adopt the "best effort" mode in this solution.

3.3. Schedule Control and System Interval Timer

Packet audio systems demand low end-to-end latency [27]. The requirement of low latency means that a packet audio system must launch small packets frequently, rather than big packets less often [27]. Packet video systems also have this requirement. As a result, the stream transmission scheme requires a schedule to launch small packets frequently.

On the other hand, federates may have many simulation tasks besides stream transmission. So it is necessary to schedule a stream transmission task in order to share system resource with other federate tasks.

The schedule requirement can be realized depending on the system interval timer provided by most operating systems. The interval timer services allow an application to schedule periodic timer events — that is, the application can request and receive timer messages at application-specified intervals. The interval timer services allow applications to schedule timer events with the greatest resolution (or accuracy) possible for the hardware platform. These timer services are useful for applications that demand high-resolution timing [28].

It is easy to use the system interval timer by following these steps [28]:

1. Obtaining and setting timer resolution
2. Starting timer events. An initialization function is used to start timer events. One of the function's parameters is the address of a callback function that is called when the timer event takes place. An Interval timer can be configured to generate periodic timer event every time a specified number of millisecond elapse.

3. Implementing application specific tasks in callback function. The Developer should put the real-time operation in the callback function. Timer calls the function automatically when the timer event takes place.
4. Canceling a timer event when it is not required.

In the Windows operating system, the system interval timer is named as "Windows Multimedia Timer" [28].

In the Unix operating systems, the system interval timer is named as "POSIX.1b Interval Timer" [29].

Depending on the system interval timer, we can arrange the schedule of stream package transmission. The sender federate divides stream data into packages and sends them out one by one periodically by calling *sendInteraction()* in the callback function. The receiver federate calls *receiveInteraction()* in the callback function in order to receive the incoming packages periodically.

At the same time, other real-time tasks can use the interval timer in the same way so that an overall schedule of the federate process can be maintained in the same mechanism.

3.4. Latency and Jitter Control

A real-time stream transmission through a network should consider latency and delay jitter issues.

In a shared packet network, routers operate on a first-in first-out basis and statistically multiplex traffic from different sources [30]. The impact of this behavior on real-time traffic is to introduce jitter to the inter-packet timing relationship [30]. RTI Version 1.3 is

working over a TCP/UDP/IP network, which is also a shared packet network. Thus we must consider jitter elimination issues for stream transmission over RTI.

For an audio stream, jitter must be removed, since it renders the speech unintelligible [30]. A voice reconstruction buffer is used to artificially add latency to the audio stream to smooth out the jitter [30]. The longer the reconstruction buffer is, the larger the jitter that can be eliminated through it.

On the other hand, the longer the reconstruction buffer is, the longer latency is added. Since the minimum end-to-end latency is also an important objective of stream transmission, the length of the reconstruction buffer should be set carefully.

For a video stream, the effect of jitter on video is jerky rendering of frames, a degradation that can be tolerated by users [30]. So the video stream does not need a reconstruction buffer.

3.5. Implementation of Audio Stream Retrieve

We made an experiment that uses the upper mentioned method to retrieve an audio stream from one PC to multiple PCs within the same LAN.

In the experiment, there are one sender federate and multiple receiver federates joining in the same federation. After opening a local wave file, the sender federate reads and sends out audio data to the network. All receiver federates receive and play back the data.

3.5.1. Sender Federate Architecture

Figure 3.2 gives the architecture of sender federate.

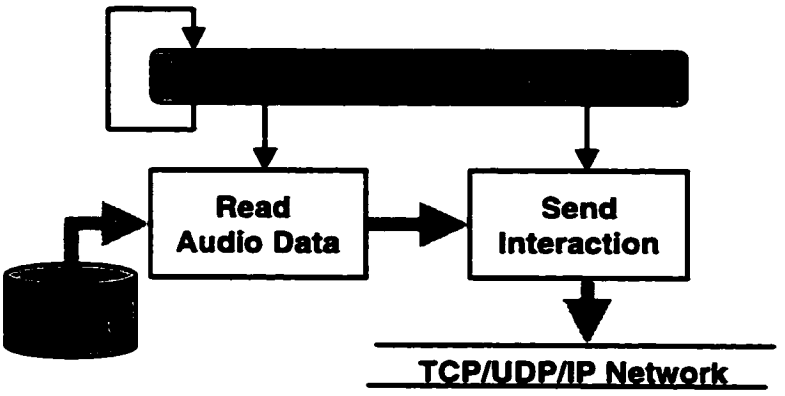


Figure 3.2 Sender Federate Architecture

A system interval timer invokes a callback function periodically, which contains the audio stream processing method. This method implements two operations:

1. Reads audio stream data from a local harddisk;
2. Calls sendInteraction() to send out an audio package.

3.5.2. Sender Federate Scenario

1. Create and join Federation;
2. Publish and Subscribe Interaction and Interaction Parameters;
3. Open wave file;
4. Read the audio stream format from the wave file, including duration, average data rate, data size, silence data, codec type, etc;
5. Send a FORMAT message with audio stream format parameters to receiver through RTI API sendInteraction();
6. Setup interval timer, including interval and callback function. Then activate the interval timer. Operating system begins to invoke the callback function periodically.

7. The callback function implements a sending cycle with two operations:

   Read audio data from wave file;

   Send DATA message with audio data package to receiver through sendInteraction();

8. When reaching the end of wave file, the callback function sends a *TRANSFER_END* message to the receiver through *sendInteraction()*;

3.5.3. Receiver Federate Architecture

Figure 3.3 gives the architecture of the receiver federate.

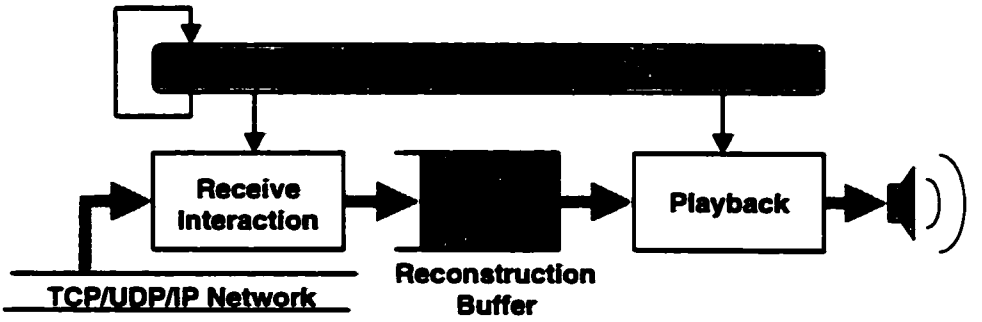


Figure 3.3 Receiver Federate Architecture

As mentioned above, the reconstruction buffer is used to eliminate delay jitter. A system interval timer invokes a callback function periodically, which contains the audio stream processing method with two operations:

1. Receives audio stream data from network by calling receiveInteraction() and puts the data into the reconstruction buffer;
2. Sends the audio data in the receive buffer to the audio device to playback.

In RTI1.3, *receiveInteraction()* is a callback function invoked by *tick()*, so the audio processing method in the receiver federate calls *tick()* to invoke *receiveInteraction()* rather than invoking it directly.

The reconstruction buffer is a FIFO buffer which is realized using a ring architecture (refer to Figure 3.3). There are two pointers to record the current read position and the current write position. The audio playback device reads data from the buffer and advances the current read position pointer. At the same time, the *receiveInteraction()* function writes data into the buffer and advances the current write position pointer.

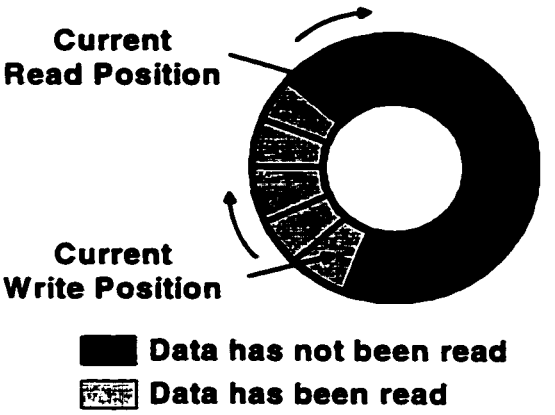


Figure 3.4 Reconstruction Buffer

3.5.4. Receiver Federate Scenario

1. Create and join Federation;
2. Publish and Subscribe;
3. Wait for incoming message by calling tick().
4. Receive FORMAT message.
   - Initialize reconstruction buffer depending on the received audio format;
   - Initialize audio playback device;
   - Setup interval timer, including interval and callback function. Then activate the interval timer to make it go into effect at once. Operating system begins to invoke the callback function periodically.

The callback function implements a receiving cycle with two operations:

1. Receive message from network by invoking *receiveInteraction()* through invoking *tick()*;
   - If receiving *DATA* message, the audio data package in the message is copied to reconstruction buffer.
   - If receiving *TRANSFER_END* message, go to step6).
2. Read audio data from reconstruction buffer and send the data to audio device for playback.

End of transmission:

- Release interval timer;
- Release audio playback device;

Go to step 3).

3.6. Performance analysis

3.6.1. Experiment Environment

Operating System:	Windows NT4.0
LAN:	100BaseT
PC Style:	PentiumII-400
RTI:	1.3v5
Audio sample:	CD Quality Music, 44.1khz/16bits/PCM/Stereo (172KB/S)

3.6.2. Quality of Audio

3.6.2.1.Average Speed

For a specific quality audio stream, the average speed is identified. To maintain this average speed, a specific equation should be satisfied.

Assuming that the average data rate is x with unit bytes/sec; the size of a data package sent by *sendInteraction()* is y with unit bytes; the time interval of calling *sendInteraction()* is z with unit millisecond, we can get following equation:

$$x = y * (1000 / z)$$

We can satisfy the average transmission speed by adjusting two parameters:

- Audio data package size
- Time interval of calling sendInteraction().

3.6.2.2.Jitter

We know that the average transmission speed is not enough to assure the quality of the audio stream. Just as mentioned before, the delay jitter renders the speech unintelligible so that it must be eliminated using a reconstruction buffer. On the other hand, the reconstruction buffer adds extra latency to the audio stream, which we do not want. Finally, what we can do is to find the best tradeoff between two targets: choose the minimum reconstruction buffer to eliminate the delay jitter.

We made some transmission experiments to identify the relationship between audio quality and length of reconstruction buffer and timer interval. Table 3.1 gives some typical data:

Timer interval (msec)	Reconstruction buffer length (msec)	Effect
100	300	Good quality; No distortion; No noise
150	300	Poor quality; Has distortion
50	200	Good quality; No distortion; Has inconspicuous noise between times

40	200	Good quality; No distortion; No noise
70	210	Has distortion; Has noise
60	180	Poor quality; Has distortion; Has background noise
45	180	No distortion; Has background noise;
10	150	No distortion; Has background noise
20	100	Has distortion; Has background noise; But contents are clear

Notes: Test audio is CD quality with average rate 172kbytes/sec (44.1khz/PCM16bit/Stereo)

Table 3.1 Quality and Reconstruction Buffer

After many experiments, we find the following rules:

- Quality of audio is sensitive to the length of reconstruction buffer.
- For CD quality audio, 200~300 milliseconds reconstruction buffer is enough to get high quality undistorted audio.
- For Voice communication, 100 milliseconds reconstruction buffer is enough to get understandable voice.
- Quality of audio is not sensitive to the timer interval. But the time interval should satisfy the condition that the length of reconstruction buffer is larger than 3 times the timer interval. A shorter reconstruction buffer requires a larger multiple. For example, a 300msec reconstruction buffer requires a 100msec timer interval, while a 200msec reconstruction buffer requires a 40msec timer interval.
- It is not necessary that the sender timer interval is equal to the receiver timer interval, but the same interval can get the best quality.
- Two fields of the RTI configuration in the RTI.rid file are related to audio quality:

receive_buffer_size and send_buffer_size of udpSocketManager. The value of these two fields should be adjusted based on the timer interval. The default value (128kbytes) is suitable to 50msec interval. For 100msec timer interval, these fields should be upgraded to 256kbytes in order to assure audio quality, otherwise, noise is generated.

3.6.2.3.Latency

From Figure 3.1and Figure 3.2, it is easy to identify the factors related to latency:

- Interval timer latency at sender;
- Processing latency at sender;
- Variable latency experienced by audio packets traversing the network. – transmission latency;
- Interval timer latency at the receiver;
- Processing latency at the receiver;
- Reconstruction buffer latency at the receiver.

The total latency is not just the value of adding up all latency of each part, since there is a parallel processing relationship among them.

1. Latency from reconstruction buffer

Assume latency of the reconstruction buffer is D_{buffer} . D_{buffer} should be 100~300msec based on the quality requirement of the audio. (Refer to Table 3.1).

2. Latency of processing

Assume the processing latency is $D_{process}$. To measure the processing latency, we can mark the time at both entry point and exit point of the interval timer callback functions in the sender federate and the receiver federate. The processing latency is just the difference between the two time values.

Table 3.2 is from an experiment of transferring CD quality audio (44.1khz-16bits-PCM-Stereo, 172kbytes/sec) between a sender federate and a receiver federate. The first column represents the time interval of calling the audio processing method in both sender and receiver federates. The second column represents the size of the audio data packet. (We can see that the values in column one and column two satisfy the equation $x = y * (1000 / z)$). The third column represents the time consumed in the callback function of the sender federate, including operations listed in step7 of the sender scenario. The fourth column represents the time consumed in the callback function of the receiver federate, including receiving data from the network to the reconstruction buffer and filling data to the audio playback device.

Timer Interval (ms)	Data Package Size (bytes)	Sender (ms)	Receiver (ms)
100	17640	6	4
50	8820	4	2
25	4410	2~3	1~2
10	1764	<1	<1

Table 3.2 CPU Time Consuming

The result shows that the audio processing function in this approach does not consume much CPU time.

3. RTI transmission latency

Assume the transmission latency as D_{rn}. We can measure its average value using a rollback method. The sender marks the time when it sends data out; the receiver returns back to the sender what it received; when the sender receives returned data, it marks the time again. The latency just equals half of the difference between two marked time values. Using this method, we got RTI interaction transmission latency (refer to Table 3.3).

Data Packet Size (bytes)	Average Latency (ms)
17640	14
8820	12
4410	10
1764	5

Table 3.3 Interaction Transmission Latency

4. Timer interval latency

Assume both sender federate and receiver federate have the same timer interval as D_{timer}. The developer can choose D_{timer} depending on the latency requirement of the system. The larger D_{timer} value, the less resource allocated by the audio processing method, but the longer latency has to be tolerated.

5. Total end-to-end latency D_{total}

Assume the timer intervals of sender and receiver federates are equal. In a general situation, the end-to-end latency should be:

$$D_{total} = D_{process} + D_{timer} + D_{rn} + D_{buffer}$$

There is only one D_{timer} value and one $D_{process}$ value is calculated since the interval timers of both sender and receiver have synchronization relationship. The audio processing in the sender federate and receive federate is parallel processing.

For example, assume we use a 200msec-length reconstruction buffer and 40msec interval timer. Since 176400/(1000/40) = 706 bytes/package, which is less than 1764bytes, we can use D_{rn} = 5msec and $D_{process}$ = 1msec (refer to Table 3.2 and Table 3.3). We get:

$$D_{total} = 1 + 40 + 5 + 200 = 246\text{msec}$$

In conclusion, this stream transmission approach over RTI can support CD quality audio transmission with less than 250msec latency in a LAN environment.

3.6.3. Multicast Effect

We run a sender in one PC and multiple receivers in different PCs. The latency difference among all receivers is imperceptible.

Chapter 4
HLA Stream Control

As an extension to the stream transmission solution, this chapter illustrates a HLA-compliant stream control mechanism through an audio/video playback application, which could be used in a DVE shopping mall to play background audio or do audio and video combined advertisements.

The system can implement two basic controls: play and stop. Other control features can be added easily using the same mechanism.

4.1. Feature Specification

The system contains one server and multiple clients, which are connected through HLA-RTI over an IP network (refer to Figure 4.1). The system runs in a LAN with Windows NT 4.0 platforms.

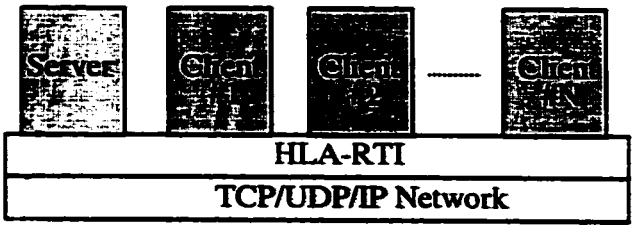


Figure 4.1 Audio/Video Retrieve System - Architecture

The server plays back the audio/video program. The clients can join the federation individually at any time. After it joins, a client becomes active. It can receive audio/video packets from the server and play them back.

The server playback can pause and restore as required. When the server pauses, all active clients pause too until the server restores.

A client can also pause and restore individually. But its pause and restore operation does not affect current operation status of the server and the other clients. When a client pauses, the server continues to multicast its program to other clients. When it restores, the client plays from the current position, as the other clients. The missed part is not recovered.

4.2. Interface Mode

In HLA-RTI, there are two mechanisms for sharing information between federates: objects and interactions. *Object classes* are comprised of attributes. Object classes describe *types* of things that can *persist. Interaction classes* are comprised of parameters. Interaction classes describe *types* of events [15].

The primary difference between objects and interactions is *persistence*. Objects persist, interactions do not.

In this system, an object is good at presenting the long lived variables such as operation status and audio format, while an interaction is still used as presenting the stream data transferred between server and client.

4.3. Objects

There are two objects used for information exchange between the clients and the server – "Operation" and "AudioFormat".

The "Operation" object indicates the current operation status of the server. Since it is visible to the clients, the clients can coordinate their operations with the server according to the value of the attribute in this object.

The "Operation" object just has one attribute – "Type" which is an enumerative variable. Its value can be "PLAY" or "STOP".

The "AudioFormat" object indicates the format of the audio playing at the server. We use the object to keep this information so that the clients can get it at any time. In this way, the clients can join the federation and receive the audio/video stream correctly even if the server has already begun playback.

The "AudioFormat" object has two attributes – "Format" and "FormatLength". The "Format" is a data block that contains necessary audio parameters such as sample rate, number of channels, bits/sample, etc. The "FormatLength" identifies the length of the "Format" data block. Since the length maybe different for different audio streams, this attribute is useful for RTI to retrieve the attribute "Format".

Instead of using "VideoFormat" object, we embed video format information into each video frame. Video retrieval is not as complex as audio retrieval, which demands device initialization and buffer preparation. Audio format is required by the client to do initialization before the real audio data comes. So we transfer audio data and audio

format separately. Video retrieval is just a receiving and displaying process, and no previous operation is required. So we transfer video data and video format together.

4.4. Interactions

There are two interactions used for stream data transmission from the server to the clients – "AudioPacket" and "VideoPacket".

The "AudioPacket" interaction is used to transfer audio data, which contains three parameters:

- Data – Presents audio contents,
- Length – Presents the length of data,
- SerialNumber – Used to maintain the sequence of audio packets.

The "VideoPacket" interaction is used to transfer video data, which contains two parameters:

- Data – Presents video contents,
- Length – Presents the length of data,
- SerialNumber – Used to maintain the sequence of video packets.

4.5. Reliable or Best-effort

RTI supports both "reliable" and "best effort" communication modes for object attribute update and interaction convey.

Since the operation status and audio format are critical for the correctness of the clients' operation, we adopt the "reliable" mode for all object attributes of the objects "Operation" and "AudioFormat".

Since the data loss and error are not very critical for audio/video stream data, it is not necessary to use the "reliable" mode. In order to improve throughput and reduce latency, we adopt the "best effort" mode for the interactions "AudioPacket" and "VideoPacket".

4.6. Server Architecture

The Figure 4.2 illustrates the architecture of the server.

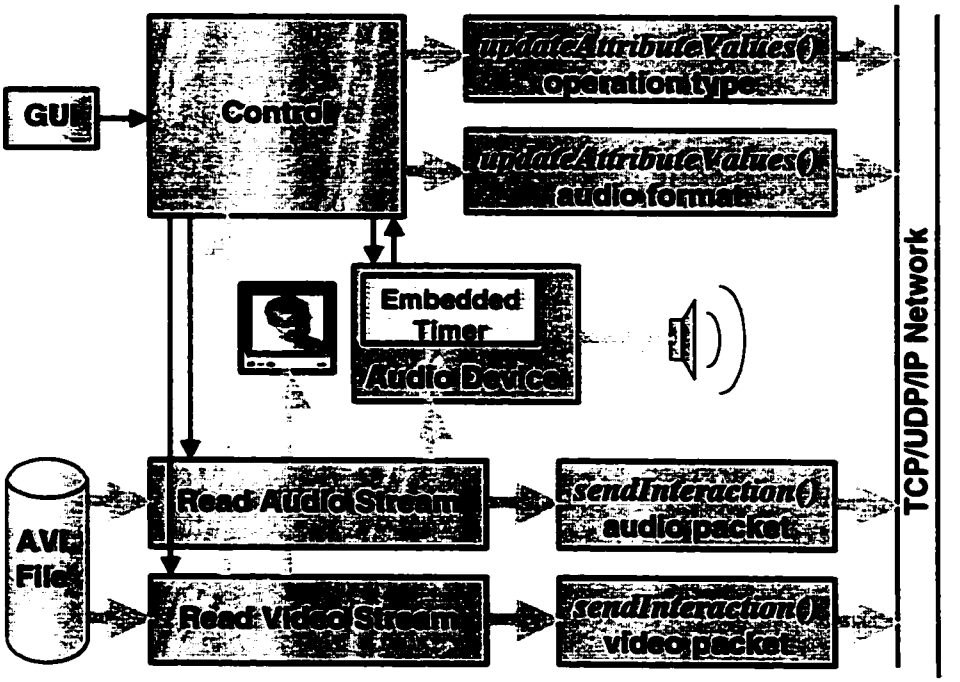


Figure 4.2 Server Architecture

The server reads audio and video data from a local AVI file and plays it back locally. At the same time, the server sends out the audio and video data to the clients through the HLA-RTI API *sendInteraction()* when there is at least one active client.

The schedule control method used in this system is a little different from the solution mentioned in 3.3. We use the *audio device embedded timer* for this system [31].

This system is designed for a DVE shopping mall. The server is also required to play back the audio/video program, the as the clients, so that an operator can know what is playing to the customers in the mall. Since the server has to use a local audio device, it is more reasonable and convenient to use the timer of the audio device rather than the system interval timer.

When the operator identifies an AVI file at a local harddisk and chooses the PLAY operation, the server reads the audio/video format from the file and initializes the audio device and video device. Then, the server reads audio data from the file and puts it into the audio buffer to play back. In the main message loop of the server code, the current playback position is calculated based on both the time got from the audio device embedded timer and the current audio stream reading position. Then, the video frame related to the position is read from the file and displayed on the screen.

Each time when the server changes its current operation, it updates the RTI object "Operation". Each time when the server begins to play a new AVI file, it updates the RTI object "AudioFormat" at first. These updates can be transferred to the clients through HLA-RTI API *updateAttributeValues()*, so the clients can cooperate with the server.

Each time when the server reads audio data and video data, it also sends the data out to the active clients through RTI *sendInteraction()*.

The Figure 4.3 shows the key classes of the server.

RTISTREAMRTIAmbassador

The class inherits from the RTI interface class *RTI::RTIambassador* which integrated all required RTI APIs. It undertakes the following tasks:

- Send operation command to client,
- Send audio format to client,
- Send video packet to client,
- Send audio packet to client.

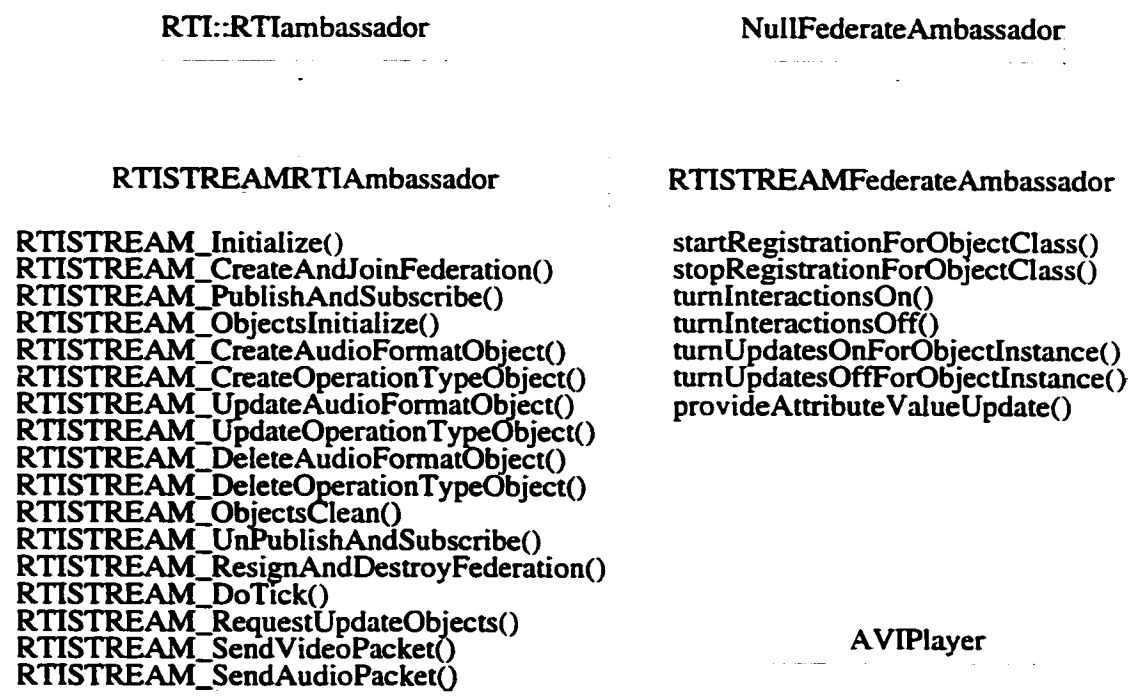


Figure 4.3 Server Class Diagram

RTISTREAMFederateAmbassador

The class implements the abstract class *FederateAmbassador* of the RTI interface. RTI can callback the server through the methods implemented in this class for the following tasks:

- Receive and handle operation query from client,
- Receive and handle audio format query from client.

AVIPlayer

The class implements audio/video operation including:

- Analyze AVI file,
- Initialize audio device,
- Manage audio buffer and timer,
- Read audio data from AVI file,
- Play back audio data,
- Read video frame data from AVI file,
- Display video frame on the screen.

4.7. Client Architecture

The Figure 4.4 illustrates the architecture of the client.

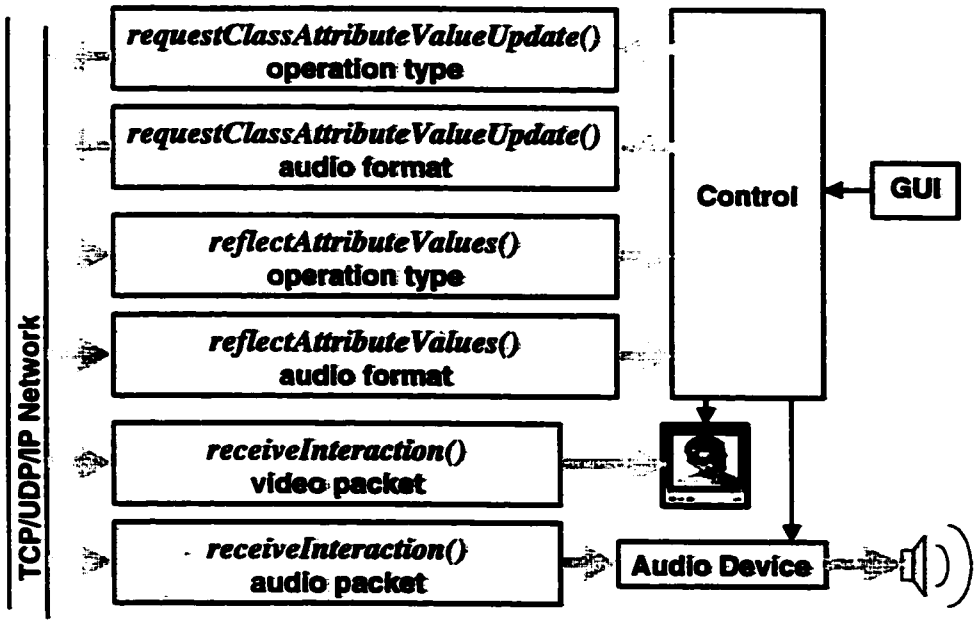


Figure 4.4 Client Architecture

As a simple solution, the client does not control the schedule locally. In the main message loop of the client code, the *tick()* method of the RTI interface is called. The

tick() invokes *reflectAttributeValues()* when the server has made a new update to the attribute of the subscribed object, and *receiveInteraction()* when there is a coming packet.

When the client gets the updated value of the object "AudioFormat", it initializes the audio device using the new format. When the client gets the updated value of the object "Operation", it starts or stops the audio device accordingly.

In PLAY status, the received audio packet is put into the audio buffer and played back. The received video packet is displayed directly.

Figure 4.5 shows the key classes of the client.

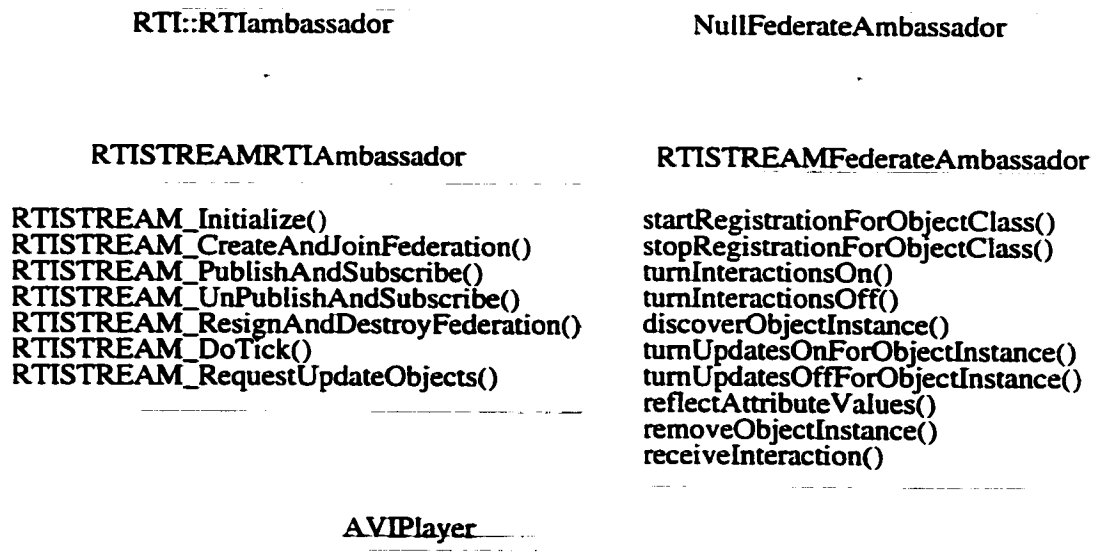


Figure 4.5 Client Class Diagram

RTISTREAMRTIAmbassador

The class inherits from the RTI interface class *RTI::RTIambassador* that integrates all required RTI APIs. It undertakes the following tasks:

- Send operation query to the server,

- Send audio format query to the server.

RTISTREAMFederateAmbassador

The class implements the abstract class *FederateAmbassador* of the RTI interface. RTI can callback the client through the methods implemented in this class for the following tasks:

- Receive operation command,
- Receive audio format,
- Receive video packet and display,
- Receive audio packet.

AVIPlayer

The class implements audio/video operation including:

- Initialize audio device,
- Manage audio buffer,
- Play back audio,
- Display video frame.

4.8. Scenario

4.8.1. Server Initialization

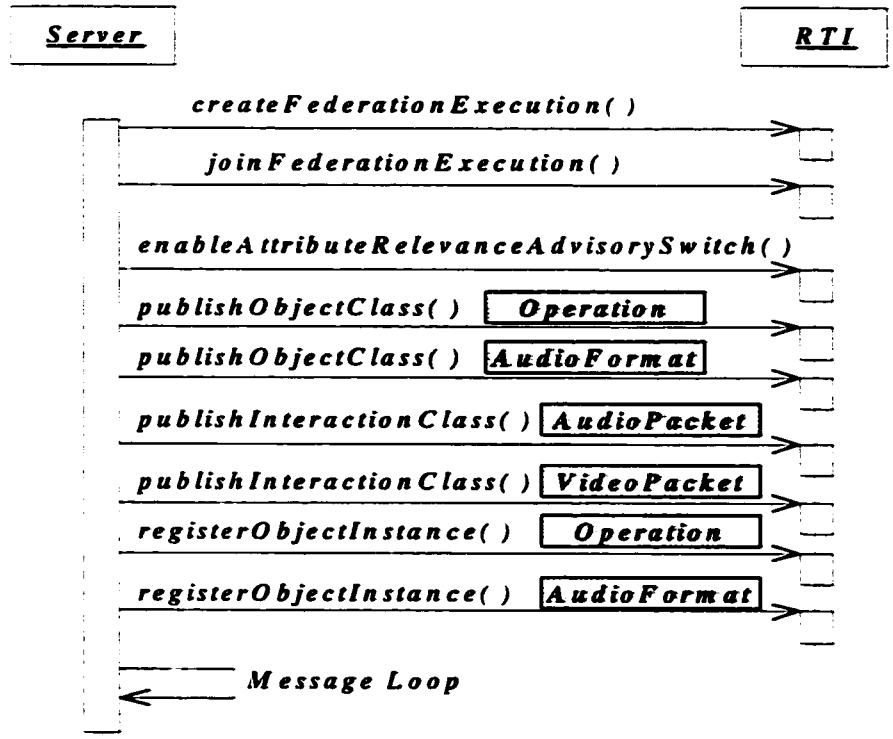


Figure 4.6 Server Initialization

When the server application is invoked, the application implements the steps listed in the Figure 4.6 to initialize. An "Operation" object and an "AudioFormat" object are created during the initialization by calling RTI API *registerObjectInstance()*.

4.8.2. Client Initialization

Figure 4.7 lists the initialization scenario of the client when the server has already started. (The clients can also startup before the server, but all the steps of the RTI callback functions in the scenario are not implemented during the initialization.).

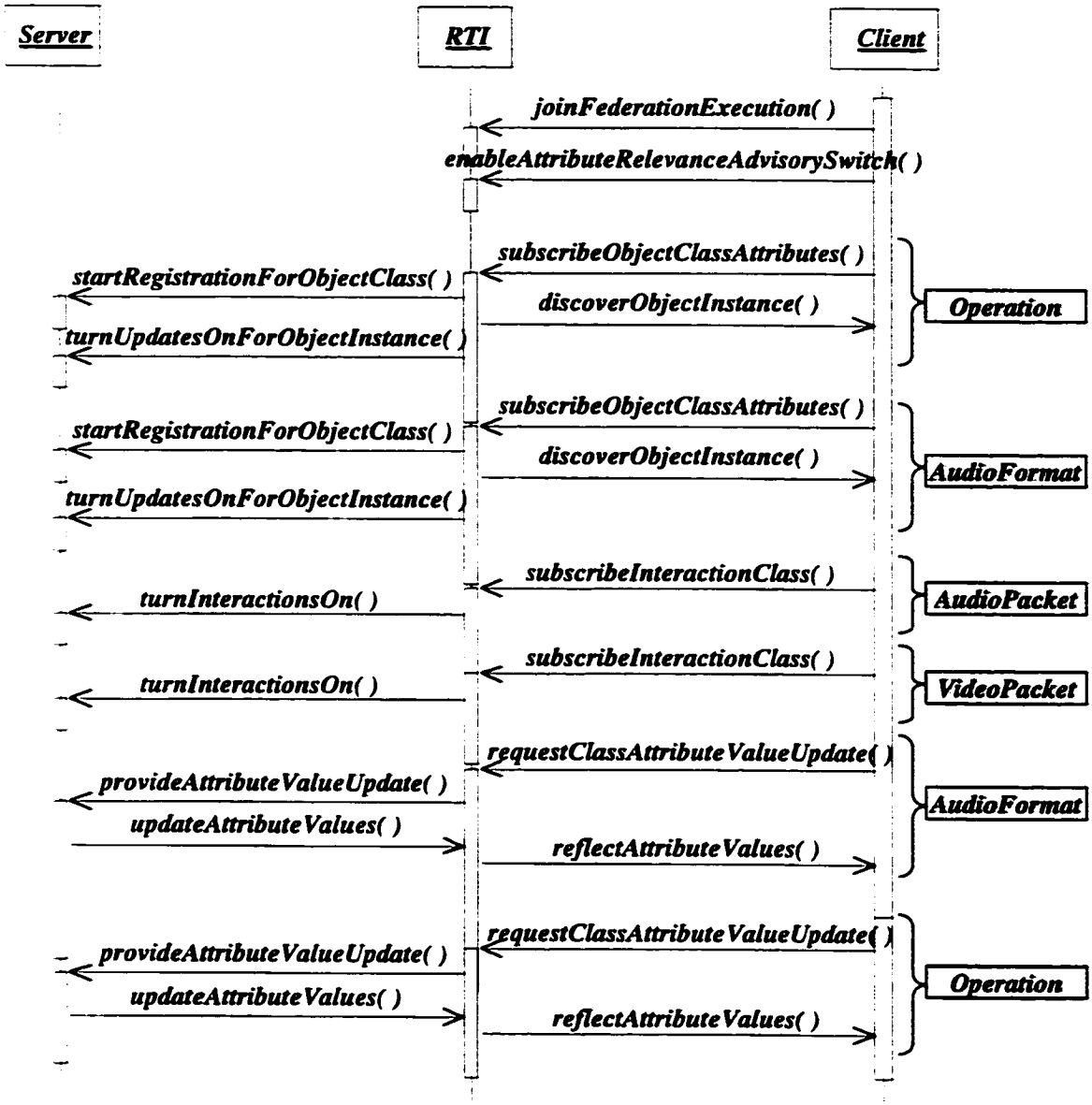


Figure 4.7 Client Initialization

The client subscribes the "Operation" object to RTI by calling *subscribeObjectClassAttributes()*. RTI invokes *startRegistrationForObjectClass()* of the server to notify it that a client startups and is interested in the value of object "Operation".

Since the server has already implemented the function *registerObjectInstance()* during its initialization, the client's callback function *discoverObjectInstance()* and the server's callback function *turnUpdatesOnForObjectInstance()* are invoked by RTI automatically after the client subscribes the object. The *discoverObjectInstance()* notifies the client there is a server active. The *turnUpdatesOnForObjectInstance()* notifies the server: from now on, it should call *updateAttributeValues()* to make notification when it updates the value of the object "Operation".

The client also subscribes to the "AudioFormat" object and implements the same process.

Then the client subscribes to the "AudioPacket" interaction to RTI by calling *subscribeInteractionClass()*. RTI invokes *turnInteractionsOn()* of the server. The *turnInteractionOn()* function notifies the server: from now on, it should call *sendInteraction()* to send the audio data to the client when it reads audio data from an AVI file.

The client also subscribes to the "VideoPacket" interaction to RTI and implements the same process.

At this moment, the client just joins the federation. The server does not tell it the current values of the objects until the next update. The client needs to know the values at once to begin playback, so it should ask for the update on it's own initiative by calling *requestClassAttributeValueUpdate()*. Then RTI invokes the server's *provideAttributeValueUpdate()* to notify the server. The server updates the object by calling *updateAttributeValues()*, which finally invokes the client's *reflectAttributeValues()* to make the client get the current value of the object.

The required update is implemented two times, so the client gets the current value of the "AudioFormat" and the "Operation".

If the current value of the attribute "Type" of the object "Operation" is "PLAY", the client should initialize the audio device based on the information in the "AudioFormat" object, then receive and play back the audio/video packets. This is the situation that the client joins the federation and receives the stream from a middle point.

4.8.3. Playback

When the operator chooses an AVI file and plays it at the server side, the server plays the AVI file locally and implements the steps listed in the Figure 4.8 to make the active clients play it at the same time.

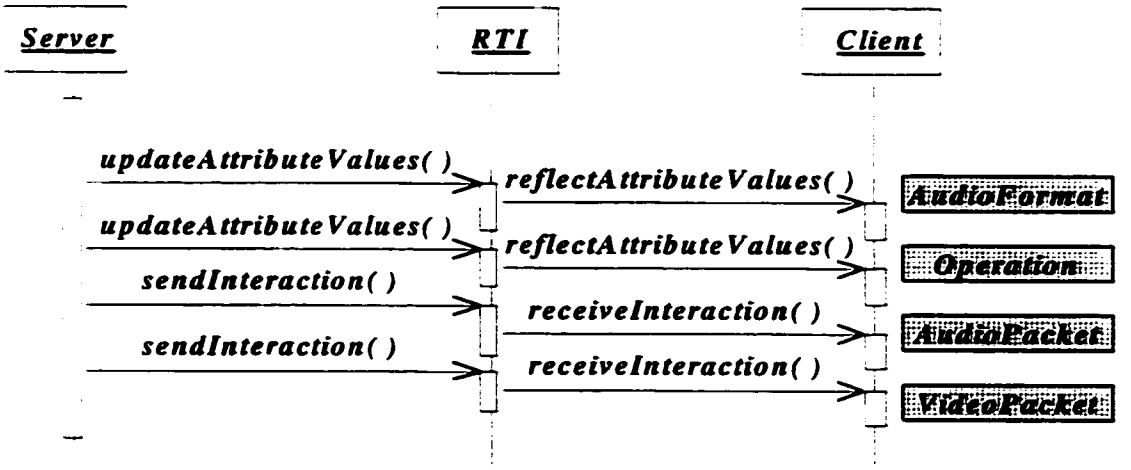


Figure 4.8 Stream Playback

Before transferring audio/video packets, the server calls the *updateAttributeValues()* for the "AudioFormat" which invokes the client's *reflectAttributeValues()* through RTI. Then the clients initialize the audio device using the given audio format.

Then, the server begins to play and invokes the *updateAttributeValues()* for the "Operation". In the same way, the clients get the current value of the "Operation" and begin playback.

During the playing, the server transfers audio and video packets to all the clients by calling *sendInteraction()*, which invokes the client's *receiveInteraction()*. Within *receiveInteraction()*, audio and video packets are separated. The audio data are put into local audio buffer queue and played back at the correct time, while the video data are drawn directly to the window of the client application.

4.8.4. Client resign

Figure 4.9 lists the resign scenario of the client when the server is active. (The client can resign after the server. In this situation, the server's callback steps should be deleted from the scenario.)

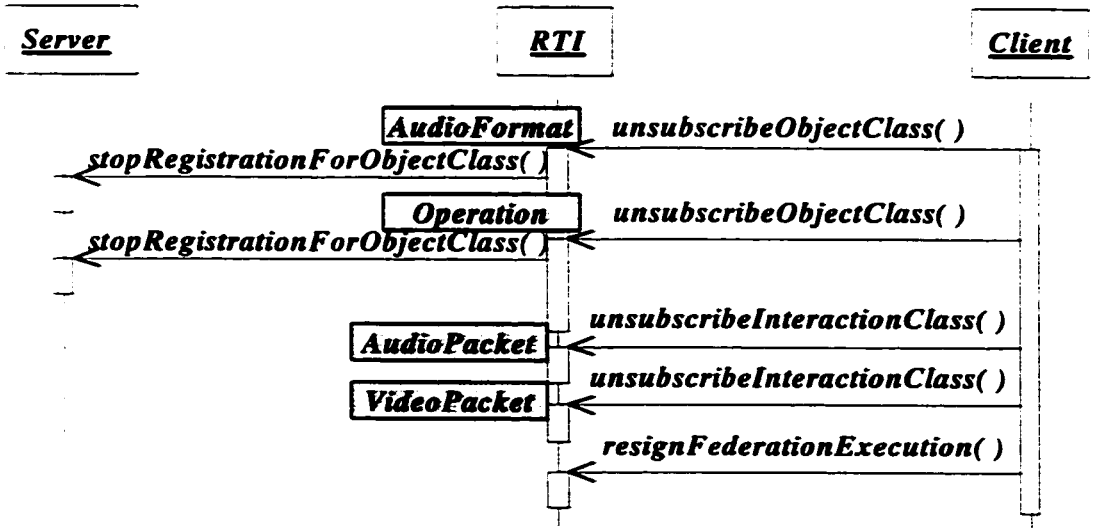


Figure 4.9 Client Resigns

When the user chooses exit at the client, the client calls *unsubsribeObjectClass()* for both the objects "AudioFormat" and "Operation", which invokes the server's *stopRegistrationForObjectClass()* through RTI to make the server know that a client is leaving and losing interest to the objects.

Then the client calls *unsubscribeInteractionClass()* for both the interactions "AudioPacket" and "VideoPacket".

Finally, the client calls *resignFederationExecution()* and *destroyFederationExecution()* to resign from the federation.

4.8.5. Server Exit

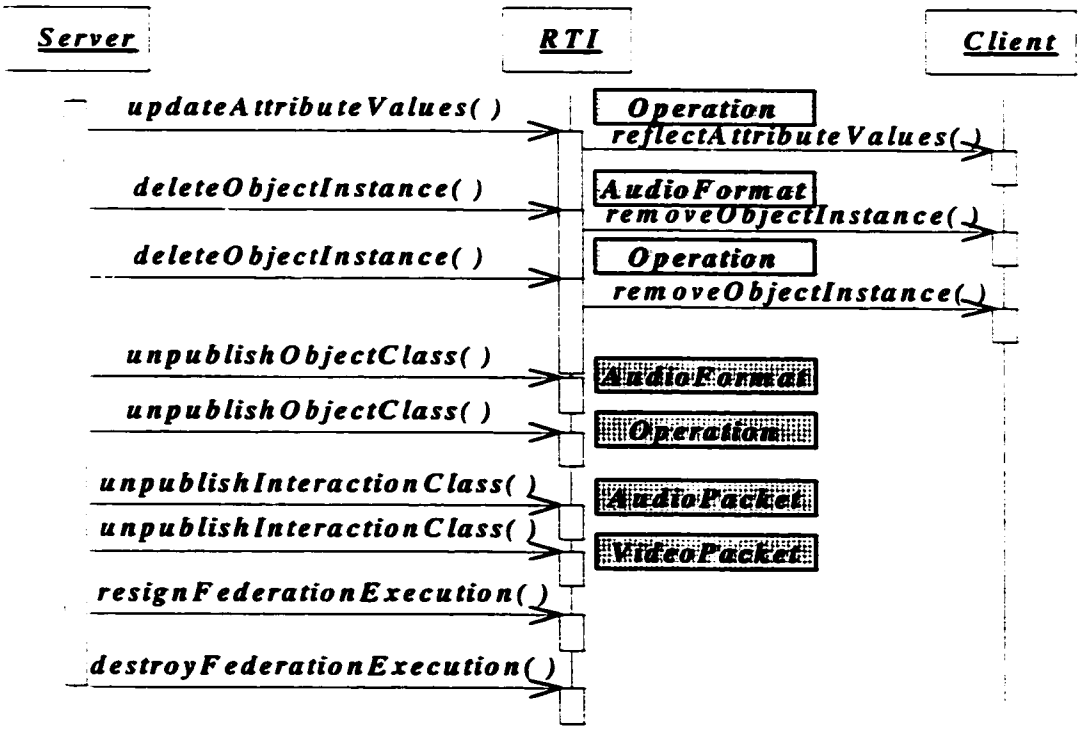


Figure 4.10 Server Resigns

Figure 4.10 lists the resign scenario of the server when the client is active. (The server can resign after the client. In this situation, the client's callback steps should be deleted from the scenario.)

When the user chooses exit from the server application, the server updates the "Operation" to notify all clients to stop playing by calling *updateAttributeValues()*.

Then the server calls *deleteObjectInstance()* which invokes the clients' *removeObjectInstance()* through RTI. This step notifies the clients that the server is resigning.

Then the server quits publishing the objects and interactions and resigns from the federation.

4.9. Implementation

We implemented the Client/Server application over RTI1.3v4 at first, and ported them to RTI-NG1.3v2 later. The system is tested in a 100baseT LAN with WindowsNT4.0 as platforms. There are one server and four clients in the test. All the features listed in *section3 Feature Specification* are realized.

The AVI file used in the test contains both audio and video streams. The audio is ADPCM format with 2 channels, 11.025K sample rate, 4bits/sample. The video frame is the RGB format with parameters 320x240x24. Each frame has 230400 bytes.

In this test system, the quality of audio and video are as good as play at local.

Chapter 5

Discussion about HLA Stream Solution

5.1. Advantages of the Stream Solution

- Compatible with HLA standard. This stream scheme uses only HLA defined APIs to realize stream transmission and control.
- Flexible and accurate schedule. It uses the system interval timer to control the schedule. This is a very flexible and accurate way to control the schedule. It is easy to integrate it with real-time applications.
- Lightweight. It is a lightweight approach. For CD quality audio, it allocates 5msec CPU time in a 100msec period or less than 1 msec in a 40msec period.
- Multiple streams support. Audio/video applications usually use the timer from the audio device to control the schedule. Since there is usually only one audio device in a computer, it can not control multiple streams at the same time. In the application of audio/video retrieving server in DVE, multiple users would access the server and retrieve audio/video streams at the same time. This solution can support this kind of requirement since it adopts the system interval timer to control the schedule, while the system interval timer is an unlimited resource of the computer.

5.2. Clock Skew

This experiment does not handle clock skew issues among different PCs. In fact, during the experiments, clock skew does not affect quality obviously.

We can analyze the effect of clock skew. If the receiver's clock is faster than the sender's clock, the current read position pointer will catch up the current write position pointer and the contents in the buffer will be repeated one time (refer to Figure 5.1). If the sender's clock is faster than the receiver's clock, the current write position pointer will catch up the current read position pointer and the contents in the buffer will be overlapped (refer to Figure 5.2).

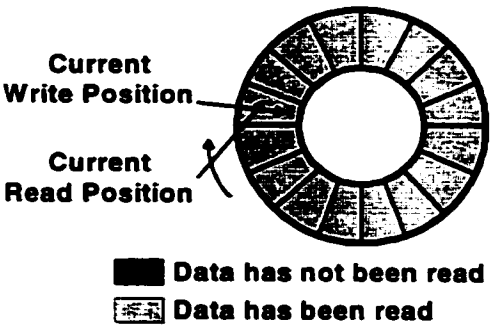


Figure 5.1 Clock Skew – Repeat Sound

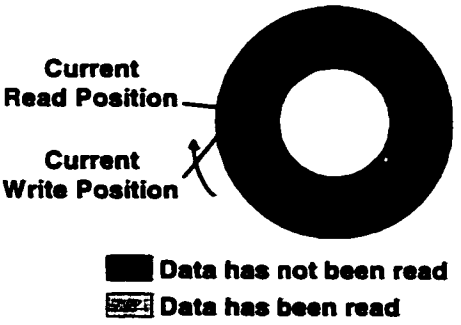


Figure 5.2 Clock Skew – Lost Sound

When we set the reconstruction buffer to 2000msec, and run sender and receiver in different PCs, we can feel the obvious repeat or drop of an audio segment. But when we set the reconstruction buffer to 300msec and test between the same two PCs, we can not feel any distortion. The conclusion is that the clock skew effect on the audio quality can be ignored when the reconstruction buffer is not very long.

In some situations where the clock skew can not be ignored, the skew adjusting algorithm should be adopted. The clock skew can be adjusted by adjusting the timer interval dynamically.

5.3. Packet Order

There is one important part about stream control that may need improvement. The client does not judge whether the received packets are in order. The client processes all packets following FIFO rule.

In the LAN environment, the quality is good for both audio and video. While, the situation is different for the best-effort Internet. The client should sort the coming packets based on their serial number and keep the synchronization between audio and video streams. On the other hand, it will be OK if the clients and the server communicate through a QoS-enabled network.

5.4. Prototype of Videoconference Federate

Obviously, this scheme can be used easily to realize interactive stream transmission. Figure 5.3 gives an architecture of a videoconference based on this scheme.

Based on the audio retrieve experiment mentioned above, we can add audio capture, video capture, video playback tasks to complete a videoconference federate for RTI.

In a videoconference application, the audio stream has higher priority than the video stream. So in this architecture, we handle the audio stream and the video stream through their individual paths. This architecture will be good for QoS control, since an application can control the audio and video transmission rates separately. When the available network bandwidth is limited, we can slow down the video stream to maintain audio quality.

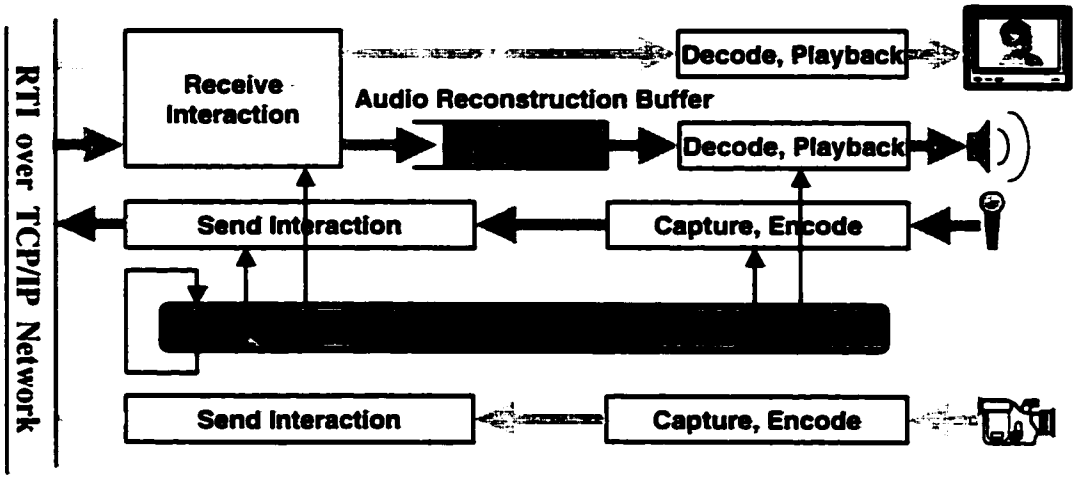


Figure 5.3 Videoconference over RTI

Since the audio stream is sensitive to delay jitter and latency, we use the system interval timer to control audio processing parts, including:

- Audio capture task
- Audio playback task
- Audio stream send task

With an accurate schedule, the audio stream can maintain stable data rate. At the same time, the Audio reconstruction buffer is used to eliminate delay jitter.

The Video part does not need a reconstruction buffer. We have mentioned that the effect of jitter on video is jerky rendering of frames, a degradation that can be tolerated by users. The video part will playback all video frames directly when it receives them [32], without using a reconstruction buffer to eliminate delay jitter.

In fact, the video part in a common videoconference is not a critical real-time part as compared to the audio part, so we do not need to use the system interval timer to control video stream schedule. Video capture and sending can be handled in a foreground loop. The quality of video depends on available CPU time.

In this system, audio and video data are received through *receiveInteraction()* which is called by *RTI::tick()*. Within *receiveInteraction()*, audio data and video data are divided to different processing paths. Audio data are put into the audio reconstruction buffer to eliminate delay jitter and then playback. Video data playback directly.

Finally, a conference system must support multiple users. This issue should be handled in the encode/decode part. RTP is a good protocol to realize this target.

Part III

HLA Real-Time Extension

Chapter 6
Real-Time Enabling Technologies

6.1. RT-DIS and End-to-End Predictability

RT-DIS and DVE (1.1) involves many mission-critical applications such as virtual surgery [33,34], pilot training [35], people/equipment in the loop simulation [36], Human modeling [37], and civilian simulations such as DVE, e-commerce, distance learning, etc.

Obviously, RT-DIS demands end-to-end predictability meaning that the DIS system must behave predictably in order to decide whether a real-time requirement is met or not. This can only happen if all the parts of the DIS system behave deterministically in addition to "combining" predictably.

- End systems: RT-DIS requires real-time response and predictable behavior from the end systems in order to interact with the physical world within specific delay bounds and present data, images, audio, video, etc to users in real-time.
- Networks: RT-DIS end-users may be distributed across the Internet or intranets. Thus, they require predictable network performance in order to provide low-latency and high-bandwidth to applications.

Briefly speaking, RT-DIS requires a range of predictability support mechanisms from both network and end systems.

6.2. Real-Time End System

The traditional dichotomy divides operating systems (OS) into General Purpose OS (GPOS) and Real-Time OS (RTOS). GPOSs (such as traditional UNIX and Windows) are designed to maximize system throughput in a multi-user, time-shared environment, while RTOSs are designed to deliver predictable services to all tasks in order that they may meet their deadlines.

6.2.1. Predictability Enhanced Technologies

There are several general OS technologies for bounding the response time and improving predictability whose use is not limited to RTOSs.

Multitasking

Multitasking makes an OS handle multiple external events independently, associating each event with a separate thread of execution and its own set of system resources. Multitasking improves the scalability of an OS since tasks can be distributed to multiple processors running independently in a multiprocessor environment. With multitasking OSs, the system capability can be improved in a straightforward manner by an increase in the number of processors. Usually, OSs implement multitasking through a multi-thread mechanism.

Priority and Scheduling

In a multitasking environment, it is possible for more than one task to simultaneously require the use of the processor. Hence an algorithm is needed to decide which task will be executed first. A preemptive fixed-priority scheduling approach is commonly used in all modern OSs for this purpose.

With the fixed-priority scheduling approach, a program assigns priority to a task before it is submitted to an OS and the priority will not change during run-time. On the other hand, the OS ensures that the CPU is always allocated to the highest priority task that is ready to run. The OS kernel maintains a ready queue for each priority level. Whenever a task is ready to execute, the kernel places it in the ready queue of the task's current priority. For the tasks in the same ready queue, further scheduling approaches are used to identify the task executed first:

- First-In-First-Out (FIFO) Scheduling – all tasks in the queue execute one after another.
- Round-Robin (RR) Scheduling – share the CPU fairly among all tasks in the queue by time slicing.

The most commonly used algorithm for preemptive fixed-priority scheduling is the Rate Monotonic Scheduling (RMS) [38]. The RMS demands different priority levels for each real-time task. A complex system might need many priority levels. Generally speaking, 128 priority levels are a must [39], while 256 system priority levels are sufficient even for the most complex systems [40].

Predictable Task Synchronization

In addition to competing for processor use, multiple tasks compete for other resources such as memory, I/O ports, etc. Therefore OSs should address the requirements of both mutual exclusion and task synchronization.

All modern OSs provide a semaphore mechanism for this purpose. A semaphore protects each resource that is possibly accessed by multiple tasks simultaneously. A task is granted access to a resource depending on the indication of the semaphore associated with the resource.

The OS kernel maintains a waiting queue for each semaphore. Whenever a task is suspended and waiting for that semaphore, the kernel places it in the queue. GPOSs usually implement semaphore queues in FIFO order and do not consider priorities of the waiting tasks. A FIFO order semaphore queue causes the possibility that a low priority task gets the resource prior to a high priority task, which is undesirable in real-time applications. Some RTOSs implement the semaphore queue in priority order. A higher priority task always gets a semaphore prior to a lower priority task.

Priority inheritance

Priority inversion happens in some real-time systems when a low-priority task that owns a semaphore is preempted by a high-priority task that requires access to the resource guarded by the semaphore. When the high-priority process blocks, pending on that semaphore, a third, medium-priority process runs, leaving the low-priority task that owns the semaphore pending as well. In this case the high-priority task could remain blocked indefinitely.

A priority-inheritance algorithm can solve priority inversion. The priority-inheritance algorithm assures that a task that owns a resource executes at the priority of the highest-priority task blocked on that resource. Once the task priority has been elevated, it remains at the higher level until all mutual-exclusion semaphores that the task owns are released, after which the task returns to its normal priority. Hence, the "inheriting" task is protected from preemption by any intermediate-priority tasks [41].

Fast interrupt response/context switch

Another key facility in real-time systems is hardware interrupt handling since interrupts are the usual mechanism to inform a system of external events.

RTOSs usually use the fastest possible means to deal with interrupts by running interrupt service routines (ISRs) in a special context outside of the task context [41]. Therefore, handling an interrupt does not involve the overhead associated with a task context switch, which denotes the term of "fast context switch".

Interrupt Handling – Priority DPC

Typically an ISR will simply post a request for a deferred procedural call (DPC) to be queued and then relinquishes the CPU. The DPC is responsible for the major work to be done in response to the interrupt – the DPC will run at a later time on behalf of the ISR.

Usually GPOSs add all pending DPCs to a FIFO queue. Once executing, a DPC will run to completion and cannot be preempted by another DPC or a thread. ISRs always run before DPCs and DPCs always run before user threads [39]. Same as the FIFO semaphore queue, a FIFO DPCs queue causes unpredictable delay to higher priority tasks and is not suitable for RTOSs.

Some RTOSs implement a priority based DPC queue. A higher priority DPC is inserted into the queue in the position prior to all lower priority DPCs so a higher priority DPC is always handled earlier.

Priority Message Queue

Modern real-time applications are constructed as a set of independent but cooperating tasks. Message queues are the primary means for inter-task communication. These queues can be in a FIFO order or priority order. With a priority message queue, the higher priority message is inserted before all messages with lower priority so it can reach the destination earlier.

Memory Lock

Many OSs perform memory paging and swapping. These techniques allow the use of more virtual memory than there is physical memory on a system by copying blocks of memory to disk and back.

These techniques impose severe and unpredictable delays in execution time and are therefore undesirable in real-time systems. RTOSs usually do not use virtual memory mechanism, or they have *page-lock* facilities to declare that certain blocks of memory must not be paged or swapped.

Thread-pool

It has long been known that one of the ways to improve the performance of a software application is to reduce the overall number of objects created and destroyed [42]. A thread-pool mechanism is used to improve application performance by decomposing thread execution from thread creation and destruction.

A thread-pool is a collection of N threads that are used repeatedly to run small tasks [43]. The N threads are usually created at application initialization and destroyed at application termination. When a task needs to be processed, it is handed off to the thread-pool and the thread-pool runs it inside one of its N threads. If all N threads are busy processing other tasks. the task waits for one to become available.

Number of Jobs	Without Pool (Approx. ms)	With Pool (Approx. ms)
<15	17	25
100	64	30
1,000	603	47
50,000	24100	1980

Notes: Sun Ultra-2 with two 400MHz UltaSPARC-II processors and 512MB of RAM, JDK1.2.2.

Table 6.1 Thread Pool Performance Improvement [42]

Some tests (refer to Table 6.1) illustrate an important point: thread pools will not always lead to better performance. However, as soon as the number of jobs reaches around 100. performance with the pool is at least two times better than that without the thread-pool. With about 1,000 jobs the performance with a pool is more than 10 times better [42]. The standard specification of Real-Time CORBA has adopted the thread-pool as its necessary mechanism [47].

Known OS Behaviors

To provide an accurate and predictable response, OSs should also specify all the timing of system calls and the system behaviors. Usually only RTOSs can provide the timing sequence of system calls and behaviors.

6.2.2. OS Standard

To provide the portability of applications among various OSs, ISO/IEC [44] IEEE and The Open Group [45] released the Portable Operating System Interface (POSIX) standard [46], which is a suite of API standards. Applications that conform to the standards are portable at the source code level across other compliant OSs. POSIX1003.1 defines the basic functions of an UNIX-like OS. All modern GPOSs support this standard.

The real-time (1003.1b) and thread (1003.1c) extensions of POSIX1003.1 define a subset of functions for multithreaded real-time applications, which identify the necessary features to facilitate predictability. As the only standard for RTOS, all current RTOSs identify it as an objective to be compliant with. Real-Time CORBA also identifies POSIX1003.1b compliant OSs as the Real-Time CORBA capable platforms [47].

6.2.3. Current Operating Systems Overview

Since the first RTOS was produced more than 20 years ago by DEC for the PDP family of machines [48], there have been many more RTOSs in the current market [49] with VxWorks of WindRiver [50] and QNX of QNX Software Systems [51] sharing the leading position.

Known as GPOS, Sun's Solaris and Microsoft's Windows NT also provide a set of mechanisms that allow real-time activities to be performed.

I summarized the real-time features of major OSs (Table 6.2) based on their product specifications or their real-time programmer's guides [52,53,54,55,56,41 ,57,58].

Not all of the predictability improvements summarized in section 6.2.1 are controllable (such as *fast context switch*) but most of them can be controlled via the OS's native API. In order to achieve the best possible real-time performance, a real-time application can enable all of an OS's real-time features via a native API. For the purpose of portability, a real-time application can enable part of OS's real-time features via POSIX.1b interfaces limited in the priority multitask, priority message queues and memory locking.

Operating Systems		**POSIX 1003.1b**	**Solaris**	**WinNT**	**VxWorks**	**QNX**	**IRIX**	**RT Linux**
Scheduling Policy		Fixed/Dyn	Fixed	Fixed	Fixed	Fixed	Fixed	Fixed
Priority Levels		32	60	7	256	29	32	100
Scheduling Classes	**FIFO**	✓	✓	✓	✓	✓	✓	✓
	RR	✓	✓	✓	✓	✓	✓	✓
Priority Inheritance		✓	✓	×	✓	✓	✓	×
Priority Message Queue		32	×	×	1	32	32	32
Priority Semaphore Queue		-	×	×	✓	×	✓	✓
Fast Context Switch		-	×	×	✓	✓	✓	✓
Priority DPC Queue		-	×	×	✓	✓	✓	×
Memory Locking		✓	✓	✓	✓	✓	✓	×
Application Class		-	Soft	Soft	Hard	Hard	Hard	Hard

Table 6.2 Real Time Supports of Operating Systems

To provide an accurate and predictable response, RTOSs usually specify all the timing of system calls and the system behaviors.

Predictable Critical Characteristics	VxWorks5.3.1		QNX4.25		WinNT4 With RTX4.2		WinNT4	
	Ave. (μs)	Max. (μs)	Ave. (μs)	Max. (μs)	Ave. (μs)	Max. (μs)	Ave. (μs)	Max. (μs)
Interrupt Latency	1.9	2.8	2.0	3.6	11	19	3.8	22.7
Interrupt Dispatch (128 threads)	3.2	8.4	–	–	13.2	33.6	13.6	178.1
Thread Creation	30.8	60.3	950	1022	114	195	510.4	1155.8
Thread Deletion	9.5	16.8	–	–	101	126	12.3	16.0
Thread Switch (10 threads)	2.5	16.0	1.2	9.8	5.7	24.2	5.4	53.1
Thread Switch (128 threads)	3.3	28.4	2.0	11.1	8.9	27.3	8.0	1255.3
Semaphore Creation	6.4	11.3	12.3	20.7	9.9	22.2	81.4	178.0
Semaphore Deletion	5.0	23.4	11.3	15.1	11.3	28.9	20.7	63.2
Semaphore Acquisition	< 0.2	0.6	0.7	1.5	1.8	12.3	3.3	38.0
Semaphore Release	< 0.2	0.2	0.7	7.8	1.6	19.1	2.6	6.8
Mutex Priority Inversion Prevention	10.3	32.9	25.0	73.3	10.6	18.1	–	–

**Interrupt Latency: task to interrupt handler*

***Interrupt Dispatch Latency: interrupt handler to task*

Table 6.3 Real-Time Characteristics of Various OSs

Many evaluation reports [59] from the *Dedicated Systems Magazine* give detail performance testing result of various OSs. Table 6.3 shows some real-time characteristics of the OSs.

The result shows that not only RTOSs such as VxWorks and QNX can provide excellent predictability, but also GPOSs such as Windows NT can provide reasonable predictability with a real-time extension module. The real-time feature is not a luxury of OSs anymore. The DIS applications can expect real-time support from today's OSs.

Not only RTOSs such as VxWorks and QNX can provide excellent predictability, but also GPOSs such as Solaris and Windows NT provide reasonable predictability and may provide more accurate real-time predictability via their extensions [60,61].

6.3. Real-Time Network Communication

Nobody doubts that the Internet is becoming the dominating data network that is going to provide integrated services including media-on-demand, interactive-multimedia and even the traditional telephone services. Due to its great flexibility, it is the best network candidate for RT-DIS, provided that it can exhibit the appropriate predictability. With the continuous effort of the Internet community, the Internet is going to support different levels of predictability.

It is the aim of this section to investigate the possibility of obtaining QoS in the Internet and Campus Networks. It is helpful to present a brief introduction of the Internet architecture, its device components and the mechanisms and standards that strive to provide QoS over this network.

6.3.1. The Internet Architecture

The Internet is a collection of interconnected component networks that share a common addressing structure, a common view of routing, and a common view of a naming system [62]. It consists of Internet Service Clients (ISC) and Internet Service Providers (ISP) (Figure 6.1). The ISCs could be either Network Access Clients (NAC) such as enterprise, campus networks or Dial Access Clients (DAC) such as end users connecting to the Internet through PCs and modems. The ISPs can be divided into several categories according to their service category and geographical coverage:

Access ISPs (Local ISPs): These are terminating point-to-point modem connections (including V.90, Cable/DSL and ISDN digital modems.) from local residents, small enterprises through their Point-of-Presence (PoP), aggregating data traffic from 56Kbps downlinks to T1(1.544Mbps) or T3 (45Mbps) uplinks and providing Service Management and Authentication, Authorization and Accounting (AAA) as well as value-added services such as Web-Hosting, E-mail Server, etc.

Regional ISPs: These provide: provisioning of permanent access to NACs and access ISPs with some level of QoS (bandwidth, delay, reliability, etc); higher value-added services such as Virtual Private Network (VPN); further aggregating data traffic from T1, T3 speed downlinks up to OC-192 (622Mbps); connecting to neighbor ISPs through Peering connections.

National/International ISPs (Transit ISP): These provide: carrier facilities (including ATM, Frame Relay, SONET/SDH, DWDM, Satellite Links, Undersea cables)

for national and international high speed data transmission; Connection to other transit ISPs through Network Access Points (NAP) for comprehensive connectivity.

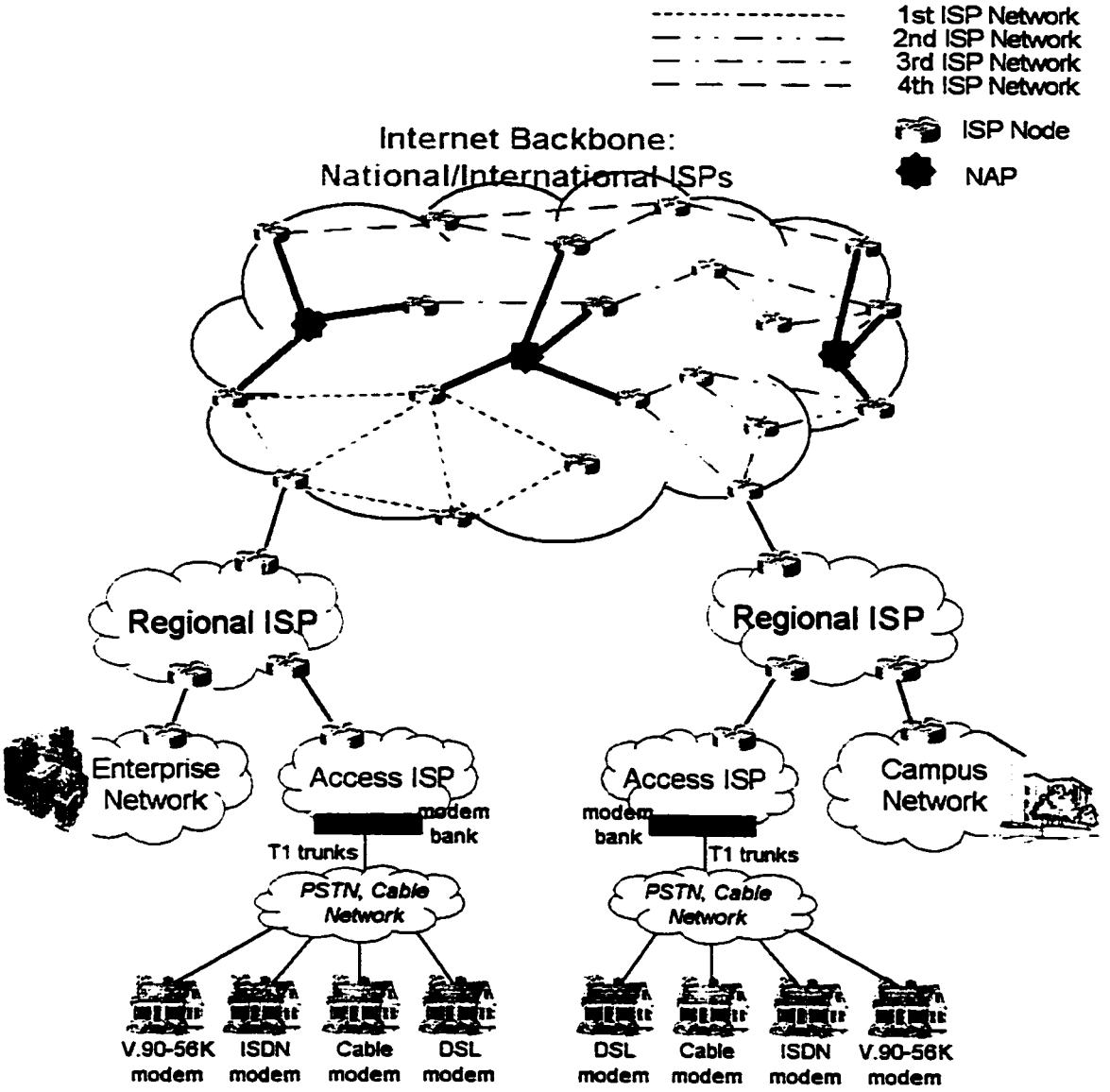


Figure 6.1 Internet Architecture

The different ISPs at different scales cooperate with one another and provide seamless end-to-end connectivity and service to all the clients.

6.3.2. Internetworking Devices

Many network equipments are used in the Internet, connecting *end systems* and *local area networks* together such as Hubs, Switches, Routers, etc.

Hub is the simplest and cheapest device that connects workstations to an Ethernet and provides a logical bus attached to by all workstations. The bandwidth of the bus shared by all the workstations could be 10Mbps (Ethernet) or 100Mbps (Fast Ethernet).

Layer 2 Switches are alternatives to building an Ethernet, which provide a switch fabric that all end systems connect to. Each port of the switch has a dedicated 100Mbps (Fast Ethernet) or 1000Mbps (Gigabit Ethernet) bandwidth such that high communication performance among the attached end systems is provided. Ethernet switches use Layer 2 MAC addresses to identify the packets and end systems rather than IP address.

Routers are special-purpose computers connected to at least two networks. Each router determines the next network point to which a packet should be forwarded toward its destination by computing and maintaining a routing table. A classical router works on Layer 3 using IP addresses as identification of packet and source/destinations.

Layer-3 switches are alternative devices for routers. A layer-3 switch divides router functionality into two parts: 1) route computing, 2) packet forwarding. They use ASICs to implement packet forwarding to improve the performance and are very fast compared to software-based routers.

Multi-layer switches/routers address intelligent requirements of internetworking devices by accessing Layer 2, 3, and 4 information including the MAC address, IP address, TCP/UDP port, protocol ID, etc. Multi-layer switches/routers can provide

sophisticated QoS according to this information. It is also possible to access higher layers for billing, accounting, and provisioning purposes. The main difference between multi-layer switches and multi-layer routers is that the software-based multi-layer router performance can drop with an increase in the amount of information read, whereas ASIC-based multi-layer switches work at wire speed regardless of the features enabled.

ATM Switches consist of cell-based circuit switching technology. The advantages of ATM switches are scalability (from 25.6 Mbps to 2.4Gbps) and QoS (guaranteed bandwidth, loss rate, delay and jitter). In the Internet, ATM switch was ubiquitously used as Layer 2 technology spanning from LAN to the Internet backbone providing scalable bandwidth and QoS. This situation has been changed due to the introduction of Gigabit Ethernet and 10 Gigabit Ethernet technologies and 802.1p Ethernet QoS technologies.

The goal of providing QoS has been addressed by various mechanisms and standards with varying degrees of success. The QoS verification and results presented in the next section may be presented in better light if a short discussion of these mechanisms and standards at the various layers (ranging from network to application) precedes that presentation.

6.3.3. Network QoS Principle

QoS means providing service differentiation and performance assurance for Internet applications according to their requirements. Performance assurance addresses bandwidth, loss, delay and jitter.

Data prioritization and queuing systems are the mainstream QoS tools available today. Once data is prioritized using implicit or explicit techniques, queuing/scheduling

algorithms and congestion control mechanisms are used to provide the appropriate or desired QoS.

Queue/Schedule Management

Switches and routers mark traffic and use internal queue management and scheduling mechanisms to enforce differential services according to traffic priorities. Different priority traffics are held in different queues and high priority traffic is sent to the network before low priority traffic.

A queue management scheme decides which packets can be stored as they wait for transmission. Scheduling mechanisms control the actual transmission of packets. Many queuing algorithms are used in current IP routers and switches, such as FIFO queuing, Priority Queuing, Custom Queuing, Weighted Fair Queuing (WFQ) [63], which provide different behaviors for realizing QoS.

Let's use a WFQ to illustrate how a queue scheme can support QoS.

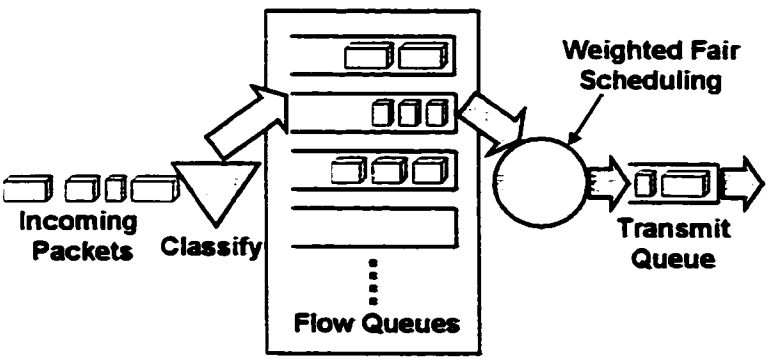


Figure 6.2 Weighted Fair Queue [63]

WFQ is a flow based queuing algorithm as shown in Figure 6.2. Each flow is identified according to its address, protocol, port number, etc. Each flow is assigned a queue to keep incoming traffic. The weighted fair scheduling algorithm is used to

calculate the outgoing rate of these queues. The weight is determined by QoS requirements of flow and policy.

WFQ can support both differentiated and guaranteed QoS and is desirable for providing consistent response time. It also ensures that different categories of traffic cannot resource starve other categories, refer to Figure 6.3.

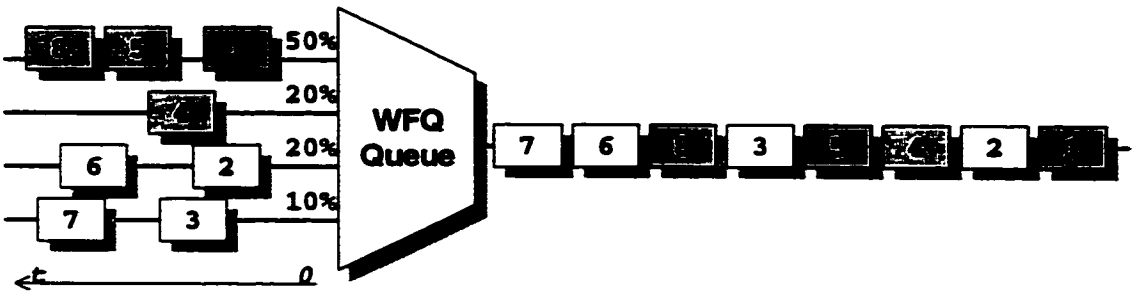


Figure 6.3 Weighted Fair Queue [62]

Congestion Control

Congestion is the dominant reason for packet loss. Congestion control usually causes packet retransmission that degrades network performance and predictability. So congestion control mechanisms are important for delay-sensitive applications.

The Random Early Detection (RED) algorithms are popularly used to avoid congestion. RED works by monitoring traffic load and stochastically discarding packets if the congestion begins to increase. The result of the drop is that the source detects the dropped traffic and slows its transmission. Weight RED (WRED) can selectively discard lower-priority traffic when the congestion increases, to provide differentiated performance characteristics.

6.3.4. Layer 2 QoS – IEEE 802 Ethernet QoS

Ethernet is the most widespread LAN access technology and will be used as MAN/WAN technology due to the Gigabit Ethernet and coming 10Gigabit Ethernet evolutions. As a result, the importance of providing QoS mechanisms ought not to be neglected.

It is well known that ATM and Frame Relay, used as Layer 2 transit technologies in the Internet can support abundant QoS, while Ethernet, another Layer 2 technology, is believed to lack QoS support. However, this situation has been altered by QoS improvements in the Ethernet standard, some of which are briefly outlined in order to provide the necessary background information.

6.3.4.1.IEEE 802.1p – Traffic Prioritization & Dynamic Multicast Filtering

A relevant standard is that of IEEE 802.1p entitled "Supplement to MAC Bridges: Traffic Class Expediting and Dynamic Multicast Filtering" which was later incorporated into the IEEE 802.1D standard [64] (June 1998 draft revision) for "LAN MAC Bridges". It covers traffic classes and dynamic multicast filtering in bridged LANs and describes important methods for providing QoS at the MAC level through separate queuing of time-critical frames to reduce the delay and jitter in a MAC level switch.

Traffic Prioritization

Under 802.1p, a 4-byte Tag Control Info (TCI) field is inserted in the **Layer 2 MAC header** which contains three priority bits (Figure 6.4), identifying eight traffic types (Table 6.4).

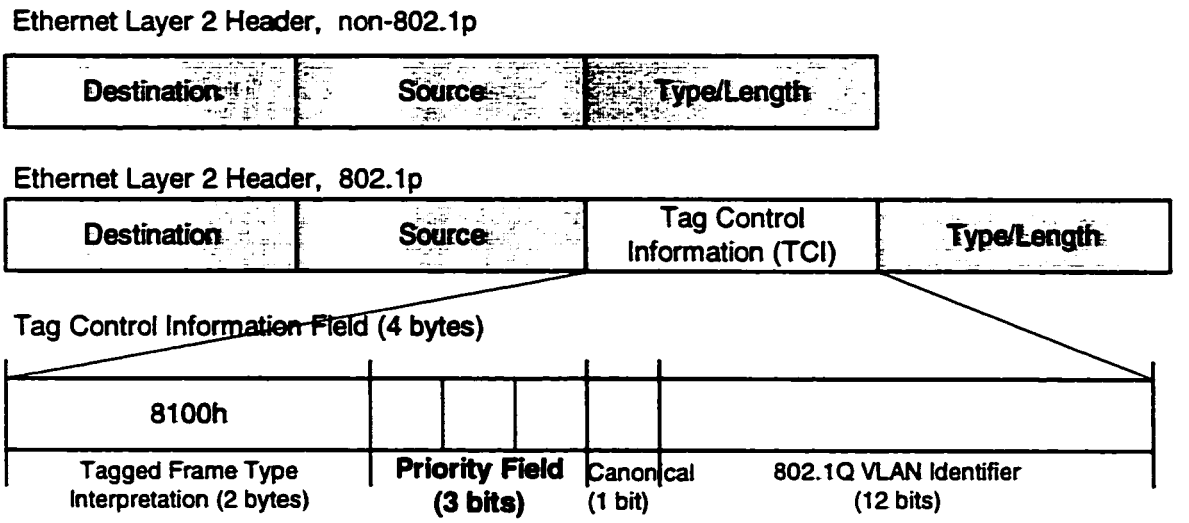


Figure 6.4 802.1p Tag

Tag	Traffic Type
0	Best Effort (Default Value), which is the traffic flowing on most networks, today
1	Background, which can flow on the network without impacting other users and applications, such as bulk transfers.
2	Spare for future use
3	Excellent Effort (Business Critical) which will be treated with the best best-effort service that a network can provide
4	Controlled Load (Streaming Multimedia) which are important business applications subject to some form of admission control, ranging from bandwidth pre-provisioning through reservation per flow requested at the start of the flow
5	Video (Interactive Media), less than 100 milliseconds latency and jitter which requires less than 100 ms delay
6	Voice (Interactive Voice), less than 10 milliseconds latency and jitter which requires less than 10 ms delay and jitter (one way transmission through the LAN infrastructure of a single campus)
7	Network Control (Reserved Traffic) which is required to keep the network operational and must get through the network with the highest priority

Table 6.4 IEEE 802.1p Traffic Types [65]

The eight traffic types each demand different QoS. New Ethernet switches usually enable this feature through their queuing mechanism.

In an Ethernet switch, a forwarding process may provide one or more transmission queues for each bridge port and provides storage for queue frames and waits for an opportunity to submit these transmissions where frames will be assigned to each queue according to their priorities.Different vendors may implement 802.1p in different number of traffic queues. For example, Cisco catalyst 8500 switches support 4 queues at their bridge ports, while Nortel Passport 8000 routing switches support 8 queues.

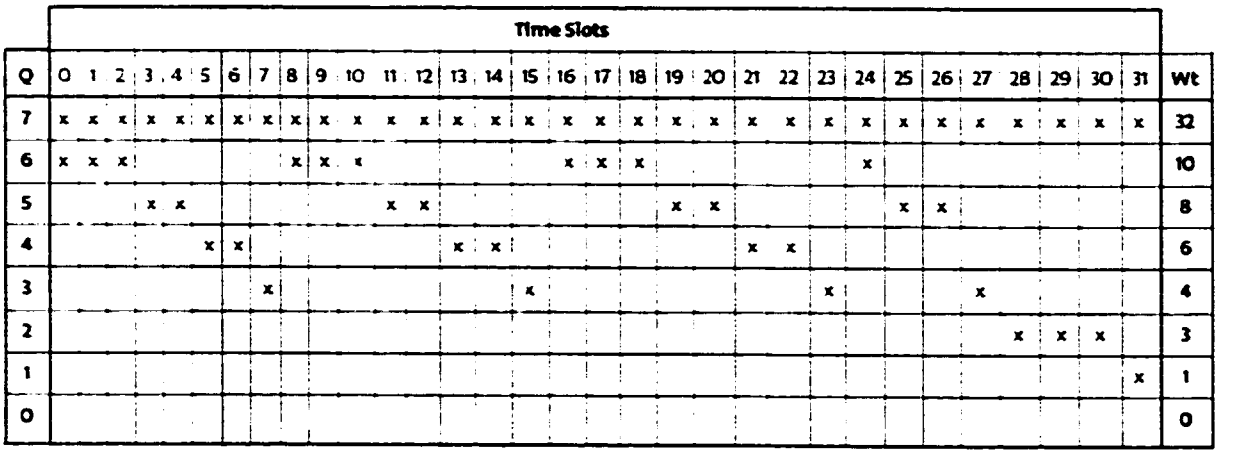


Figure 6.5 802.1p WFQ in Passport 8000 routing switch [65]

The example in Figure 6.5 illustrates eight queues with different weights assigned (queue 7 with 32 weights, queue 6 with 10 weights, etc). Using this weight assignment, the switch can handle two packets in each time slot. Different priority packets get differentiated throughput and latency when going through the switch.

Dynamic Multicast Filtering

Is this a part of 802.1p? Yes. The purpose of Dynamic Multicast Filtering (DMF) is to restrain multicast traffic in a switched LAN. By default, a LAN switch floods multicast

traffic to all ports of the switch, consuming a lot of bandwidth if many multicast groups are active in the segment. With DMF, the switch listens to the group reports from each port in a LAN and builds a filtering database of multicast group members per port. Multicast frames received for a group are forwarded only to those ports that have members in the group addressed. IGMP Snooping, an implementation of DMF, has been supported by major internetworking device vendors such as CISCO and NORTEL. CISCO also implements DMF in a proprietary way named CGMP [65,66].

6.3.4.2. IEEE 802.3x – Flow Control

The IEEE 802.3x standard [67] addresses flow control over Fast Ethernet and Gigabit Ethernet.

Flow control is a mechanism created to manage the flow of data between two full-duplex Ethernet devices. A switch can send PAUSE packets on a port operating in full duplex mode if the amount of packets received on this port is more than the switch can handle.

Flow control can match up the sending and receiving device throughput. It is very useful to support high-performance workstations that have high throughput ability that may flood the switch.

6.3.4.3. IEEE 802.3ad – Link Aggregation

The IEEE 802.3ad [68] Link Aggregation allows one or more links to be aggregated together to form a Link Aggregation Group such that a MAC Client can treat the Link Aggregation Group as if it were a single link. It allows multiple parallel links between switches, or between a switch and a server to provide greater resiliency and bandwidth.

Typically, the IEEE 802.1d [64] Spanning Tree Protocol (STP) is used to prevent loops forming between these parallel links.

6.3.5. Layer 3 QoS – IETF IP QoS

With the continuous effort of the Internet community, the Internet is going to support different levels of predictability through QoS mechanisms. There are three major QoS approaches from the Internet Engineering Task Force (IETF): IntServ [69], DiffServ [70], MPLS [71], in addition to the old IP TOS mechanism.

IP Precedence / Type of Service (TOS)

This needs an introduction about how Upper QoS approaches will be presented. An early attempt to allow specification of levels of quality was in the definition of IP precedence settings in the original IP protocol specification in September 1981 as a part of RFC 791. Within the **Layer 3 IP header** an eight-bit type of service (ToS) was defined to identify precedence (3 bits for 8 levels), delay (1 bit), throughput (1 bit), reliability (1 bit), etc. Unfortunately, this feature has not been widely implemented.

Integrated Services (IntServ)

The earliest approach to QoS and beginning in 1990, Integrated Services model proposed by the IETF IntServ working group inherits the connection-oriented approach from telephony network design. In this scheme, an application reserves network resources (such as queue size or bandwidth) through a signaling protocol (typically, the Resource Reservation Protocol – RSVP) along the transmission path. Subsequently, the application can transfer data through the reserved resources. These resources will not be assigned to other applications until the reserving application releases them. With resource

reservation, the application's quality requirements such as bandwidth and end-to-end delay can be guaranteed.

In the context of network technology, flow is defined as an individual unidirectional data stream transferred from sender to receiver and uniquely identified by the 5-tuple – protocol type, source IP address, source port number, destination IP address and destination port number. With flow as its operation unit, IntServ supports fine-grained service differentiation based on each individual flow.

Integrated Services (IntServ) offers two QoS control services for delay-sensitive traffic. *Guaranteed service* [72] guarantees a deterministic upper limit on delay as a packet travels through the network (suitable for real-time interactive audio and video). *Controlled Load service* [73] provides a fairly stable, although probabilistic (predictive), upper limit on delay (suitable for audio and video streaming like WebTV).

However, IntServ has a scalability problem, holding it back from widespread use. Its flow-based processing and resource reservation mechanism is impractical for a router or switch in the Internet core that may contain hundreds of thousands of flows.

Differentiated Services (DiffServ)

The IETF DiffServ working group presents another architecture for implementing scalable service differentiation for the Internet.

Before entering a DiffServ network, the application's traffic is classified and marked with a priority level at network boundaries or hosts. Within the DiffServ network, each network element stores and forwards the marked packet depending on the priority information contained in the mark. High priority packets get preferential treatment but

there is no resource reservation for any traffic.The DiffServ model can also support a wide range of services from bounded delay and jitter to different throughput and drop precedence services.

Compared to IntServ, DiffServ is aggregate-based and hence provides good scalability with the cost of coarse-grained service differentiation.

Multi-Protocol Label Switching (MPLS)

The IETF MPLS working group is standardizing the label switching technology that can support traffic engineering and constraint routing (including QoS routing), not provided by DiffServ, hence allowing for MPLS to improve quickly along with DiffServ in IETF.

MPLS domain resources are divided into multiple channels with different QoS characteristics. Before the user's traffic enters an MPLS domain, each traffic packet is marked with a label that identifies a channel according to its QoS requests at the domain boundary node. Each channel is a fixed path that passes through specific interior network nodes. Boundary nodes compute the route and identify the route with a label, while interior nodes just forward the packet along the route according to the label and do not compute the route (This is an advantageous characteristic for a high-speed router or switch of the Internet core).

Strictly speaking, MPLS is a mechanism between Layer 2 and Layer 3. It can use Layer 2 carriers such as ATM technology as well as provide label-switching and routing services to Layer 3 packets. Similar to DiffServ, MPLS is aggregation-based and so it exhibits good scalability but no coarse-grained service differentiation.

Similar to DiffServ, MPLS is aggregation-based and so it exhibits good scalability but no coarse-grained service differentiation.

Unlike the DiffServ prioritization mechanism, MPLS provides service differentiation through constraint routing. The QoS is realized when calculating routing with a QoS consideration such as bandwidth or delay requirement. Another difference between the two is that MPLS only works among network nodes and does not interface with workstations directly. Unlike IntServ and DiffServ, MPLS does not explicitly define service categories. A channel can bind with any QoS request and can provide any category of service guarantee, allowing it to support a wide range of services.

For carrier ISPs using ATM switches at the core of their networks, MPLS provide a more scalable and manageable networking solution than overlaying IP over an ATM network.

6.3.6. Application QoS Signaling – RSVP

Although applications can take advantage of priority-based mechanisms such as 802.1p network and DiffServ network by marking packets with different priority levels, they cannot identify explicit QoS such as bandwidth, latency and jitter requirements in priority-based mechanisms.

To express QoS explicitly, QoS signaling is necessary. RSVP presents the only QoS signaling between applications and the IP networks.

As introduced above, RSVP provides the highest level control ability of IP QoS and it is the only QoS signaling that connects the application and the network. From the perspective of a developer of an RT-DIS system, the RSVP is a critical component.

6.3.6.1. RSVP Entity

Both the host and the network must have entities capable of handing RSVP messages as shown Figure 6.6.

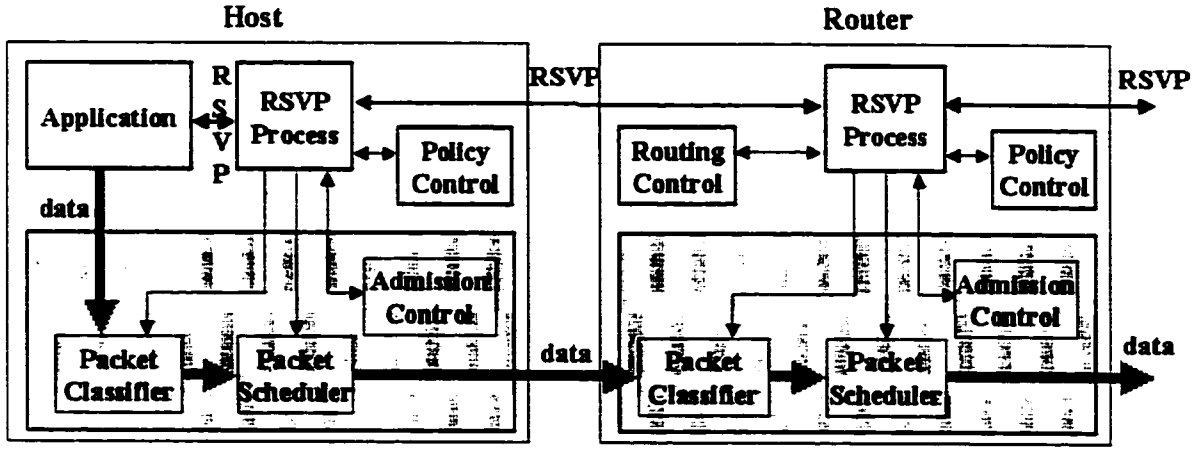


Figure 6.6 RSVP in Hosts and Routers [74]

- ***RSVP process*** – manages the reservation algorithm with the help of the other modules. It is responsible to ask the *policy control* and *admission control* for their permission to set up the reservation and if permission is granted the RSVP process sets parameters in a *packet classifier* and *packet scheduler* to obtain the desired QoS.
- ***Policy control*** – addresses the administrative issue of who is allowed to make reservations and what kind of QoS he may reserve. RSVP specification [74] defines a mechanism for transporting policy information along with reservations. Failing to get the policy's approval for the reservation results in the resources not being reserved and the requester getting an error message.
- ***Admission control*** – determines whether the node has sufficient resources available to meet the needs of the reservation request. If admission control fails at any node, the RSVP process returns an error message to the application that made the original

request.

- ***Packet classifier*** – If both admission control and policy control returns a positive decision, then RSVP process passes incoming data packets to a packet classifier that determines the QoS class for each packet. In addition, packet classifier also has a filter function. The filter is used to pass through a sub-flow rather than the whole flow when according to the request the flow is hierarchical and the flow receiver just requires a sub-flow.
- ***Packet scheduler*** – Packets are queued and prioritized as necessary in a packet scheduler that allocates resources for transmission on the particular link (it may also allocate some other system resources such as CPU time and/or buffers). This module is responsible for achieving the promised QoS.

6.3.6.2.Traffic representation

RSVP reserves network resources for each flow that is identified by the 5-tuple. The characteristics of a flow are described in RSVP with a "flow spec" and a "filter spec".

Flow Spec.

Flow spec describes traffic characteristics. *Tspec* is used to describe the traffic characteristics and describes the traffic generated by senders and the traffic for which receivers want to reserve resources.

The *Tspec* uses a token-bucket model (refer to Figure 6.7), which is usually a combination of a *token-bucket* and a *leaky bucket.*

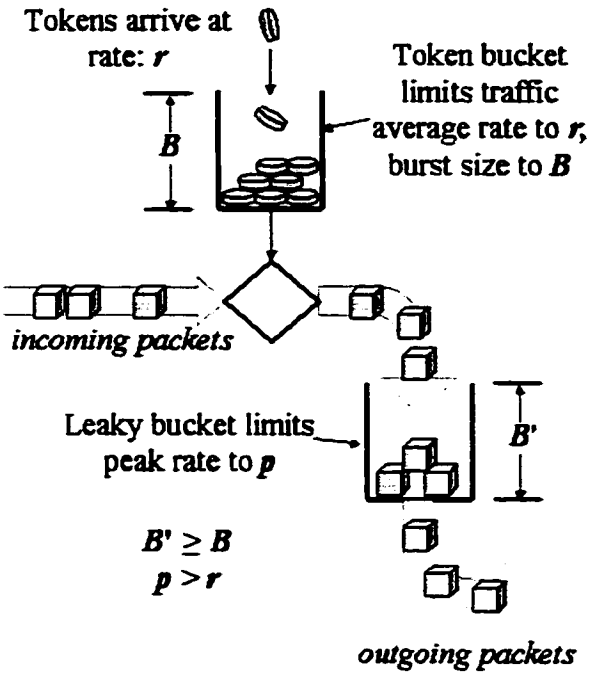


Figure 6.7 Token Bucket Model

- Token rate (r) – the continually sustainable bandwidth (bytes/second) requirements for a flow or in other words the average data rate;
- Token-bucket depth (*B*) – the extent to which the data rate can exceed the sustainable average for short periods of time;
- Peak Rate (*p*) – the maximum send rate (bytes/second);
- Minimum policed size (*m*) – the size of the smallest packet generated by the sender. All packets smaller than *m* are treated as size *m* and policed in order to reduce the overhead;
- Maximum packet size (*M*) – the size of the biggest packet generated by the sender. Any packet of larger size is considered out of spec and as a result may not receive QoS-controlled services.

Another structure (*Rspec*) is only used when the Guaranteed service model is chosen. Receivers use *Rspec* to specify the desired bandwidth, delay and jitter for a flow when senders do not use *Rspec*. *Rspec* is composed of two parameters:

- Reserved Rate (R) – signifies the desired bandwidth for the flow where R must be greater than or equal to r. Higher rates mean lower queuing delay.
- Slack term (S) – signifies delay jitter of the flow.

Filter Spec

The receiver defines which packets use the reserved resources and how they are used. Filter spec contains two parts: Flow ID and Reservation Style.

- Flow ID is used to identify a flow as an entity for resource reservation. In the general approach, *filter specs* may select arbitrary subsets of the packets in a given session but in the interest of minimum layer violation, the *filter spec* defined in RFC2205 [74] has a very restricted form containing the sender IP address and the UDP/TCP port number.
- Reservation Style concerns the treatment of reservation for the different senders within the same session. RSVP defines three reservation styles as shown in Table 6.5:

Reservation Style	Description
Fixed Filter (FF)	A receiver establishes a distinct reservation for each sender's flow
Shared Explicit (SE)	A receiver explicitly identifies the list of senders' flows and these flows share the reserved resources.
Wildcard Filter (WF)	A shared "pipe" is created among senders and receivers where the pipe size is the largest of the resource requests from all the receivers. All flows share this pipe from any sender to any receiver.

Table 6.5 Reservation Styles of RSVP

6.3.6.3.RSVP Operation Scenario

RSVP messages are sent hop-by-hop between RSVP-capable routers as "raw" IP data grams with protocol number 46 [74].

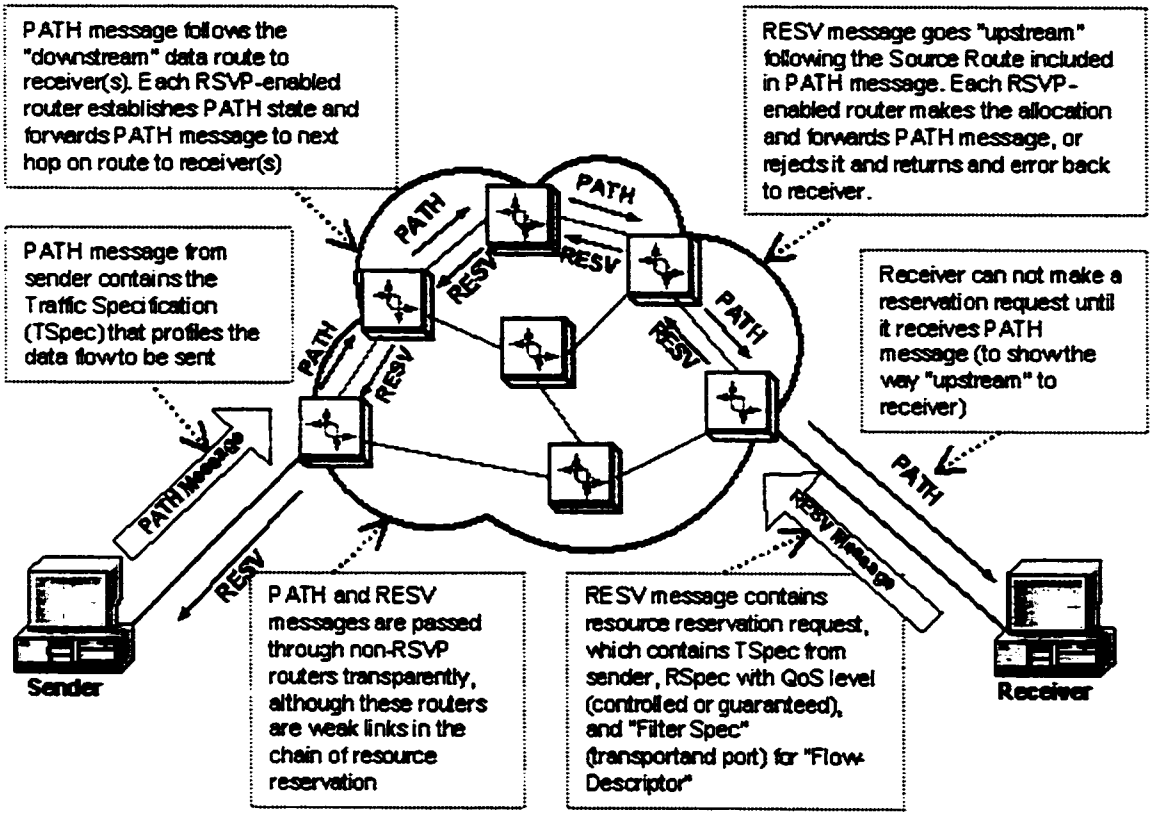


Figure 6.8 RSVP Operation [75]

Setup path

Like the phone signaling in a POTS (Plain Old Telephone System), RSVP is the signaling used to create connection between sender and receivers in a QoS-capable IP network. But unlike the signaling in POTS with fixed resources for each connection, RSVP has to reserve resources along the connection. A point worthy of mentioning is that receivers are in charge of reserving resources in RSVP despite the sender originating the signaling process.

Figure 6.8 shows the path setup scenario using RSVP protocol:

- RSVP sends a *PATH* message (containing the *Tspec* discussed in 6.3.6.2) from the sender to the receiver(s).
- Each RSVP-enabled router along the downstream route establishes a "path-state" that includes the previous source address of the *PATH* message (for the later upstream relaying of *RESV* messages). The router should also calculate the route for the next downstream hop using the routing protocol running on the router.
- When the *PATH* message reaches a receiver, receiver performs the following steps:
  - Check the *Tspec* to see what the sender can provide.
  - According to *Tspec*, choose a desired QoS and specify it using *Tspec* and *Rspec* (if Guaranteed service has been chosen)
  - Identify a filter spec that characterizes the packets for which the reservation is being made. - Send an *RESV* message upstream to sender with the *Tspec*, *Rspec* and *filter spec*.
- When each RSVP router along the upstream path receives the *RESV* message, it uses the admission control process to authenticate the request and allocate the necessary resources. If the request cannot be satisfied (due to lack of resources or authorization failure), the router returns an error back to the receiver. If accepted, the router sends the *RESV* upstream to the next router.
- When the last router receives the *RESV* and accepts the request, the path is setup.

Different venders may have different solutions for reserving resources in RSVP-capable routers. CISCO, the largest vender of routers, is using WFQ (Weighted Fair Queuing) and WRED (Weighted Random Early Detection) as the workhorse for RSVP and the set up of the packet classification and scheduling required for the reserved flows [76].

Refresh Soft State

RSVP takes a "soft state" approach to managing the reservation state in routers and hosts. RSVP's soft state is periodically refreshed by *PATH* and *RESV* messages. The soft state approach has advantages in a dynamic environment. If a route or a QoS request changes, related host or routers simply start sending revised *PATH* and/or *RESV* messages. The result will be an appropriate adjustment in the RSVP states in all routers along the path where unused states will time out if they are not explicitly torn down.

Data Transmission over Created Path

The RSVP protocol identifies packets associated with a particular traffic flow with the use of the *filter spec* (refer to 6.3.6.2). All data packets passing through routers are matched to a specific flow and forwarded using the reserved resources for the flow.

Teardown Path

Like the hand-on signaling in POTS, RSVP uses teardown message to explicitly dismiss the reserved path. There are two types of RSVP teardown message: *PATH_TEAR* and *RESV_TEAR* originated from sender and receiver respectively.

State Advertisement

When a sender issues a *PATH* message, another data structure named *Adspec* is also included. The *Adspec* provides a way of allowing the network nodes along a RSVP route to advertise their resource availability and transmission characteristics [79] such as available bandwidth, latency, etc. This information may help a receiver decide how much bandwidth and delay to ask for.

6.3.6.4.RSVP and Multicast

From the scenario introduced in 6.3.6.3, we can see that RSVP is receiver-initiated requesting resources in only one direction. For multicast, the reservation request needs to only travel to a point where it merges with another reservation for the same source stream. This receiver-oriented design is intended to accommodate large multicast groups and dynamic group memberships.

6.3.6.5.RSVP Implementation

According to a survey of RSVP implementations [77], most network device venders and those of the popular operating systems have supported RSVP since 1998 These include CISCO, Juniper, NortelNetworks, IBM, 3Com, HP, SUN, Digital, SGI, Intel, Microsoft, etc. RT-DIS system developers may be interested in the SUN's RAPI [78] and Microsoft's GQOS [79].

6.3.7. Integration Schemes

The QoS mechanisms introduced above illustrate that different approaches have different advantages and limitations. Fortunately these approaches are well complementary to one another and can be integrated to provide a *fine-grained scalable* QoS infrastructure.

The trend is becoming clear that the Internet backbone will adopt either DiffServ or MPLS or both to provide priority based QoS and QoS-based routing, while user networks including the ISP access network and enterprise/campus networks, will adopt IntServ-based fine-grained QoS. Seamless integration of these QoS mechanisms ought not to be

neglected. Some glue technologies address this particular requirement, some of which will be briefly outlined henceforth.

RSVP over Layer 2 802.1p Ethernet – SBM

Ethernet implements QoS through 802.1p, which is a priority-based mechanism. To support the reservation based IntServ/RSVP QoS, the Subnet Bandwidth Manager (SBM)[80] signaling scheme from the IETF ISSLL Working Group [81] is to be used.

RSVP falls short in LAN because RSVP-aware hosts and routers admit or reject flows based on availability of their private resources but not based on the availability of shared resources. The SBM solves this problem by enabling intelligent devices (hosts/routers/switches) that reside on the shared network to volunteer their services as 'brokers' for the shared network's resources. These devices automatically run an election protocol that results in the most suitable device(s) being appointed *designated* SBMs (DSBM). Hosts and routers that transmit through the LAN, first discover the closest DSBM and route RSVP messages through that device. Thus, the DSBM sees all messages that will affect resources in the shared subnet and provides admission control on behalf of the subnet.

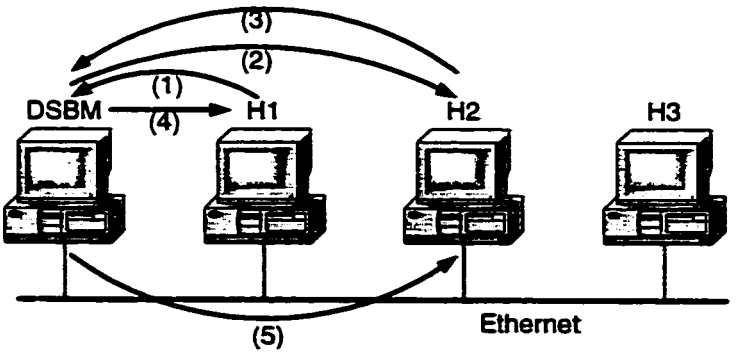


Figure 6.9 SBM Manage Subnet Resource

Figure 6.9 illustrates an example scenario of DSBM when H1 sends an RSVP *PATH* message to H2:

1. The RSVP aware application in H1 directs the PATH message to the DSBM.
2. The SBM forwards the message on to H2.
3. H2 responds with an RSVP RESV message to DSBM, indicating the traffic rate and the service type desired. The DSBM compares the requested rate against the provisioned limit (minus any resource granted in response to previously approved requests) and determines whether sufficient capacity remains in the high-priority queue to admit the request.
4. In this example, there is sufficient capacity and the request is admitted, hence the SBM forwards the RESV message on to H1.
5. As a result, H1 begins to mark traffic that is transmitted on the admitted flow with a high user-priority value.

RSVP over DiffServ – RSVP Aggregation

IETF released RFC3175 [82] (Aggregation of RSVP) for IPv4 and IPv6 Reservations in September 2001 to solve the scalability problem of RSVP. The idea is to use single RSVP reservation to aggregate other RSVP reservations across a transit routing region.

Before this RFC was released, Cisco IOS has prompted a new feature named *RSVP scalability enhancement* [83], which addressed the same problem.

When the *RSVP scalability enhancement* is enabled, a switch/router will process only RSVP messages for admission control and does not reserve resources for related RSVP flows. This feature can be used at boundary routers between an RSVP-enabled domain

and a non-RSVP-enabled domain. In the RSVP-enabled domain, RSVP works normally and reserves bandwidth for each flow. In the non-RSVP-enabled domain, routers forward RSVP messages without reserving resource or keeping the status of each flow. Instead, the RSVP flows can be classified with higher priority so they get precedence services compared to other non-RSVP flows. The RSVP flows could be classified based on their DSCP value or other Layer2/3/4 information such as 802.1p tag, IP address, protocol ID, port number, etc.

DiffServ over MPLS

IETF released an Internet draft for providing DiffServ through a MPLS network [84]. CISCO IOS new feature Diff-Serv-Aware Traffic Engineering (DS-TE) is designed to support DiffServ over MPLS network [85]. MPLS traffic engineering allows constraint-based routing of IP traffic. (The constraint based routing means to compute routes that are subject to multiple constraints, including QoS requirements, resource availability, policy, etc.) One of the constraints satisfied by CBR is the availability of required bandwidth over a selected path. Diff-Serv-aware traffic engineering extends MPLS traffic engineering to enable constraint-based routing of **guaranteed** traffic, which satisfies a more restrictive bandwidth constraint than that satisfied by CBR for regular traffic. This ability to satisfy a more restrictive bandwidth constraint translates into an ability to achieve higher QoS performance (in terms of delay, jitter or loss) for the guaranteed traffic (such as real-time voice, virtual IP leased line and bandwidth trading).

IP QoS over ATM QoS

ATM LANE version 2.0 Client provides up to 8 levels of QoS to upper layers. Each level is associated with its own set of User Network Interface call setup parameters. Call

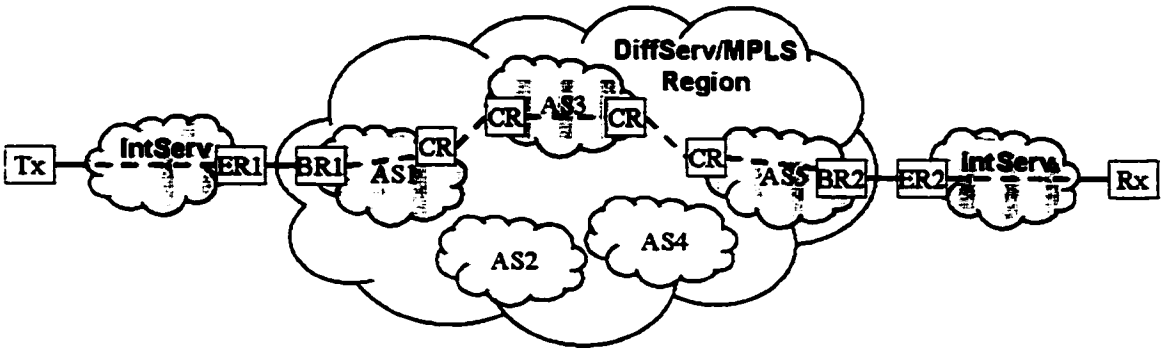
setup parameters will be defined or recommended by the IETF ISSLL WG as part of the mapping of the IETF IntServ models to ATM and IEEE networks [81].

6.3.8. Integrated Network QoS Infrastructure

RT-DIS demands both fine-grained QoS and scalability since:

- RT-DIS systems usually have a wide range of media and data types each with a wide range of its own QoS requirements. So the fine-grained QoS approach is a necessity.
- Large-scale simulation may involve a large number of computers widely distributed around the world, making scalability a necessary requirement as well.

The QoS mechanisms introduced above illustrate that different approaches have different advantages and limitations. With the glue technologies, these approaches are well complementary to one another and can be integrated to provide a *fine-grained scalable* QoS infrastructure.



Note: Tx: Sender; Rx: Receiver; ER: Edge Router; BR: Border Router; CR:Centre Router; AS: Autonomous System.

Figure 6.10 The QoS enabled Internet

Figure 6.10 shows the Internet infrastructure with QoS-capability.

The Internet interconnects multiple Autonomous Systems (AS) maintained and operated by different Network Service Providers (NAP). User networks access the Internet by connecting to one or more NAPs.

As mentioned in 6.3.7, the Internet backbone will adopt either DiffServ or MPLS or both to provide priority based QoS and QoS-based routing, while user networks including the ISP access network, enterprise/campus network, will adopt IntServ and RSVP to implement accurate and flexible QoS control, refer to the verification at Chapter 9. Between a user network and a NAP, or between two NAPs, a Service Level Agreement (SLA) is signed that specifies the traffic profile and the forward service.

The user network's administrator should map different users and their applications to the service categories in the SLA through provisioning and configuration. Each category has specific network resources and can support a specific capacity of traffic.

To avoid overloading these service categories, explicit application-network signaling is required. When the service category becomes saturated, new application requests for that category are refused. With the signaling between application and network, the network will never be overloaded and the QoS of the application admitted by the network is guaranteed. RSVP is the only signaling protocol satisfying this requirement.

On the other hand, DiffServ and MPLS domains that may be used in different ASs can not understand RSVP signaling, so the signal mapping is implemented at the boundary between user network and DiffServ or MPLS domains.

There already have been several QoS trial networks, such as Abilene UCAID [86] and vBNS [87] in the USA, and QTP [88] in Europe. These trial networks are evaluating and

testing IntServ-RSVP, DiffServ, MPLS protocols and verifying the interoperation among them.

Chapter 7
HLA Real-Time Extension

7.1. Real-time is Necessary and Possible

HLA does not currently address RT-DIS. The HLA compliant simulations can only get best-effort services from the underlying operating system and network, clearly not sufficient for RT-DIS requirements. Hence a real-time extension is needed for HLA through which the simulation can identify their response requirements, resource requirements and get the services they demand.

RT-DIS demands end-to-end predictability. This means a real-time response and predictable behavior from both the end systems and the network cloud. With RTOS mechanisms such as preemptive priority multitasking and priority message queuing and network QoS solutions such as Integrated Service (IntServ), Differentiated Service (DiffServ) and Multiple Protocol Label Switching (MPLS), end-to-end predictability is possible.

7.2. Requirements, Experiences, Trends of RT-DIS

7.2.1. Requirements

As early as 1994, the IETF Large Scale Multicast Applications (LSMA) working group has released RFC1667 [89] that summarizes the DIS requirements as *real-time response*, *multicasting* and *resource reservation*. In 1999, LSMA released RFC2502 [90] contributing to the requirements for five transportation types in DIS and especially the reliable multicast.

7.2.2. Experiences

Chassot's team reports their DIS-ATM project [91] to specify and perform the QoS required by a DIS system in a WAN environment using the RSVP and ATM. They proved that the initial traffic generated by a given DIS application conforms the "a priori" estimate *Tspec*, and the QoS would not be ensured without resource control.

Ferrall's team reports their QoS enabled STOW'97 [92]. The STOW'97 runs over a modified RTI named RTI-s [93], which uses RSVP-over-ATM technology to guarantee QoS.

These ATM based network level trials have limitations at wide deployment due to the high cost issue of ATM networks. User networks may not get ATM access services from their local regions.

Myjak's team reports their experience of implementing a Java Real-Time RTI [94] where real-time stream (audio and video) features are added. They also mention that RTI needs a multi-thread solution to minimize communication latency.

Greenhalgh's team reports their video interaction over a collaborative virtual environment with dynamic QoS control [95]. They divide the visual field of a user into several sub-zones with each different sub-zone associated with a different QoS.

We have worked out an HLA-compliant real-time stream (audio/video) solution, which uses the HLA specified interface to control and transfer audio/video packets.

These application level trials are all based on best-effort service of the underlying network, so they cannot really satisfy real-time requirements.

To solve the problem radically, the best way is to involve the predictability technologies into HLA and RTI in order to overcome the limitations of current HLA and RTI.

7.2.3. History and Trend of DIS Network

In the mid-1990's, the network topology used in Modeling & Simulation (M&S) exercises was typically that of a high-speed cross-country and trans-oceanic backbone running between wideband packet switches with tail circuits running from these packet switches to various nearby sites [89]. The network was using ST-II (an experimental Internet Stream Protocol) to provide resource reservation and multicast. Within the last few years, these networks have opted to use RSVP.

Due to cost constraints, the future implementation of WAN for the Department of Defense (DoD) will not be proprietary and closed. DoD will purchase the required bandwidth for a given network from a "service provider" that may be another government agency or a commercial one [96]. In this shared environment, priority and policy based reservation mechanisms will be necessary to maintain QoS.

We have found some DIS applications using RSVP signaling over private ATM networks but we have never found any effort for making a practical RT-DIS solution for the shared Internet based on the QoS mechanisms.

7.3. HLA Limitations for RT-DIS

From the real-time perspective of DIS, HLA has the following two major limitations:

- Lack of QoS specification interfaces – HLA does not provide interfaces to specify end-to-end prediction requirements.
- Lack of transportation types – HLA supports only two transportation types: *reliable* and *best effort*, which is not sufficient for DIS.

7.4. QoS Representation

With respect to real-time response, the entire DIS system must accommodate delays and delay variances compatible with human interaction times in order to preserve an accurate order of events and provide a realistic simulation [89]. More specifically:

1. 100 msec for exercises containing simulated units whose interactions are tightly coupled; 300 msec for exercises whose interaction are not tightly coupled; all latency requirements are expected to remain under a few hundred msec in all cases.
2. Delay variance (jitter) should be low enough that smoothing by buffering the data stream at the receiving simulator does not cause the previously stated latency specifications to be exceeded.

3. Reliability of the best-effort data gram delivery service should be such that 98% of all data grams are delivered to all intended destination sites with missing data grams randomly distributed.

All these requirements can be satisfied by the combination of RTOS and network QoS technologies.

7.4.1. Application Level Representation

Parameter	Description
Data Unit Size (Bytes)	Size of an object attribute or an interaction parameter
Average Unit Rate (Units/Sec)	Average rate of data units produced by a federate
Maximum Unit Rate (Units/Sec)	Maximum rate of data units produced by a federate
Burst Duration (Millisecond)	Continuous time of a data burst at max. unit rate
Maximum Delay (Millisecond)	Time consumed between message generating and consuming
Loss Ratio (Bytes / Million Bytes)	Acceptable packet loss during transportation
Priority Level (1~32767)	The importance level of the data unit in a federation
Security Level	The security level of the data unit in a federation

Table 7.1 Application Level QoS Parameters

Application level representation of QoS requirements should be easy to understand for federate developers. Based on this consideration, we abstract the application level QoS representation of RT-DIS as shown in Table 7.1.

Delay variance is not necessary for this matrix. It can be controlled through the maximum delay and eliminated through an application level reconstruction buffer.

7.4.2. System Level Representation

It should be mentioned that the parameters listed in Table 7.1 are not exactly the same as the parameters used by the underlying RTOS and network QoS mechanisms.

Regarding the end system, operating systems reserve computing capability mainly via a *priority* mechanism. An OS usually commits the services to higher priority tasks with lower delay.

With regard to the network, RSVP reserves communication capabilities according to traffic characteristics represented using a *token-bucket* model (refer to 6.3.6.2 and Figure 6.7).

In this combination, the *token bucket rate* is set to the *average rate* at which an application source generates traffic. The *leaky bucket rate* is set to the *peak rate* at which an application generates traffic (the burst rate).

As mentioned in 6.3.5, in the IntServ network, RSVP can invoke two categories of service: Controlled-Load Service (CLS) and Guaranteed Service (GS). The controlled-load service provides a closely equivalent unloaded best-effort service and the guaranteed service commits to provide a delay bounded and loss free service with a higher cost. Neither CLS nor GS can accept parameters such as delay or loss since current IP routing protocols calculate routes without the consideration of delay and loss. Alternatively, CLS and GS can use the RSVP *Adspec* message to measure and estimate delay and bandwidth

[97]. The measured delay and bandwidth can be used to judge whether a QoS requirement could be satisfied.

System and Network Parameters	Application Level Parameters
RSVP token rate (*r*) (Bytes/second)	Data Unit Size x Average Unit Rate
RSVP peak rate (*p*) (Bytes)	Data Unit Size x Maximum Unit Rate
RSVP bucket depth (*B*) (Bytes)	Data Unit Size x Maximum Unit Rate x Burst Duration
Operating System Priority Level	Priority Level
Network Service Category (GS or CLS)	Maximum Delay and Loss Rate
Different cryptography algorithms	Security Level

Table 7.2 Map to Operating System and Network

With the above information in mind, it is reasonable to map the application level QoS representations to system level QoS representations, as shown in Table 7.2.

The priority level parameter can be used as a reference when an OS assigns priority to the *attribute/interaction* updating/sending related task.

The *maximum delay* and the *loss rate* are used to choose a network service category in addition to the RSVP *Adspec* measurement of network delay and loss rate.

More work is needed in the area of sophisticated mapping between the application level QoS representation and the system level QoS representation. At first, this mapping will be more empirical.

7.5. Transportation Type

In the DIS/HLA community, five different transportation types are necessary [90]:

1. Reliable unicast of control information between individual RTI components.
2. Best-effort low-latency multicast of object attributes that often change continuously (For example position of mobile objects);
3. Low-latency reliable multicast of object attributes that do not change continuously but may change at arbitrary times during the simulation (For example object appearance);
4. Reliable but not necessarily real-time multicast distribution of supporting bulk data such as terrain databases and object enumeration;
5. Low-latency reliable unicast of occasional data among arbitrary members of the multicast group (For example, object collisions);

Current IP services can meet requirement *(1)* by TCP based unicast and *(2)* by UDP based multicast. As for *(3)(4)(5)* requirements, which involve reliable multicast, results to date indicate that there will not be a "one size fits all" reliable multicast protocol [98]. The most difficult challenge is providing scalable feedback from receivers with congestion control.

Fortunately, if we look at the problem from a different perspective, we can discover that IP QoS mechanisms can easily satisfy the requirements of the reliable multicast.

7.5.1. QoS Enabled Reliable Multicast

There are two reasons for packet loss: buffer overflows or bit errors. The probability of bit errors is very low on most networks, making the buffer overflow as the primary reason for packet loss.

The IntServ GS category can guarantee non-occurrence of buffer overflow [72]. A reliable multicast can be realized by implementing the following mechanisms:

- Create a multicast group using IP multicast protocols such as IGMP, PIM-SM, PIM-DM, etc.
- Reserve network resources for the multicast group using the RSVP protocol. As an ultimate case, the reserved GS for the multicast group can guarantee lack of any loss due to buffer overflow.
- Implement a negative acknowledge (NACK) [98] protocol to prevent loss due to bit error. The probability of bit error is very low in networks and so occasional bit errors will not cause a congestion problem.

7.5.2. Unicast between Multicast Group Members

As the simplest solution of the fifth transportation type discussed in section 7.5, the unicast can be implemented with identified messages in a multicast group.

Assume the reliable multicast group is created through steps given in section 7.5.1 and two group members needs a unicast channel between them. When a member sends out a packet, it should mark the packet as *public* or *private*. For the *private* packet, a *receiver ID* should be included into the packet. Either a *public* or a *private* packet is sent to the multicast group and all receivers are able to receive them. Receivers will initially check

the received packet to see whether it is *public* or *private*. For a *public* packet, all receivers process it. For a *private* packet, receivers check whether its *receiver ID* matches with its own. Only the receiver with the matched *ID* will process the packet and other receivers discard it. Since this transportation type is just used for occasional data, the solution is reasonable.

7.6. Extension to HLA Specifications

Based on the DIS requirements given in RFC1667 [89] and RFC2502 [90] and current predictability technologies of networks and operating systems, HLA could and should be extended to support RT-DIS.

Fortunately, due to its sophisticated considerations from the onset, HLA is capable of supporting RT-DIS with a minimum amount of extension. On the other hand, to be compliant with the standard HLA based DIS system and for the sake of simplicity of the development of a non-real-time DIS system, all these extensions should be made optional.

7.6.1. HLA Framework and Rules

For HLA framework and rules, only one federate rule needs to be added to the section 6 of [12]:

Federates shall be able to manage QoS in a way that will allow them to make efficient data exchange with other federates of a federation.

7.6.2. HLA Object Model Template (OMT)

In the case of OMT, three tables need to be extended: *attribute table*, *parameter table* and *transportation table* [14].

7.6.2.1. Attribute Table

The current attribute table has ten columns: object, attribute, data type, update type, update condition, D/A (divested or acquired ownership), P/S (publish/subscribe), dimensions, transportation type, order (timestamp order/ receive order).

By analyzing the seven QoS parameters listed in Table 7.1, the *data unit size* can be calculated from the *data type* of the *attribute table* and the *data type table*, while the users via OMT should provide the other seven parameters. These parameters are: *average unit rate*, *maximum unit rate*, *burst duration*, *maximum delay*, *loss ratio*, *priority level* and *security level*. In addition, one more parameter *aggregatable* is required to identify whether the attribute needs a private channel or can join in a shared channel. These eight parameters compose a QoS-tuple. A QoS-tuple adds eight extra columns to an attribute table as shown in Table 7.3

Attribute Table												
Object	*Attribute*	*data type*	....	*order*	*average rate*	*max. rate*	*burst duration*	*max. delay*	*loss ratio*	*priority*	*aggre-gatable*	*security*
Car	Model	enum	...	...	–	–	–	–	–	–	–	–
	Color	enum	...	...	–	–	–	–	–	–	–	–
	Position Accurate	array	...	...	20	30	300	200	300	16896	No	Low
	Position Coarse	array	...	...	10	15	160	400	1000	16384	Yes	Low
	Position Faraway	array	...	...	2	5	60	1000	4000	15884	Yes	Low

Notes: The units of QoS parameters refer to Table 7.1.

Table 7.3 Example of Attribute Table with QoS extension

The QoS-tuple is optional in OMT. If an attribute associates no QoS-tuple, such as the Model and Color attributes in Table 5, the best-effort service is used for it. If an attribute associates a QoS-tuple, such as the Position attributes, RTI is going to invoke QoS for its transportation. If an attribute can be updated using different QoS when supporting layered QoS [99] or adaptive QoS [100], such as the *PositionAccurate*, *PositionCoarse*, *PositionFaraway* attributes, it describes the same characteristic (*position*) of the *Car* but with a different QoS requirement. The attribute should be divided and represented through multiple attributes, each of which associating with a different QoS-tuple.

Although 256 system priority levels are sufficient even for the most complex systems [40], for further extensibility and data type simplicity, HLA priority representation is recommended to refer to real-time CORBA's priority which uses a short integer with scope [0,*32767*] where numerically higher values mean higher priority [101].

When mapping HLA priority to an underlying operating system that does not have so many priority levels, the priority sharing is necessary. For example, VxWorks has *256* priority levels; mapping HLA priority to VxWorks causes *32768/256=128* HLA priority levels to share one VxWorks priority level. In order to map two attributes to different OS priorities, their HLA priorities in OMT should be distinct carefully.

Priority could also be used to identify the attribute's priority in a priority-based QoS network such as a DiffServ network. In this situation, other items of the QoS-tuple may not be needed. RTI can use only the priority field to map each attribute's priority to a specific network service category.

7.6.2.2. Parameter Table

The QoS-tuple is also used to describe a parameter's QoS requirement. A QoS-tuple is associated with all parameters of an interaction since parameters belonging to the same interaction are always sent together. (Table 7.4)

Parameter Table												
Interaction	*Para.*	*data type*	....	*order*	*average rate*	*max. rate*	*burst duration*	*max. delay*	*loss ratio*	*priority*	*aggre-gatable*	*security*
Video LayerOne	H.Res	int	...	...	32K	64K	200	300	3000	16896	No	High
	V.Res	int										
	Codec	enum										
	Data	array										
Video LayerTwo	H.Res	int	...	...	128K	160K	200	300	3000	15884	No	High
	V.Res	int										
	Codec	enum										
	Data	array										

Notes: H.Res: horizontal Resolution, V.Res: Vertical Resolution

Table 7.4 Example of Parameter Table with QoS extension

Similar to the attribute table, an interaction supporting a different QoS should be divided and represented via different interactions.

7.6.2.3. Transportation Table

The "*4.10.3 transportation table inclusion criteria*" of the OMT [14] only identifies two types: *HLAreliable (TCP)* and *HLAbestEffort (UDP)*. According to section 7.5 of this paper, five new types should be supported:

- Type1: ReliableUnicast;
- Type2: BestEffortMulticast;
- Type3: LowLatencyReliableMulticast;
- Type4: BulkVolumeReliableMulticast;
- Type5: LowLatencyReliableUnicastInMulticastGroup.

The *HLAreliable* type involves TCP reliable unicast and TCP based emulated reliable multicast. This is not equivalent to anyone of the 5 new types. The *HLAbestEffort* type involves UDP based unicast and multicast, also not equivalent to any one of the 5 new types.

Although the new five types can cover the old two types, for backward compatibility, the transportation table should accept all seven transportation types.

7.6.3. HLA Interface Specification

The real-time extension to the HLA Interface Specification involves the following services:

7.6.3.1. Publish/Subscribe Services

Federates use the *publish* services to declare their intention of generating information and the *subscribe* services to declare their intention of receiving information.

There is no requirement for modifying the input/output arguments of these interfaces. Only some QoS related statements should be added.

In standard HLA, the *publish/subscribe* services make a judgment of interest overlap among data suppliers (*publishers*) and data consumers (*subscribers*) of *attributes/parameters*. When there is an overlap, a unicast channel or a multicast group is created among *publishers* and *subscribers*.

With the real-time extension, publish/subscribe services cause the same result but with the enhanced rules of *transportation types*, *aggregatable* field and other fields in the QoS-tuple as outlined in the attribute/parameter tables.

If all attributes/parameters listed in OMT tables do not have an associated QoS-tuple and all transportation types are either *HLAreliable* or *HLAbestEffort*, this indicates that the federation and federates are designed for standard HLA without the real-time extension, hence making this model backward compliant to the standard HLA. We call this model the *normal* model.

If at least one attribute/interaction identifies a QoS-tuple or at least one attribute/interaction chooses a transportation type from the five new types, the federation or federates are supposed to be designed for real-time HLA. We call this the *advanced* model.

The *advanced* model allows an attribute/interaction to choose a new transportation type but not identify a QoS-tuple since some of the new transportation types already imply QoS support (refer to section 7.5).

Following the standard HLA style, a real-time extension should leave the implementation detail to the RTI. RTI should assure that the final unicast channels and multicast groups satisfy the QoS requirements identified in the OMT tables.

7.6.3.2. Unpublish/Unsubscribe Services

Federates use *unpublish* services to declare their intention of stopping the generation of information and *unsubscribe* services to declare their intention of stopping the reception of information.

There is no requirement for modifying input/output arguments of these interfaces as well.

With the real-time extension, these services should be able to resign the *attribute/interaction* from the QoS capable multicast groups or unicast channels created by the *publish/subscribe services.*

7.6.3.3. Change Transportation Type

The interface *Change Attribute/Interaction Transportation Type* services change transportation types for attribute/interaction during run-time between *HLAreliable* and *HLAbestEffort.*

With the real-time extension, two situations arise:

- In the *normal* model, federates invoke these services to change between the two types of standard HLA: *HLAreliable* and *HLAbestEffort.*
- In the *advanced* model, federates invoke these services to change among the five new types of real-time HLA defined in section 7.6.2.3

7.6.3.4. Set/Get Priority

Two QoS related support services should be added to HLA for its real-time extension:

The *Set Priority* service is invoked by a federate or RTI to set the priority of a thread in a multi-threaded environment. The priority should be an HLA priority defined in section 7.6.2. The RTI is in charge of converting the HLA priority to native OS priority and setting the priority to the thread.

The *Get Priority* service is invoked by a federate or RTI to get the priority of a thread. The RTI converts a native OS thread priority to HLA priority and returns an HLA priority to the federate or RTI.

7.6.4. Advantages of the Extension

The real-time extension to HLA mainly involves OMT modifications to support QoS representation and new transportation types. The HLA interface specification [12] is not affected at input/output argument level since it can leave the QoS implementation to RTI, resulting in the following advantages:

- Backward compatibility – RTI compliant to this extension can support both standard HLA based simulation and real-time HLA based simulation.
- Federate portability – A federate developed for a standard HLA compliant RTI can be ported to an extended HLA compliant RTI with minimum modifications of the federate code. The porting could be as simple as just adding QoS-tuples to OMT attribute/parameter tables and modifying transportation types of the attributes/interaction.

7.7. Discussion – DDM and QoS

HLA reduces both the transmission and reception of irrelevant data via the data distribution management (DDM) services. This extension can also be used with the DDM to make the layered QoS and the adaptive QoS more positive and flexible.

For example, in a virtual shopping mall system [21], there are many stores in the mall and some of them use a TV that can play a video program. From a customer's perspective, when there is a long distance between him and the TV, he does not expect to see a clear video. As he comes near the TV, he expects to see it more and more clearly. With DDM and Layered QoS, this is easy to realize.

Assume both the customer and the TV are federates. The video is divided into two layers using layered QoS technology and is transferred using an interaction in the federation. The distance between the customer and the TV is divided into three regions shown in Figure 7.1. In the parameter table of OMT, each video layer is represented using an interaction associated with a QoS-tuple (Refer to Table 7.4).

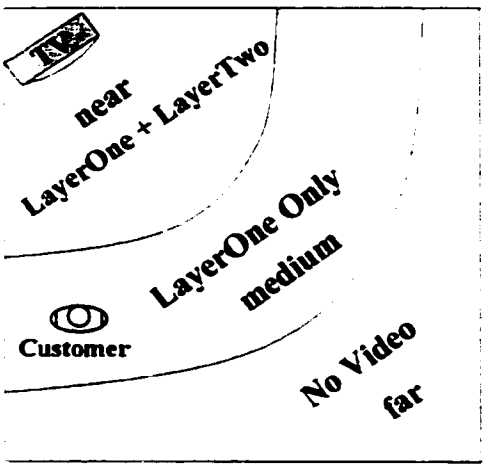


Figure 7.1 DDM & QoS

After the federations startup, the TV federate publishes the video interactions through the interface *publish interaction with region*. The customer federate subscribes to both the *LayerOne* and the *LayerTwo* interactions via the interface *subscribe interaction class with region*. When the customer-federate is in the *far region*, it cannot receive any video interaction. When the customer-federate moves to the *medium region*, it can receive *LayerOne* low quality video interactions. When the customer-federate moves into the *near region*, it can receive both *LayerOne* and *LayerTwo* video interaction, which can combine to generate high quality video.

One channel for *LayerOne* interaction and one channel for *LayerTwo* interaction should be created with reserved resources including network bandwidth or end system memory.

The channels can be created at startup. When the customer-federate crosses the region boundary. it switches from the channel associated with the old region to the channel associated with the new region. The advantage is no delay and "cut off" when it crosses the region boundary. The disadvantage is resource waste.

The channels can also be created "on the fly". When the customer-federate crosses the region boundary, it creates the new channels for the new region. Creating new channels may cause more delay since QoS signaling is involved. And it is possible that the system resources are not enough for creating the new channel at which point "cut off" occurs.

The implementations of layered QoS and adaptive QoS with this extension need further exploration.

Chapter 8
Real-Time RTI Architecture Proposal

8.1. RTI Prototypes

RTI works as a middleware and provides services to all federates in a simulation. There are some RTIs available:

- STOW RTI-s of MIT Lincoln Laboratory [102,103,104,105],
- MÄK-RTI of MÄK Technologies [106],
- Light-Weight RTI of George Mason University [107],
- Java Real-Time RTI of the Virtual Workshop Inc [94],
- Synchronous Parallel SPEEDES-RTI of Metron Incorporated [108],
- RTI-NG of SAIC [109].

Among them, the RTI-NG of SAIC, sponsored by the DMSO (Defense Modeling and Simulation Office) of the U.S. DoD (Department of Defense) is the most dominant. Although the RTI-NG can be freely downloaded, it is not open source software [110].

Stephen T. Bachinsky was the supervisor for RTI-NG. He outlined the design constructs of RTI-NG in two papers [111,112], containing enough information for us to identify its possibilities for real-time extension.

8.2. RTI-NG Limitations

RTI-NG is a state-of-the-art implementation of HLA. But from the real-time perspective, it still has following limitations:

- Lack of QoS enforcement – RTI-NG does not specify how to treat events differently depending on priority. It does not use predictability improvement mechanisms provided by OSs and network QoS mechanisms.
- Lack of scalability – RTI-NG reaches its performance ceiling when seven ~ eight federates run on the same machine. Adding a machine to a federation may have a greater negative impact on execution time than adding a federate [113]. We have not found the statistical data of RTI-NG's capability of supporting objects, but its previous version RTI1.3v5 can support less than 100 entities with real-time response requirements [114].
- Lack of resource management – RTI-NG uses a two-threaded model and a coarse-grained locking mechanism [112], which is not efficient enough for resource management.

8.3. Real-Time RTI Architecture

RT-DIS demands end-to-end predictability potentially satisfied through RTOS mechanisms and network QoS. A real-time RTI architecture based on this idea is presented in Figure 8.1.

In this architecture, some RTOS mechanisms are explicitly involved to insure predictability of any task processing at the end system including priority-based multithread and thread-pool and priority based message queues. One thing worthy of

mention is that many other RTOS mechanisms are also necessary for strict predictability, while such mechanisms as priority semaphore, priority DPC, priority inheritance, fast context switch, etc, can not be controlled by programmers explicitly.

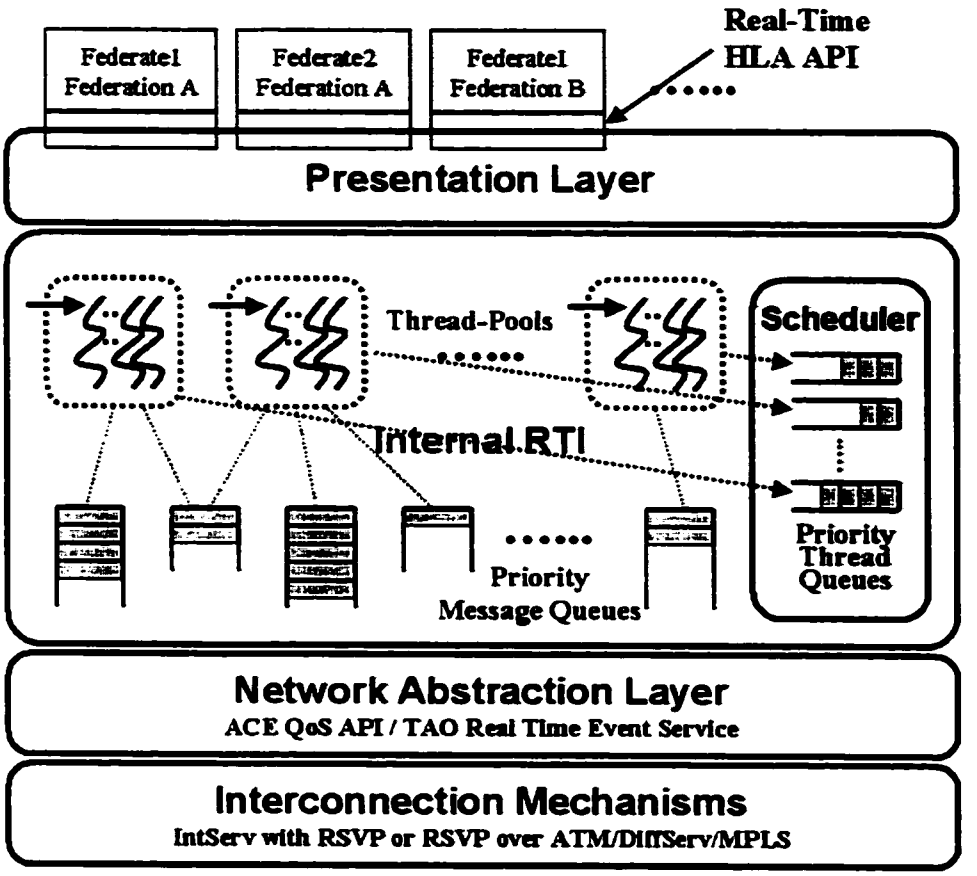


Figure 8.1 Real-Time RTI Architecture

RT-RTI controls the QoS network through RSVP signaling, so real-time information exchange through the network is guaranteed among federates. RT-RTI does not need to care about how it could be guaranteed but leaves this as the responsibility of the network service provider. RT-RTI needs only to parse the federate network requirements and map them to network resources.

8.4. Multi-thread and Thread-pool for RTI

RT-RTI must be Multi-thread enabled in order to improve the predictability, scalability and I/O throughput.

Multi-thread is not only the basic mechanism for parallel processing in a multi-processor platform and the fundamental part for preemptive priority approaches, but also a key mechanism to improve I/O throughput.

RTI has been using a single process approach until RTI-NG, which adds a two-threads-model to isolate the *downcall* from *upcall* [111]. This approach has been proved to exhibit scalability [113]. To satisfy the U.S. DoD's goal of building DIS systems with up to 100,000 simulated objects [90], a fully multi-threaded approach is necessary.

Threads in RT-RTI should be managed under a thread-pool mechanism in order to further improve its performance.

A thread-pool is a collection of N threads that are used repeatedly to run small tasks. The N threads are usually created at application initialization and destroyed at application termination. When a task needs a process, it is handed off to the thread-pool and the thread-pool runs it inside one of its N threads. If all N threads are busy processing other tasks, the task waits for one to become available.

It has long been known that one of the ways to improve the performance of a software application is to reduce the overall number of objects created and destroyed [115] (The data in Table 6.3 show that thread creation has terrible latency compare to other thread operations). The thread-pool improves performance by avoiding thread creation and destruction. Since the thread-pool is a successful resource reservation technology for end

systems used for improving performance in a multi-thread environment, the standard specification of Real-Time CORBA has even identified the thread-pool as a necessary mechanism of its implementation [101]. RTI-NG has not supported a thread-pool, but it has been considering its use [111].

Some tests illustrate an important point: as soon as the number of jobs reaches around 1000, the thread pool improves the performance by a factor of 10 [42], refer to Table 8.1.

Number of Jobs	Without Pool (Approx. ms)	With Pool (Approx. ms)
<15	17	25
100	64	30
1,000	603	47
50,000	24100	1980

Notes: Sun Ultra-2 with two 400MHz UltaSPARC-II processors and 512MB of RAM, JDK1.2.2.

Table 8.1 Thread Pool Performance Improvement [42]

Obviously, to support the 100,000 simulated objects and to improve the response performance, the thread-pool is a necessary mechanism for real-time RTI.

The characteristics of a thread-pool can be specified through parameters in the RTI Initialization Data (RID) file. It will be more flexible if RTI could provide APIs through which a federate could configure a thread-pool at run-time.

During simulation run-time, there will be two kinds of threads: federate explicitly created threads and RTI implicitly created threads. All these threads should share the same priority scope and the thread-pool.

During its initialization, RTI creates a thread-pool according to the configuration information retrieved from the RID file including:

- Minimum and maximum numbers of threads (*minThreads*, *maxThreads*). These two parameters identify how many system resources are allocated.
- Minimum and maximum priority levels of threads (*minPriority*, *maxPriority*).
- Default priority level of thread in the thread pool (*defPriority*). The threads belonging to the thread pool keep *defPriority* as their priority level before a federate or RTI calls *SetPriorit()* to change it.
- This thread framework can support many working models including:
- One thread per federate – backward compliant to RTI-NG single thread model with *minThreads* = *maxThreads* = *1*.
- Two threads per federate – backward compliant to RTI-NG *downcall/upcall* two threads model, with *minThreads* = *maxThreads* = *2*.
- Multiple threads per federate – fully re-entrant multi-thread model with *maxThreads* > *minThreads* > *2*.

Some thread-pool implementations [116] provide a valuable reference for implementing a thread-pool in real-time RTI.

At run-time, the RTI can submit to a thread-pool any task such as presentation manager, object manager, data distribution manager, queues manager, time manager, etc [112]. On the other hand, when a federate needs a thread for its task, it adds the task to the thread-pool. The thread-pool then assigns the task a null thread.

8.5. Preemptive Priority Scheduling

A complex federate may also be implemented via multiple threads, resulting in multiple federate threads and multiple RTI threads running on the same platform. Furthermore, it is possible to run multiple federates on one computer causing multiple thread-pools running in an OS.

All the threads in these thread-pools are marked with HLA priority levels. The RTI and the federate can change the priority of the thread using an HLA priority level. RTI is in charge of checking the priority scope of native OS and mapping the HLA priority level to the native OS priority level. The OS services all threads running in it depending on their priority and independent of their thread-pools.

To process important tasks first, RTI should use the preemptive priority mechanism of the underlying OS. By assigning higher priority level to critical tasks and delay sensitive tasks, they would experience shorter delay than others lower priority tasks.

8.6. Synchronization

The primary thread synchronization model makes use of a *mutex* for protection and *condition variables* for communication. A mutex allows one thread to *lock* shared data while using it so that other threads cannot accidentally interfere. A condition variable allows a thread to wait for shared data to reach some desired state.

These mechanisms are well supported in all modern OSs. RT-RTI may use these OSs provided that synchronization mechanisms for insuring inter-thread communication and synchronization are available. A potential DIS feature in this regard is protecting

Federate databases and Object databases that may be accessed by multiple threads simultaneously.

RT-RTI can also use other OS supported synchronization mechanisms such as semaphores and pipes.

8.7. Communication Assurance

RT-RTI should retrieve and interpret the QoS requirements of attributes and parameters via OMT and create private or aggregate communication channels via RSVP for quality assured communication among federates.

RTI-NG is currently built over ACE/TAO [117]. ACE is an object-oriented framework that implements core concurrency and distribution patterns for communication software. TAO is an implementation of real-time CORBA built over ACE. RTI-NG is using TAO's real-time CORBA event services to provide flexible asynchronous communication among distributed objects [118].

ACE already has a QoS API (AQoSA) which provides a portable C++ encapsulation of two separate implementations of RSVP: the GQOS implementation on Windows 2000 and the RSVP API (RAPI) implementation on UNIX.

TAO's real-time event services encapsulate real-time features of underlying OSs, but it may have limitations in supporting the new transportation types. As an alternative of upgrading TAO to support the new transportation types, RTI can build an event service directly from the AQoSA. RTI-NG should add some code to interpret and map each attribute/parameter's QoS requirements to ACE/TAO's QoS APIs.

8.7.1. Publish/Subscribe Implementation

In 7.6.3.1, we mentioned that HLA should leave the QoS implementation to RT-RTI. Here we give some suggestions for RT-RTI to invoke underlying QoS mechanisms according to an attribute/interaction's requirements.

When an attribute/interaction is published and subscribed by federates, RT-RTI should implement the following operations to create communication:

1. Calculate the overlap between the publisher and the subscriber(s). (Assume there is an overlap).
2. Check the transportation type of the attribute in OMT. For *Type1*, RTI creates a TCP based unicast channel and connects the publisher and the subscriber, then goes to step 4 of 8.7.1; For *Type2~5*, RT-RTI creates a UDP based multicast group and joins the publisher and the subscribers to the group.
3. Evaluate network performance and assign service categories. When the multicast group is created, the topology of the group is identified. RT-RTI invokes an RSVP *Adspec* message to measure the performance of the multicast group including delay, available bandwidth, etc. According to the result of the measurements, RT-RTI should assign GS or CLS services to the multicast group. When the available bandwidth is very large, meaning loss due to buffer overflow is almost impossible, the CLS service is good enough for a reliable multicast requirement; otherwise, the reliable multicast group should be assigned with GS services.
4. Retrieve the QoS-tuple and reserve resources for objects and interactions along the path. RT-RTI reads the QoS-tuple of the attribute from OMT and invokes RSVP

PATH/RESV messages to reserve resources according to the information of the QoS-tuple.

After these operations, the unicast channels and multicast groups can be created with required QoS assurance.

8.7.2. Flow-based and Aggregated Reservation

In 8.7, we illustrate how RT-RTI can create QoS communication among federates for each attribute/interaction. In some situations, an attribute/interaction may not need a private unicast channel or multicast group. For a more efficient use of network resources, RT-RTI should support an aggregate reservation that allows multiple attributes/interactions sharing the same unicast channel or multicast group with reserved network resources.

The *aggregatable* column of an attribute/parameter table in OMT is a flag for this purpose. For an attribute/interaction with "No" in this column, RT-RTI has to create a private channel for it while a "Yes" in this column, results in RT-RTI transferring data of the attribute/interaction through a shared channel. The resources reserved for the channel is based on the statistics of QoS-tuples of all attributes/interactions that share the channel.

8.7.3. Priority of Attribute/Interaction

The *priority* field in a QoS-tuple identifies the priority level of an attribute/interaction. This helps RT-RTI judge how to process an attribute/interaction.

RT-RTI could assign a dedicated thread to handle an attribute/interaction update task if the priority of the attribute/interaction is high enough. By assigning the thread with a

high OS priority, updating to the attribute/interaction will experience shorter processing latency in operating systems especially in RTOSs.

It is also possible that an *aggregate* channel is shared by multiple attributes/interactions. So the message from a sender thread may be saved in an outgoing queue after multiplexing and before delivering. The incoming messages from the aggregate channel may be saved in an incoming queue before being de-multiplexed to the threads that share the channel. The priority of the attribute/interaction could be used to identify the position of its messages in the queues. Messages from higher priority attributes/interactions are inserted anteriorly into the queues.

Part IV

Building Networks for RT-DIS

Chapter 9
Building QoS Networks – Trials

This chapter presents the results obtained by verifying the various QoS mechanisms available and the possibility of their integration, including RSVP-based IntServ, DiffServ, RSVP over DiffServ.

9.1. Building RSVP-enabled Networks

In this section, the processes of building various RSVP enabled networks are illustrated. The performance analysis is shown that RSVP-based mechanism can guarantee QoS.

9.1.1. Building End-to-End RSVP-enabled Unicast Network

9.1.1.1. Connecting the network

As shown in Figure 9.1, the three C3640 routers are used as Layer3 Backbone and the two C2620 routers are used as LAN gateways. The IOS version of C3640 routers is C3640-IS-M 12.2(T). The IOS version of C2600 routers is C2600-IS-M 12.2(T).

The routers are connected through WIC-1T serial WAN interface and V.35 DTE/DCE cables. The Hubs (CentreCom MR820TR IEEE 802.3 10Base-T Multi-port Hub/Repeater) connect to the routers through Ethernet ports.

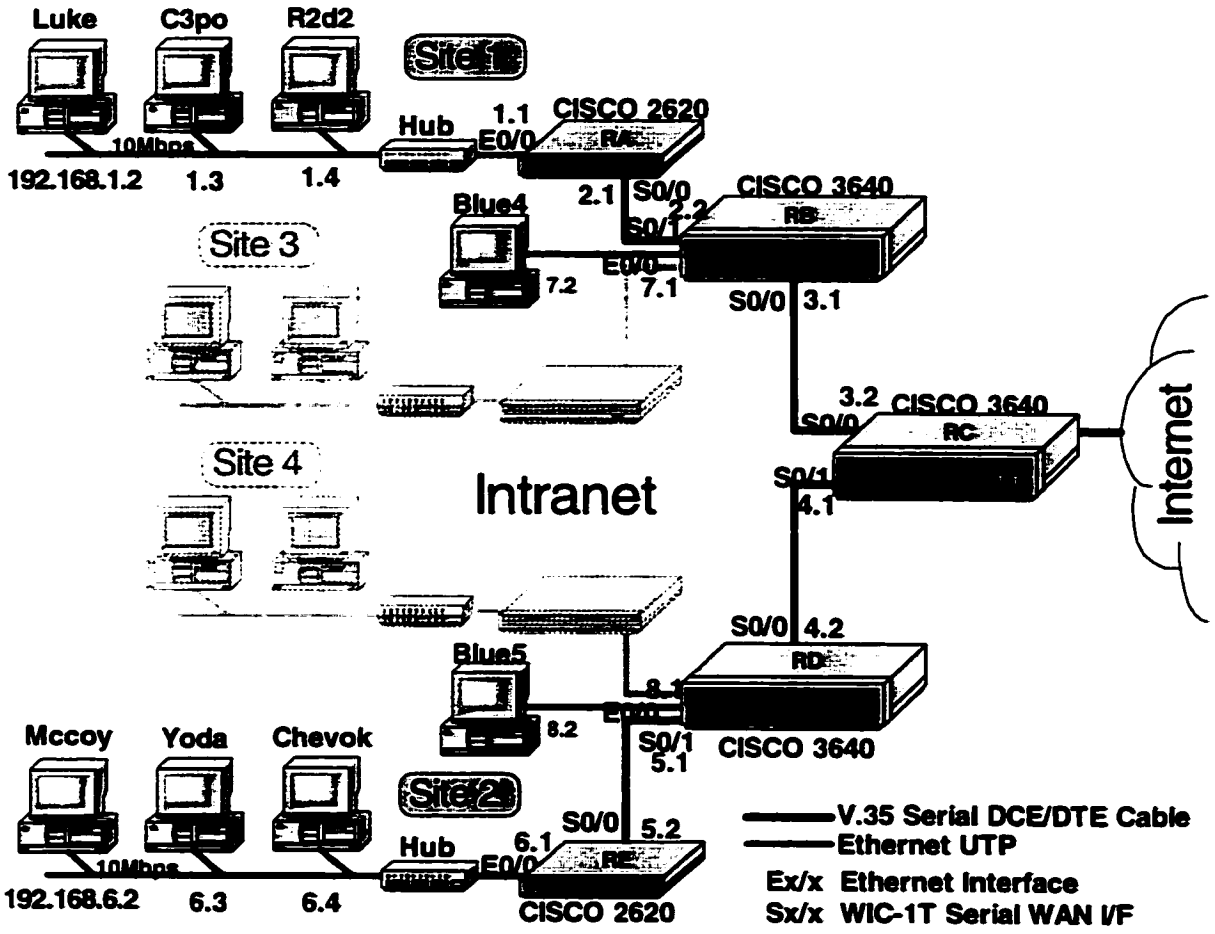


Figure 9.1 Topology of the Unicast

Luke, C3po, Mccoy, R2D2, Yoda, and Chevok are Pentium II 400 PCs with 128M memory and 3Com EtherLink 10/100 PCI NICs. Two local domains are built and named *Site1* and *Site2*. We connect the two domains through the network. Blue4 and Blue5 are Pentium IV 933M with 384M memory and Intel PRO/100VE connection NIC.

Domain *Site1* has Luke as domain controller and *Site2* has Mccoy as domain controller. Windows2000 Advanced Server is installed at the domain controllers and Windows2000 Professional at the other PCs.

9.1.1.2. Install Site1

Domain Controller

Windows 2000 Advanced Server was installed on the domain controller Luke. To support QoS, the Windows operating system need to install several un-default components. When the installation wizard prompts to choose the optional components, two additional components are chosen:

- *Network Monitor Tools* (in Management and Monitoring Tools).
- *QoS Admission Control Service* (in Networking Service).

When the installation wizard prompts to choose*network settings* between *typical* and *custom*, *custom* is chosen. Subsequently, the following operations are performed:

- Install Service *QoS Packets Scheduler*
- Install Protocol *Network Monitor Driver*
- TCP/IP properties: set IP address to *192.168.1.2*, set subnet mask to *255.255.255.0*, set default gateway to *192.168.1.1*

After the installation, the *Windows 2000 Configure Your Server* window showed up when a user first login to the system. Chose *"this is the only server in my network"*. Windows then installs Active Directory, DHCP and DNS into the domain controller. When the installation asked for domain names, *Site1.local* was entered as domain name.

Workstations

Windows 2000 Professional was installed at C3po.

The following components and services are installed or activated as a part of the network settings:

- Install Service *QoS Packets Scheduler*
- Install Protocol *Network Monitor Driver*
- TCP/IP properties. set IP address to *192.168.1.3*, set subnet mask to *255.255.255.0*, set default gateway to *192.168.1.1*
- Start RSVP service at: *Control Panel* | *Administrative Tools* | *Services* | *QoS RSVP*
- Set *Control Panel* | *Administrative Tools* | *Services* | *QoS RSVP* | *Startup Type* to *Automatic* (default is *Manual*)

C3po then joined the domain *Site1.local* through Control Panel | System | Network Identification. R2d2 was installed and configured with IP address 192.168.1.4, and joined the same domain.

Enable DSBM

As introduced in 6.3.7, the SBM scheme is necessary to support RSVP on an Ethernet. In a Windows 2000 network, the DSBM feature is integrated with the OS's QoS Admission Control service (ACS). The ACS is available in the Windows 2000 server only. It operates on hosts that provide network management functionality. ACS determines the subsets of network traffic that gain access to specific resources in the network based on 'policy' – a description of the quality and efficiency objectives to be met by the network.

- Set subnet

QoS Admission Control in Start | Programs | Administrative Tools is opened. In the console tree, *Subnetwork* Settings are activated and at the Action Menu, *Add subnetwork* is Chosen. (subnet 192.168.1.0/24).

- Configure DSBM host

On the Action Menu, the Properties of subnet *192.168.1.0/24* are set as follows: Traffic Tab, *Enable Admission Control Service on this subnet.* The limits (in Kbps) for Controlled Load and Guaranteed service can be specified individually or alternatively, the aggregated resource limit for all services to be assigned to this subnet can be specified. In our case, it is kept as the default value – unlimited.

In order to identify Luke as the DSBM of the subnet, on the *Action* Menu, followed by *Properties Server* configuration is altered by the *Add* command and Luke is added to the list of servers allowed to run QoS Admission Control (DSBM).

- Verify the DSBM configuration

A DSBM detection tool named *wdsbm* can be downloaded from Microsoft website [119] to verify the DSBM configuration. By implementing *wdsbm* at C3po command prompt, the following output can be seen if the DSBM has been set correctly:

```
C:>wdsbm

Host Name C3po
Interface 192.168.1.3
Found DSBM on interface 192.168.1.0/24

IP address of the DSBM:                0x206a8c0/192.168.1.2
DSBM election priority:                4
I_AM_DSBM refresh interval in secs:    5
```

DSBM dead interval in secs:	*15*
Max token bucket rate w/o resv:	*-1*
Max token bucket size w/o resv:	*-1*

- Setup policy

IETF SBM is a protocol entity only and intercepts RSVP messages for the purpose of participating in admission control but it does not make admission control decisions itself. ACS integrates a policy server that can decide to grant or reject a service request according to the policy setup by the network manager/administrator.

Due to ACS flexibility, the network manager can assign network bandwidth based on specific users. Policy configure point can be reached by following these steps:

- Open QoS Admission Control.
- In the console tree, click Enterprise Settings or subnet 192.168.1.0/24.
- On the Action Menu, click Add Policy.

Our objective is to verify the performance improvement of QoS LAN, so we setup the policy server with coarse policy – 4Mbps for both *flow limits* and *aggregate limits* for all users.

9.1.1.3. Install Site2

Following the same scenario as 0:

- Install Windows 2000 Advanced Server at Mccoy and set it as domain controller with IP address *192.168.6.2, 255.255.255.0, 192.168.6.1*, domain name *Site2.local.*
- Install Windows 2000 Professional at Yoda with IP address *192.168.6.3, 255.255.255.0, 192.168.6.1*

- Install Windows 2000 Professional at Chevok with IP address *192.168.6.4, 255.255.255.0, 192.168.6.1*
- Set subnet *192.168.6.0/24* and Mccoy as DSBM.
- Set same policy as domain *Site1.local.*

9.1.1.4. Install Background Traffic Generator

Install Blue4 and Blue5, but with different IP addresses:

- Install Windows 2000 Professional at Blue4 with IP address *192.168.7.2, 255.255.255.0, 192.168.7.1*
- Install Windows 2000 Professional at Blue5 with IP address *192.168.8.2, 255.255.255.0, 192.168.8.1*

9.1.1.5. Router configuration (Refer to Appendix A.1.)

Please refer to Appendix A.1 for the detail configuration scripts.

All interfaces are RSVP-enabled through "*ip rsvp bandwidth p1 p2*" command so they could handle RSVP messages and reserve bandwidth for QoS flows. Parameter *p1* represents the maximum amount of bandwidth that may be allocated by all RSVP flows while *p2* represents the maximum bandwidth that may be allocated to a single flow.

The maximum throughput of the serial interface among routers is 4Mbps identified by "*clockrate 4000000*" command (DCE needs to set *clockrate* while DTE does not). RSVP enabled flows can reserve part of the whole bandwidth (in our case, 1.158Mbps).

Cisco routers use WFQ as the workhorse for RSVP [76], so we enable WFQ for all the interfaces through "*fair-queue*" command. The parameter *p1* represents the number of

packets allowed in each queue; *p2* represents the number of dynamic queues used for best-effort traffic; *p3* represents the number of reserved queues used for QoS traffic.

In the verification, we use static IP routing to simplify the configuration.

9.1.1.6. Verifying RSVP connectivity

After checking the connectivity using commands *ping* and *tracert*, we use command *pathping* to check whether the RSVP feature has been enabled on all PCs and routers. Implement *pathping* at Yoda to check RSVP service between Yoda and C3po:

```
C:\pathping -R 192.168.1.3

Tracing route to C3PO [192.168.1.3] over a maximum of 30 hops:
    0  yoda.HLA.local [192.168.6.3]
    1  192.168.6.1
    2  192.168.5.1
    3  192.168.4.1
    4  192.168.3.1
    5  192.168.2.1
    6  C3PO [192.168.1.3]

    Checking for RSVP connectivity
    ------------------------------
    92.168.6.1  RSVP AWARE!
    92.168.5.1  RSVP AWARE!
    92.168.4.1  RSVP AWARE!
    92.168.3.1  RSVP AWARE!
    92.168.2.1  RSVP AWARE!
    92.168.1.3  RSVP AWARE!
```

The node with *RSVP AWARE* output associated with it indicates that RSVP service is enabled at that node. Otherwise, *Non RSVP Aware* output is associated with the node.

9.1.1.7. Background Traffic

We use *LanTraffic* to generate background UDP traffic with the following specifications between Blue4 and Blue5:

- Service type: Best Effort
- Random packet size: 1024 ~ 32768 bytes
- Random packet interval: 10 ~ 25 msec
- Mean throughput: 5.85Mbps

Recall (section 9.1.1.4) that the maximum bandwidth between two routers is 4Mbps and this traffic can congest the links between the site 1 and site 2.

9.1.1.8. Performance Measurement

We use *Chariot* to generate one measurement traffic flow between *endpoints* Yoda and C3po with 1Mbps rate and 65,200-byte packets. We assign the *best-effort guaranteed service* and *controlled load service* to the flow separately and measure the performance. Both *guaranteed service* and *controlled load service* are identified by the token-bucket model with a 128,000 bytes/sec *token-rate* and a 64,000-byte *token-bucket-size*.

Throughput Comparison

Figure 9.2 to Figure 9.4 illustrate the throughput of best-effort service, guaranteed service and controlled load service. It is obvious that the flows with guaranteed service and controlled load service can get enough and stable throughput in the congestion situation, while the flow with best-effort service cannot.

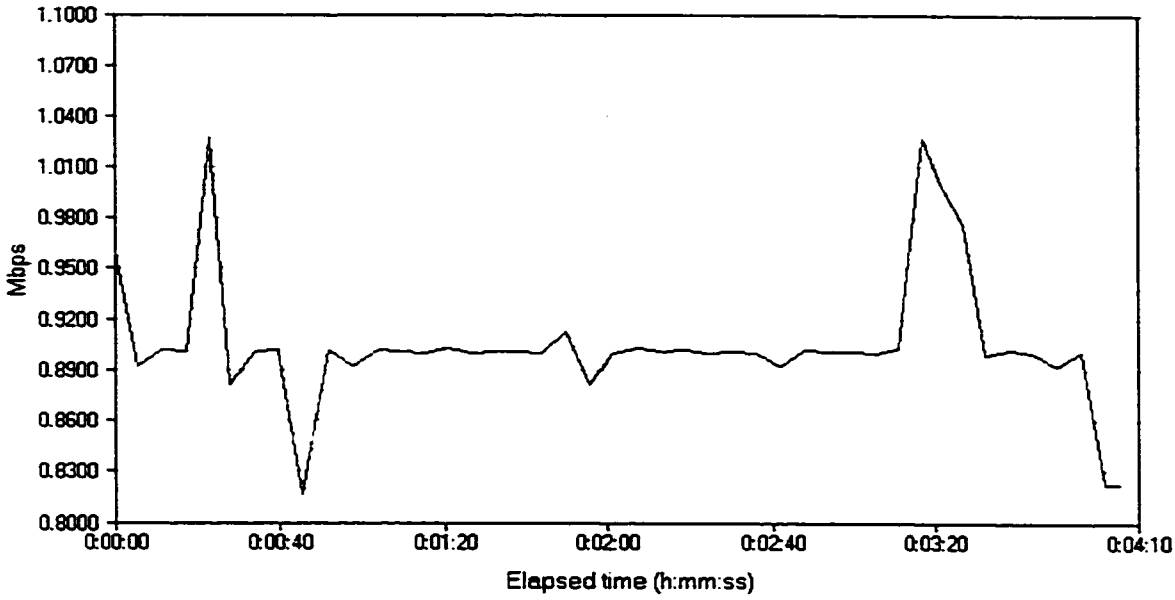


Figure 9.2 Throughput of Best-Effort

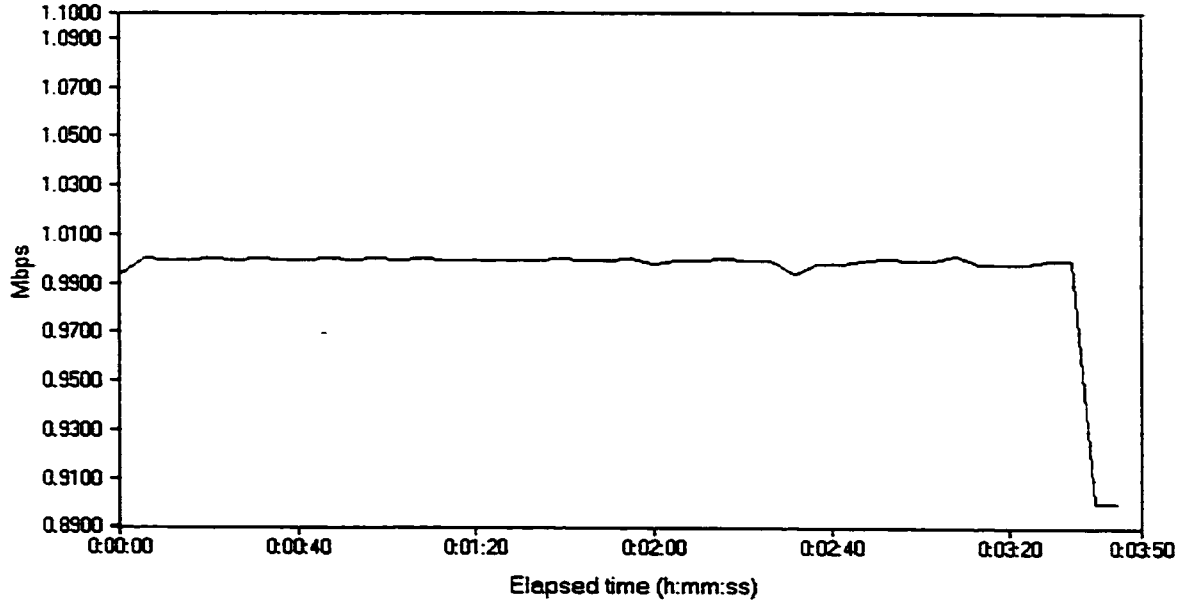


Figure 9.3 Throughput of Guaranteed Service

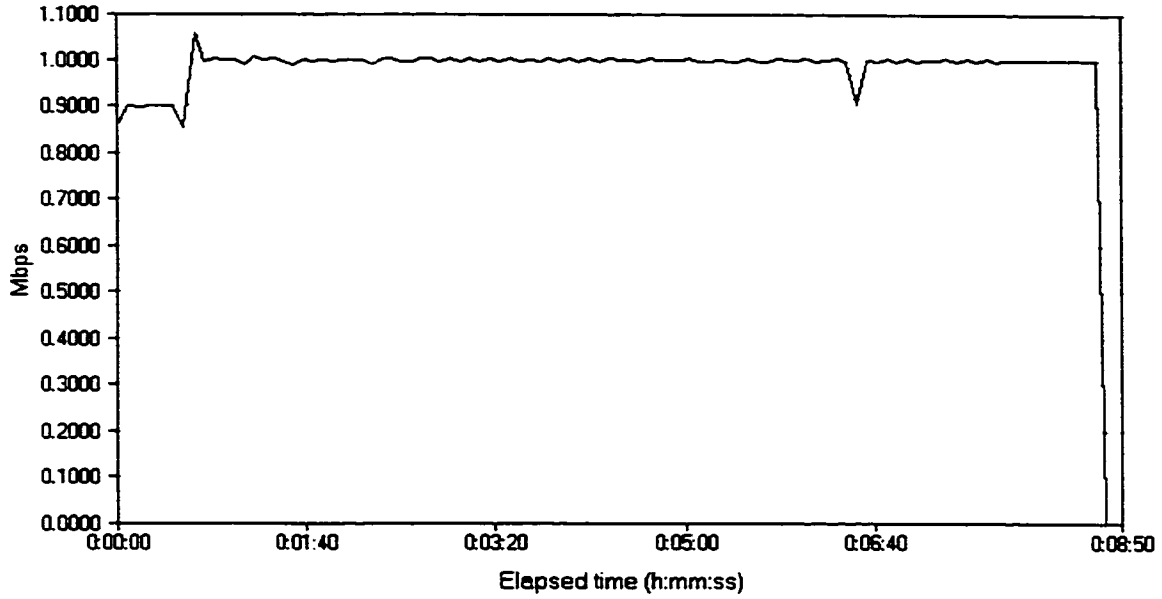


Figure 9.4 Throughput of Controlled Load Service

One-way Delay Comparison

Figure 9.5 to Figure 9.7 illustrate the one-way delay of best-effort service, guaranteed service and controlled load service. It can be observed that the flows with guaranteed service and controlled load service stable one-way delay even under congestion, while one-way delay of the flow with best-effort service varies considerably. It seems that guaranteed service and controlled load service cause longer delay to packets than best-effort service, we will discuss delay issues of RSVP in session 9.1.3.

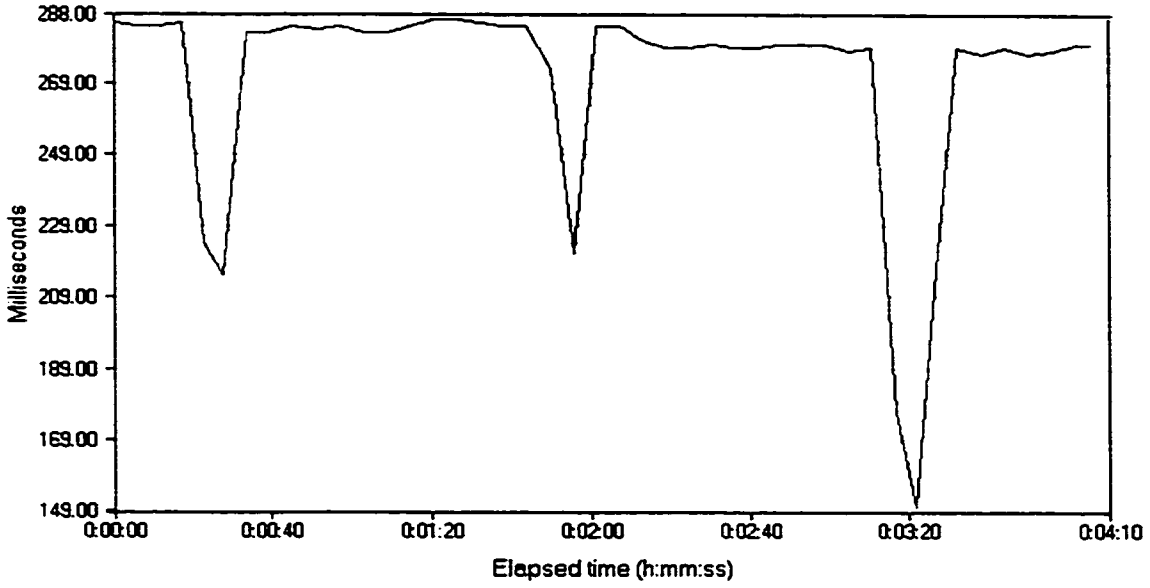


Figure 9.5 One-way Delay of Best-Effort Service

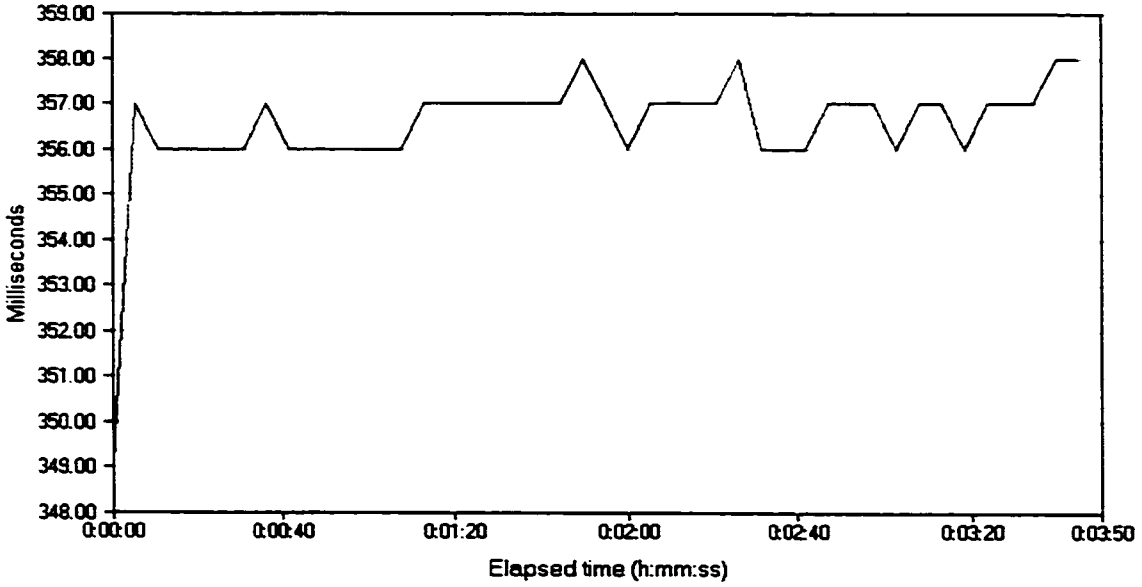


Figure 9.6 One-way Delay of Guaranteed Service

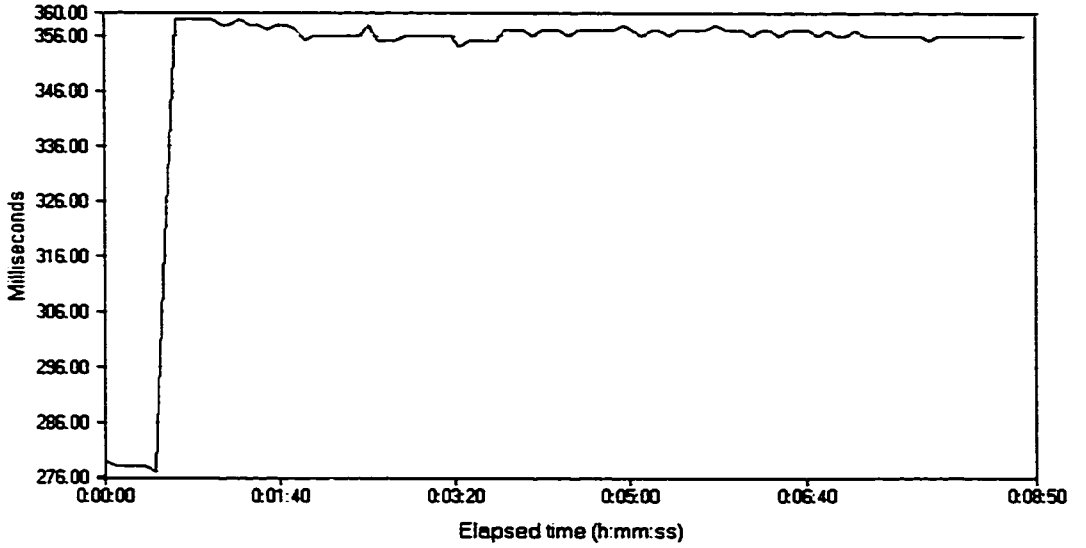


Figure 9.7 One-way Delay of Controlled Load Service

Loss Rate Comparison

Figure 9.8 to Figure 9.10 illustrate the loss rate of best-effort service, guaranteed service and controlled load service. It is obvious that the flow with guaranteed service gets near-zero loss rate and the flow with controlled load service gets very low loss rate in the congestion situation, while the flow with best-effort service gets high loss rate.

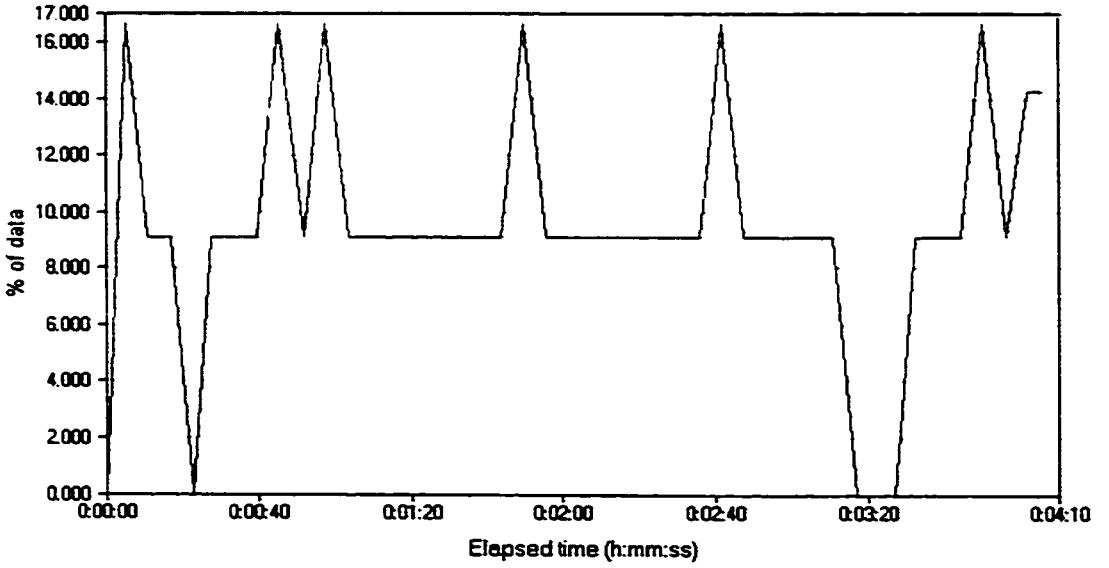


Figure 9.8 Loss Rate of Best-Effort Service

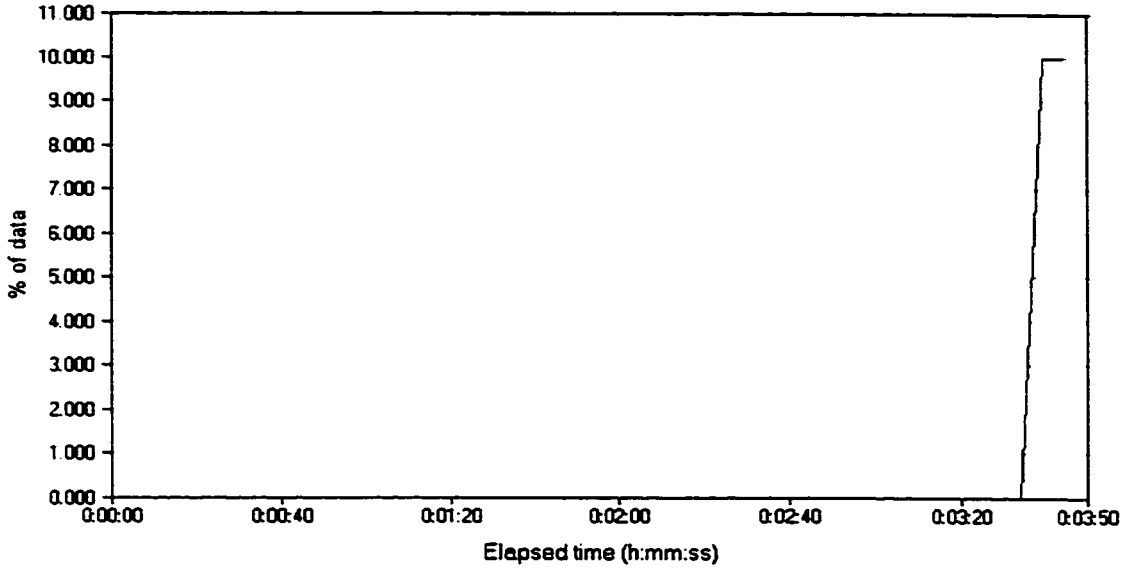


Figure 9.9 Loss Rate of Guaranteed Service

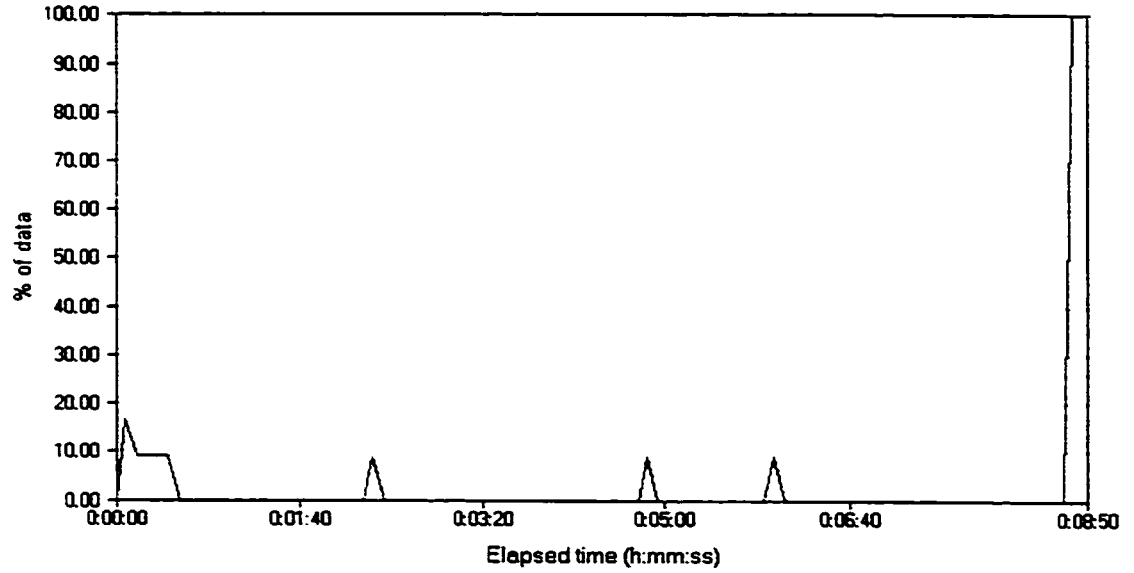


Figure 9.10 Loss Rate of Controlled Load Service

Jitter Comparison

Figure 9.11 to Figure 9.13 illustrate the jitter of best-effort service, guaranteed service and controlled load service. Obviously, the flows with guaranteed service and controlled

load service get jitter less than 1ms, while the flow with best-effort service gets jitter as high as 14ms.

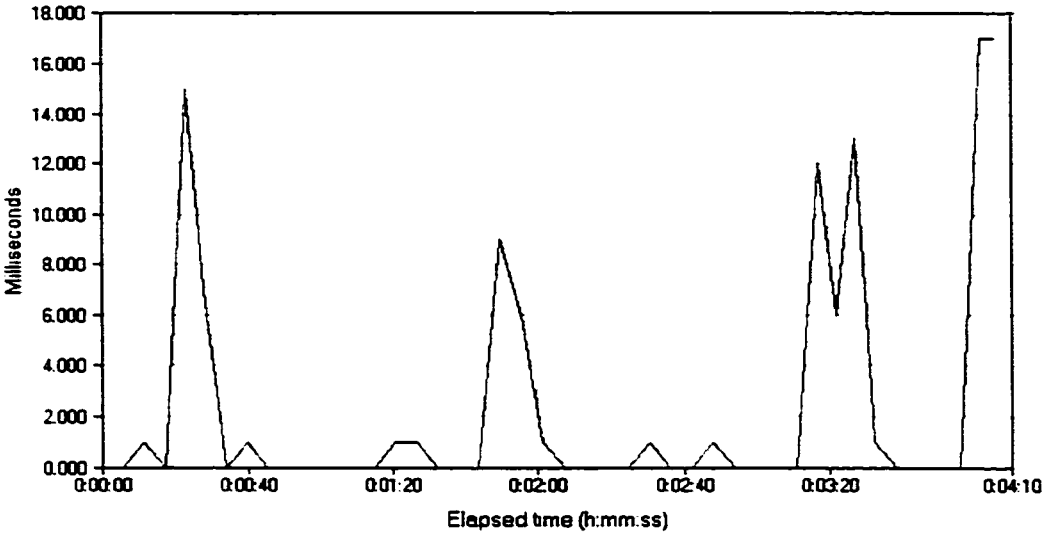


Figure 9.11 Jitter of Best-Effort Service

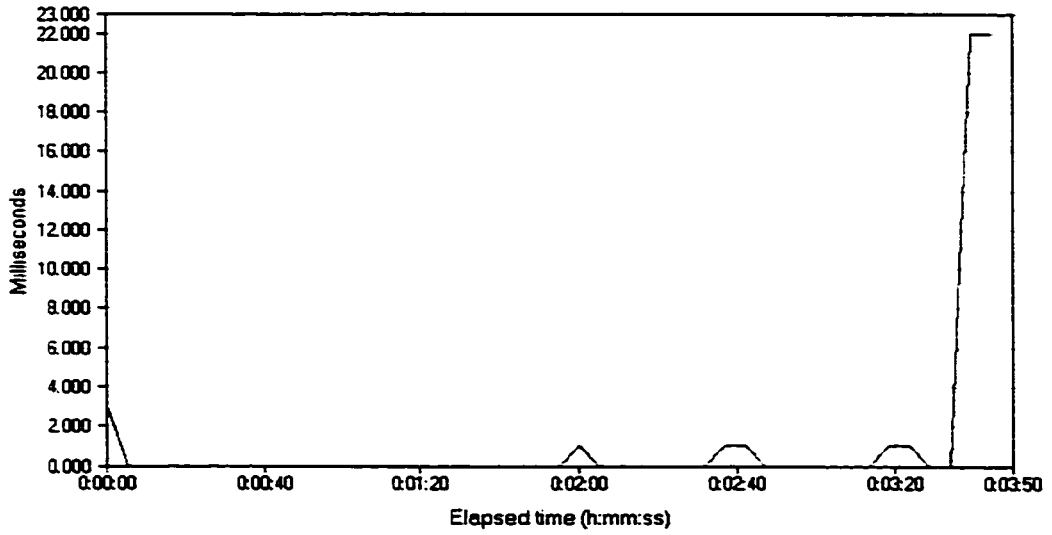


Figure 9.12 Jitter of Guaranteed Service

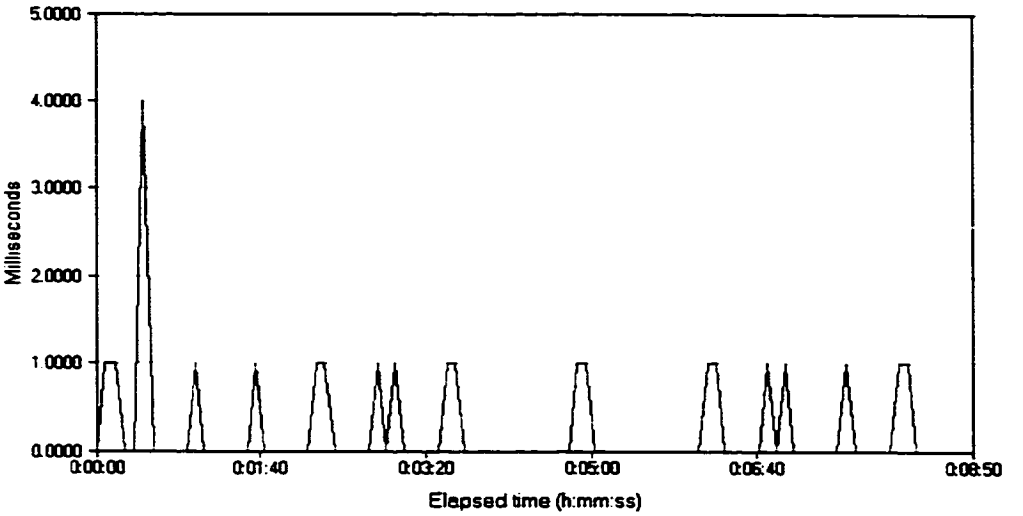


Figure 9.13 Jitter of Controlled Load Service

9.1.1.9. Statistics

Test	Test1	Test2	Test3
Service Quality	Best Effort	Controlled Load	Guaranteed
Token Rate (bytes/sec)	N/a	128000	128000
Token Bucket Size (bytes)	N/a	64000	64000
Offered Load (Mbps)	1.000	1.000	1.000
Packet Size (bytes)	65200	65200	65200
Average Throughput (Mbps	0.906	0.989	0.997
Average One-way Delay(ms	270.558	351.490	356.488
Lost Rate	9.362%	1.000%	0.233%
Jitter (ms)	2.163	0.090	0.698

Table 9.1 Network Performance Comparison under Different QoS

9.1.1.10. Summary Observations

- Even during the existence of congestion, flows with *guaranteed service* and *controlled*

load service could get the required bandwidth.

- Even in a congested situation, flows with *guaranteed service* and *controlled load service* could guarantee low loss rate.

9.1.2. Building End-to-End RSVP-enabled Multicast Network

Using the same routers and PCs as before but by changing the topology, an RSVP enabled multicast network is built.

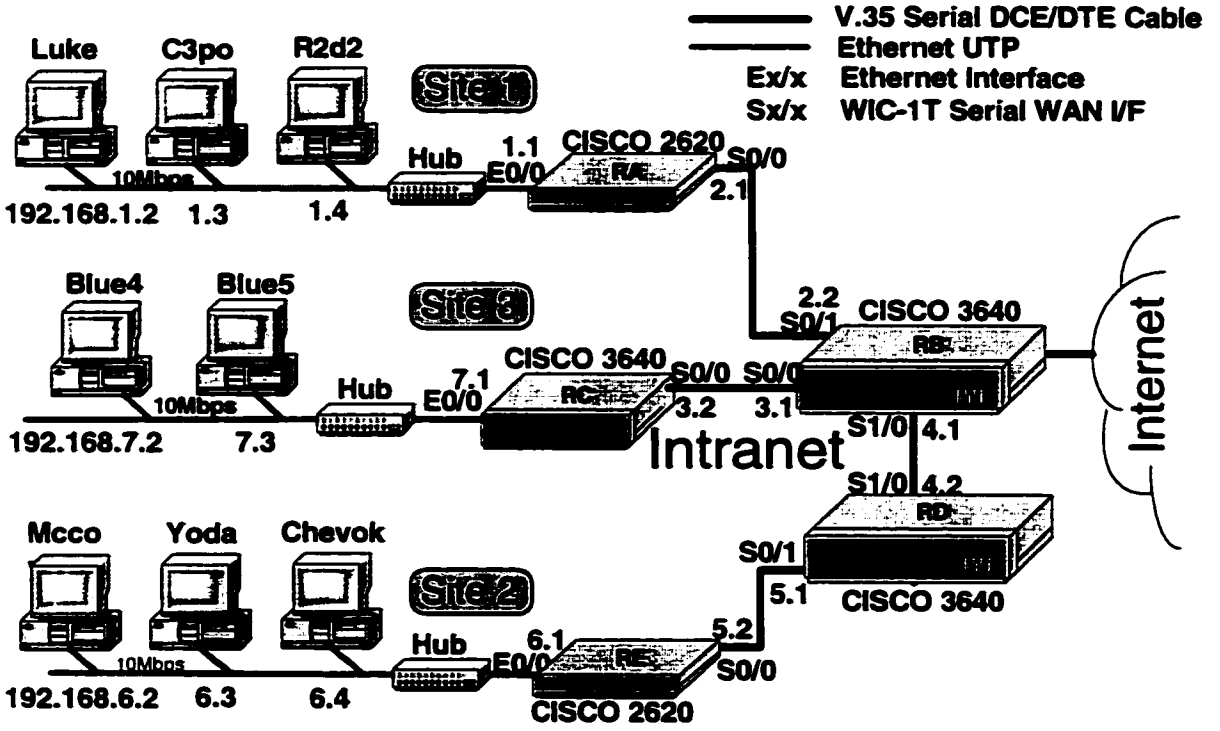


Figure 9.14 Topology of the Multicast

9.1.2.1. Connecting the network

As shown in Figure 9.14, two 3640 routers are used as Layer 3 Backbone, the last 3640 router and the two 2620 routers are used as LAN gateway for the multicasting configuration.

9.1.2.2. Install Site 3 Operating Systems

Following the same scenario as 0:

- Install Windows 2000 Advanced Server at Blue4 and set Blue4 as domain controller with IP address *192.168.7.2*, subnet mask *255.255.255.0*, default gateway *192.168.7.1*, domain name *Site3.local*.
- Change Blue5's IP address to *192.168.7.3*, *255.255.255.0*, *192.168.7.1*.
- Set subnet *192.168.7.0/24* and Blue4 as DSBM.
- Set same policy as domain *Site1.local*.

9.1.2.3. Router configuration

Please refer to Appendix A.2 for the detail configuration scripts.

The configuration is similar to 9.1.1.5, while in addition, all interfaces should add the command "*ip pim dense-mode*" to support multicasting.

9.1.2.4. Performance Measurement Result

We use two multicast groups for one sender (*192.168.1.3*) and two receivers (*192.168.6.3* and *192.168.7.3*) located in different domains.

Group one is used as background traffic to generate congestion with the following characteristics:

- Multicast group address: *224.8.8.8:8888*.
- 4Mbps UDP traffic with 32768 bytes packet size.
- Service quality is always identified as best effort.

Group two's traffic is used to measure the performance with different QoS. The result is summarized in Table 9.2.

Pair	Test1	Test2	Test3
Multicast Address	*224.6.6.6:6666*	*224.6.6.6:6666*	*224.6.6.6:6666*
Sender	192.168.1.3	192.168.1.3	192.168.1.3
Receiver 1	192.168.6.3	192.168.6.3	192.168.6.3
Receiver 2	192.168.7.3	192.168.7.3	192.168.7.3
Protocol	RTP	RTP	RTP
Service Quality	Best Effort	Controlled Load	Guaranteed
Token Rate (bytes/sec)	n/a	128000	128000
Token Bucket Size (bytes)	n/a	64000	64000
Offered Load (Mbps)	0.500	0.500	0.500
Packet Size (bytes)	Random packet size obeying normal distribution. With min as 64 bytes and max. as 1024 bytes.		
Average Throughput (Mbps)	0.444	0.501	0.501
Average One-way Delay (ms)	151.929	182.318	225.4
Loss Rate (%)	11.392	0	0
Average Jitter	2.358	0.546	5.55

Table 9.2 RSVP Multicast Performance

9.1.2.5. Summary Observations

- Even with congestion present, flows with *guaranteed service* and *controlled load service* could get the required bandwidth for multicast communication.

- Flows with *guaranteed service* and *controlled load service* could guarantee low loss rate for multicast communication even with congestion present.

9.1.3. RSVP Delay Analysis

From the last two tests, it seems that the *guaranteed service* and *controlled load service* experience longer delay. To make clear the relationship between the service quality and one-way delay, we compared one-way delay based on different QoS parameters (refer to Table 9.3).

Test	Service Quality	Token Rate	Token Bucket Size	Max. Delay (ms)	Max. Jitter (ms)	Packet Size (Bytes)	One-way Delay (ms)
1	Best Effort	n/a	n/a	—	—	4096	21.35
2	Controlled Load	64000	32000	not set	not set	4096	31.97
3	Guaranteed	64000	32000	not set	not set	4096	30.14
4	Guaranteed	128000	64000	20	20	4096	31.07
5	Guaranteed	64000	32000	20	20	4096	32.02
6	Guaranteed	64000	**1024**	20	20	4096	**75.25**
7	Guaranteed	64000	32000	20	20	2048	17.64
8	Guaranteed	64000	32000	20	20	1024	11.82
9	Guaranteed	64000	32000	20	20	512	6.93

Table 9.3 Delay and Token Bucket Model

Observations

- Test 1 to 3 show that guaranteed service and controlled load service cause longer delay than best-effort service.

- Test 3 to 5 show delay is not sensitive to token-bucket parameters.
- Test 6 shows the delay increases dramatically when the token-bucket size is too small.
- For guaranteed service, the maximum delay and maximum jitter parameters seem to be ignored by the routers.
- Test 5, 7, 8, 9 show that delay is more sensitive to packet size, refer to Figure 9.15.

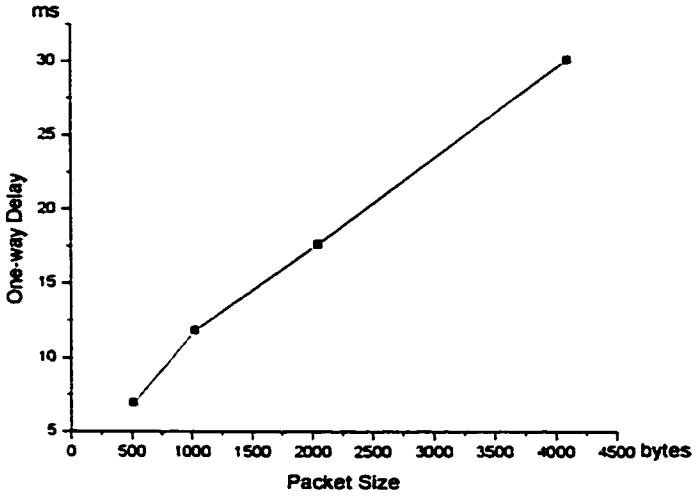


Figure 9.15 Relationship of One-way Delay and Packet Size for RSVP flows

9.2. Building a Differentiated Service Networks

9.2.1. Building a DiffServ Network

9.2.1.1. PHB and DSCP

A DiffServ network implements QoS through Per-Hop Behaviors (PHB) according to a marked DSCP value. IETF has identified four standard PHBs: Default PHB (RFC2474), Class-Selector PHB (RFC2474), Assured Forwarding (RFC2597) and Expedited Forwarding (RFC2598) [70]. The relationship between PHBs and DSCPs is shown in Table 9.4:

PHBs	DSCPs	Comments
Default	000000	Best Effort Service
Class-Selector	xxx-000 3 bits xxx for 8 level IP precedence	To preserve backward-compatibility with IP precedence scheme
Assured Forwarding	xxx-yy-0 3 bits xxx for 4 priority classes (001,010,011,100); 2 bits yy for 3 drop classes (01,10,11)	Provide a service similar to the Controlled Load service in IntServ model
Expedited Forwarding	101110	Provide a service similar to the Guaranteed service in IntServ model

Table 9.4 DSCPs and PHBs

9.2.1.2. Connecting the network

Figure 9.16 illustrates three 3640 routers used to simulate the backbone ISP and two 2620 routers used to simulate the campus network gateway. This overall topology is used for testing a DiffServ network.

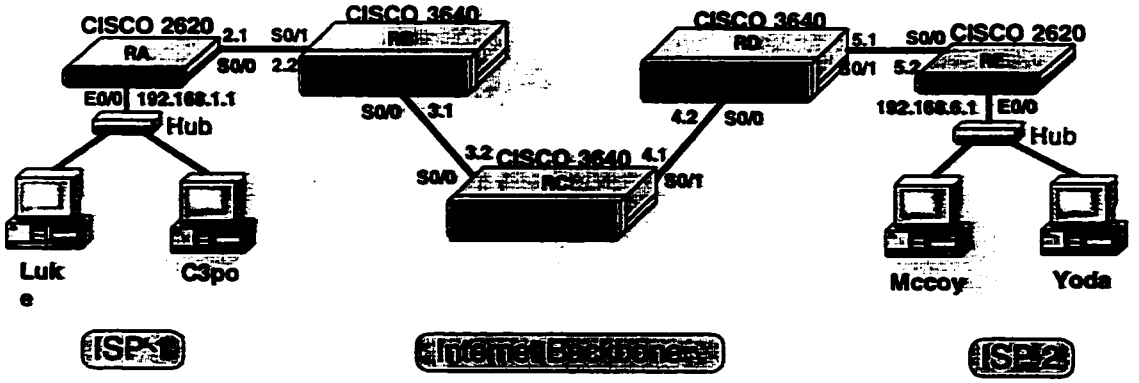


Figure 9.16 Implementing End-to-End DiffServ

9.2.1.3. Enabling DSCP marking at end systems

Windows 2000 allows applications to mark TOS and DSCP fields through Winsock2 APIs. However, a modification to the Registry is required: adding the following DWORD value at the endpoints:

> *HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\DisableUserTOSSetting = 0*

9.2.1.4. Configuring Routers

Please refer to Appendix A.3 for the detail configuration scripts.

The command "*class-map*" defines several service classes: *platinum*, *gold*, *silver*, *bronze*, *iron* according to different DSCP codes in "*match ip dscp*" command.

The command "*policy-map*" reserves different resources for different service classes:

- *priority 500* : provides priority queuing with 1Mbps reserved bandwidth for *platinum* service
- *bandwidth remaining percent x* : reserves x% remaining bandwidth for service
- *random-detect dscp-based*: enables WRED (see section 6.3.3) to use the DSCP value when it calculates the drop probability for a packet.

The command "*service-policy*" enables a policy at an interface.

9.2.1.5. Performance Measurement

Six RTP flows are sent from 192.168.6.3 to 192.168.1.3 across the DiffServ network with DSCP fields of 46, 10, 18, 26, 34 and 0 respectively (Table 9.5). The flow with 0 DSCP always has 4Mbps send rate to congest the network, while other flows use different flow rates from 0.2Mbps to 2.0Mbps (All flows use 1024 bytes packet size).

Pairs	Endpoint 1	Endpoint 2	Protocol	DSCP (Bin)	DSCP (Dec)	Service Category
1	192.168.6.3	192.168.1.3	RTP	101110	46	Expedited Forwarding
2	192.168.6.3	192.168.1.3	RTP	001010	10	Assured Forwarding AF11
3	192.168.6.3	192.168.1.3	RTP	010010	18	Assured Forwarding AF21
4	192.168.6.3	192.168.1.3	RTP	011010	26	Assured Forwarding AF31
5	192.168.6.3	192.168.1.3	RTP	100010	34	Assured Forwarding AF41
6	192.168.6.3	192.168.1.3	RTP	000000	0	Best Effort

Table 9.5 DiffServ Performance Measurement Flows

Throughput

Flows		Offered Load Per Flow (Mbps)				
No.	DSCP	0.200	0.500	1.000	1.500	2.000
1	46	0.191	0.429	0.486	0.488	0.559

2	10	0.191	0.427	0.917	1.360	1.369
3	18	0.191	0.428	0.919	1.006	1.025
4	26	0.192	0.427	0.894	0.598	0.609
5	34	0.192	0.428	0.619	0.399	0.407
6*	0	2.375	1.703	0.003	0.008	0.002

Note: Flow 6 send rate is always 4Mbps

Table 9.6 DiffServ Performance Measurement – Throughput

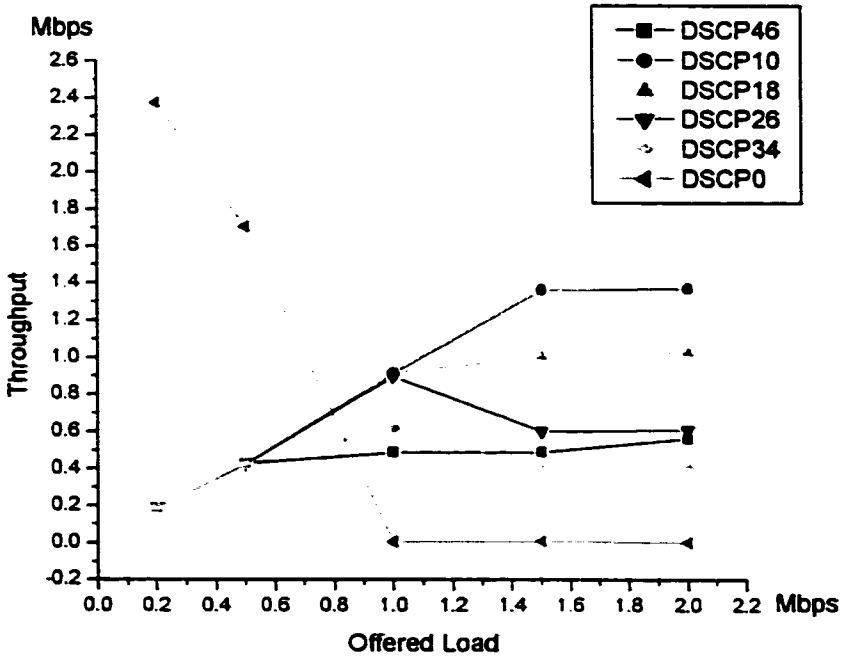


Figure 9.17 DiffServ – Throughput

Each service category has its own reserved bandwidth and the precedence to use its reserved bandwidth. If the category does not use up all its bandwidth, the remaining bandwidth can be shared by other categories. In the situation of congestion, it is guaranteed that all service categories get their reserved bandwidth.

One-way Delay

Flows		Offered Load Per Flow (Mbps)				
No.	DSCP	0.200	0.500	1.000	1.500	2.000
1	46	16.378	17.650	20.410	19.436	19.159
2	10	16.600	18.385	22.402	21.356	21.939
3	18	17.767	19.539	26.549	371.909	340.984
4	26	16.156	20.313	228.968	660.955	565.031
5	34	18.667	22.311	554.367	862.806	833.218
6*	0	22.433	314.054	16,018.000	22,366.333	10,421.000

Note: Flow 6's send rate is always 4Mbps

Table 9.7 DiffServ Performance Measurement – One-way Delay

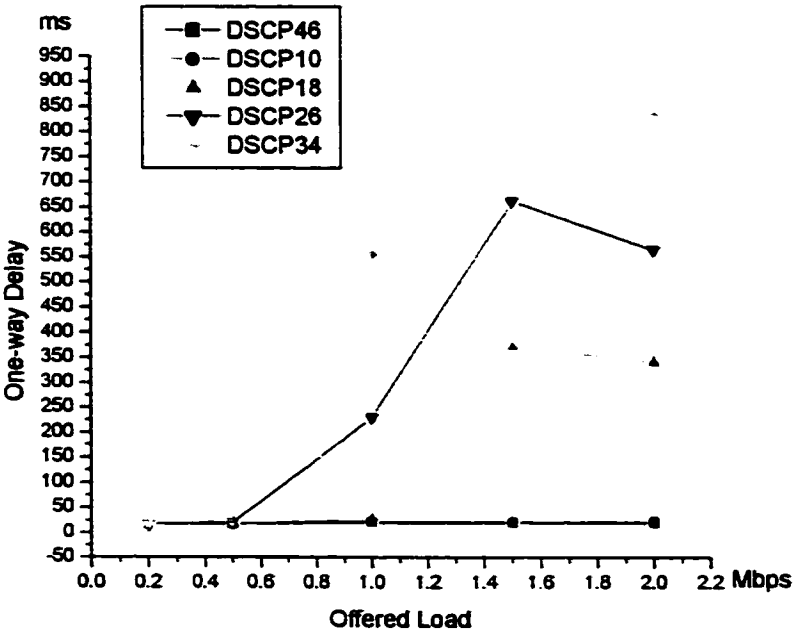


Figure 9.18 DiffServ – One-way Delay

Loss Rate

Flows		Offered Load Per Flow (Mbps)				
No.	DSCP	0.200	0.500	1.000	1.500	2.000
1	46	0.000	0.000	46.875	63.575	63.236
2	10	0.000	0.000	0.125	0.000	0.000
3	18	0.000	0.000	0.100	24.080	22.701
4	26	0.000	0.000	2.437	54.327	53.429
5	34	0.000	0.000	31.017	68.967	68.386
6*	0	0.552	40.425	100	100	100

Note: Flow 6's send rate is always 4Mbps

Table 9.8 DiffServ Performance Measurement – Loss Rate (%)

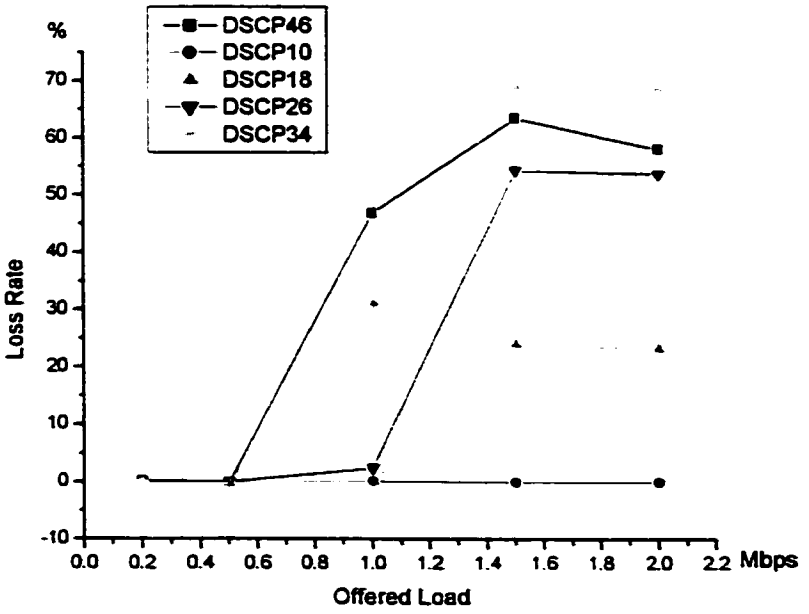


Figure 9.19 DiffServ – Loss Rate

One-way delay can be controlled through resource reservation. The more bandwidth is reserved for a category of traffic, the less delay the traffic is going to experience. If the traffic greatly exceeds the reserved bandwidth for that category, the delay experienced by the traffic increases dramatically.

If the traffic volume does not exceed its reserved bandwidth, the loss rate of the traffic will be very low. If the traffic exceeds its reserved bandwidth, the exceeding part is dropped, causing high loss rate.

Jitter

Flows		Offered Load Per Flow (Mbps)				
No.	DSCP	0.200	0.500	1.000	1.500	2.000
1	46	0.022	0.000	0.006	0.000	0.000
2	10	0.111	0.006	0.044	0.003	0.002
3	18	0.178	0.050	0.202	0.118	0.135
4	26	0.000	0.173	0.539	0.628	0.392
5	34	0.556	0.311	0.911	0.666	0.558
6*	0	1.433	1.504	503.000	154.333	752.000

Note: Flow 6's send rate is always 4Mbps

Table 9.9 DiffServ Performance Measurement – Jitter (ms)

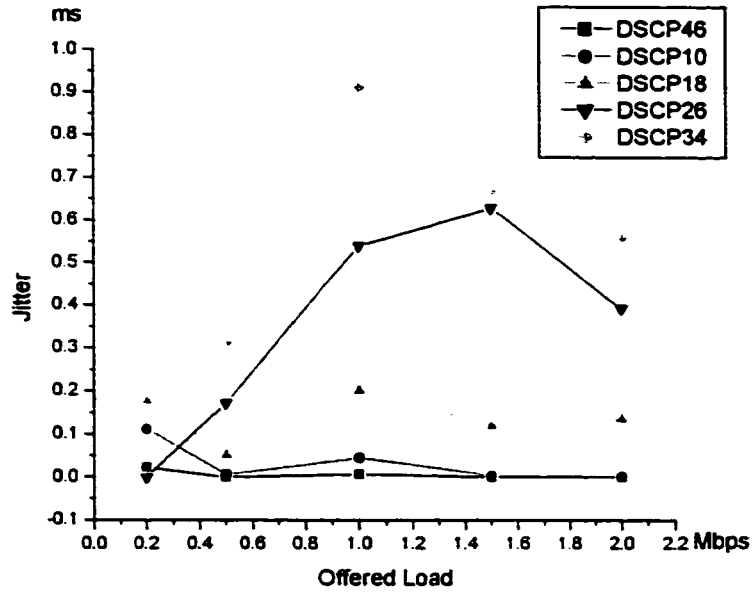


Figure 9.20 DiffServ – Jitter

The traffic category with more bandwidth reserved for it will experience shorter jitter.

9.2.2. Differentiated Service through Access List

In the previous section, packets were classified according to the DSCP field of the IP header. Alternatively, packet classification can be done through "Access-list", which allows classification of packets based on different fields (such as port number, IP address, MAC address, etc). This section presents the verification results of classifying packets using "Access-list" and providing QoS through DiffServ as was done in the previous section.

9.2.2.1. Connect the network

The network architecture is as same as 9.2.1.2.

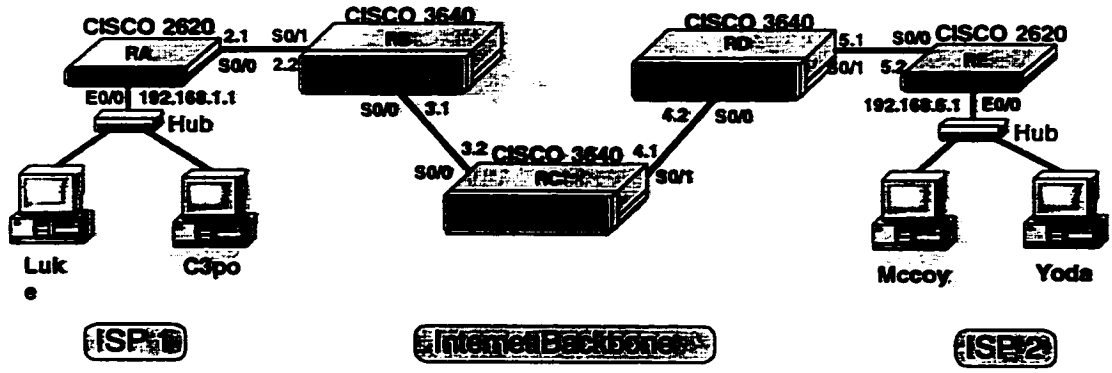


Figure 9.21 Differentiated Service Network through Access List

9.2.2.2. Router Configurations

Please refer to Appendix A.4 for the detail configuration scripts.

According to this configuration, all UDP flows are divided into several service categories according to their port number. For example, all UDP flows with port number between 60000~65535 will get *platinum* service.

9.2.2.3. Performance Measurement

Six RTP flows are sent from 192.168.6.3 to 192.168.1.3 across the DiffServ network with DSCP fields of 46, 10, 18, 26, 34 and 0 respectively. The flow with 0 DSCP always has 4Mbps send rate to congest the network, while other flows use different flow rates from 0.2Mbps to 2.0Mbps (All flows are 1024 bytes at packet size).

Pairs	Endpoint 1	Endpoint 2	Protocol	Port No.	Access Group
1	192.168.6.3	192.168.1.3	RTP	6000X	101
2	192.168.6.3	192.168.1.3	RTP	5000X	102
3	192.168.6.3	192.168.1.3	RTP	4000X	103
4	192.168.6.3	192.168.1.3	RTP	3000X	104
5	192.168.6.3	192.168.1.3	RTP	2000X	105
6	192.168.6.3	192.168.1.3	RTP	1000X	Other

Table 9.10 Access List – Flows

Throughput

Flows		Offered Load Per Flow (Mbps)				
No.	Group No	0.200	0.500	1.000	1.500	2.000
1	101	0.191	0.430	0.486	0.487	0.488
2	102	0.192	0.429	0.915	1.360	1.363
3	103	0.191	0.428	0.918	1.005	1.007
4	104	0.191	0.431	0.900	0.597	0.602
5	105	0.191	0.432	0.622	0.399	0.400
6*	Other	2.376	1.689	0.014	0.002	0.002

Note: Flow 6's send rate is always 4Mbps

Table 9.11 Access List - Throughput

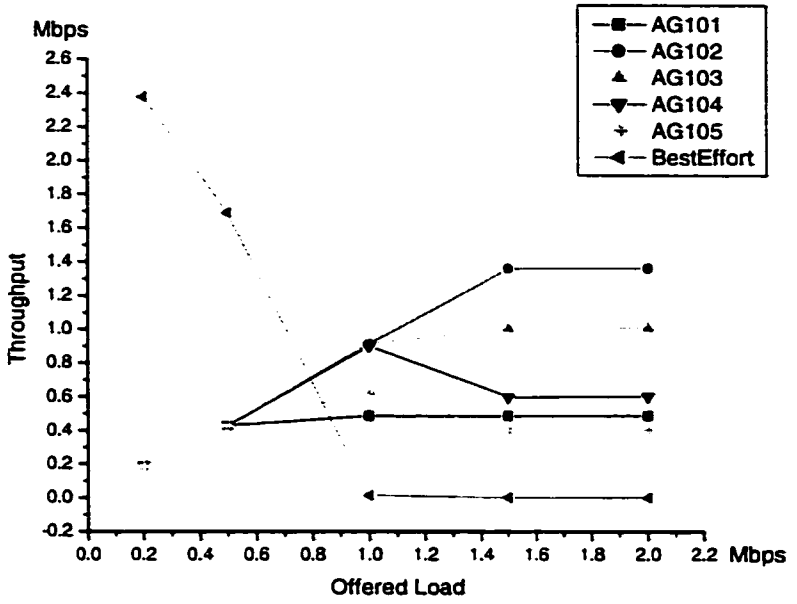


Figure 9.22 Access-List – Throughput

One-way Delay

Flows		Offered Load Per Flow (Mbps)				
No.	Group No	0.200	0.500	1.000	1.500	2.000
1	101	16.811	16.411	18.013	18.400	20.325
2	102	17.222	17.633	20.177	20.626	22.135
3	103	17.667	18.306	24.344	523.862	522.435
4	104	15.967	19.928	306.397	874.267	871.094
5	105	18.789	19.973	827.144	1,285.662	1,286.682
6*	Other	21.256	409.117	8,079.200	11,028.000	11,013.000

Note: Flow 6's send rate is always 4Mbps

Table 9.12 Access List - One-way Delay

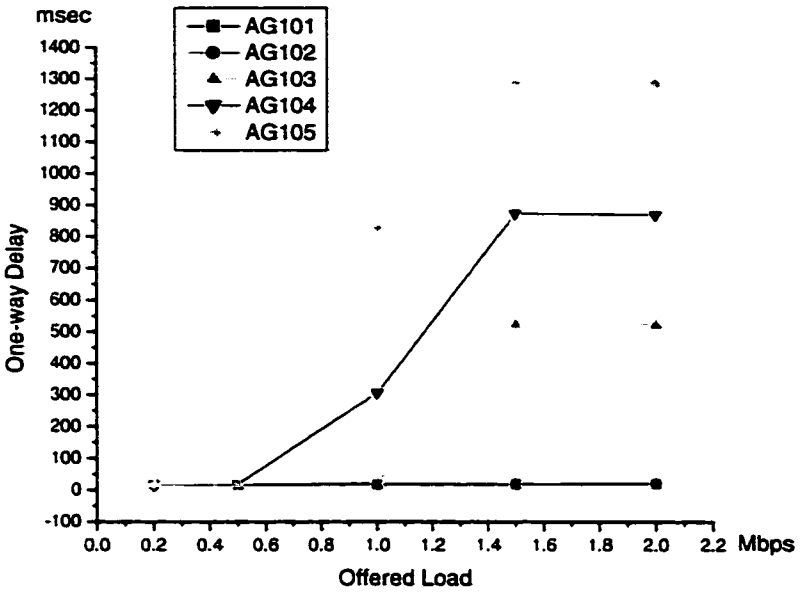


Figure 9.23 Access-List – One-way Delay

Loss Rate

Flows		Offered Load Per Flow (Mbps)				
No.	Group No	0.200	0.500	1.000	1.500	2.000
1	101	0.000	0.000	46.202	63.600	63.600
2	102	0.000	0.000	0.050	0.025	0.000
3	103	0.000	0.000	0.000	23.644	23.522
4	104	0.277	0.000	1.458	53.790	53.676
5	105	0.000	0.000	30.185	68.385	68.368
6*	Other	0.277	47.200	100	100	100

Note: Flow 6's send rate is always 4Mbps

Table 9.13 Access List – Loss Rate

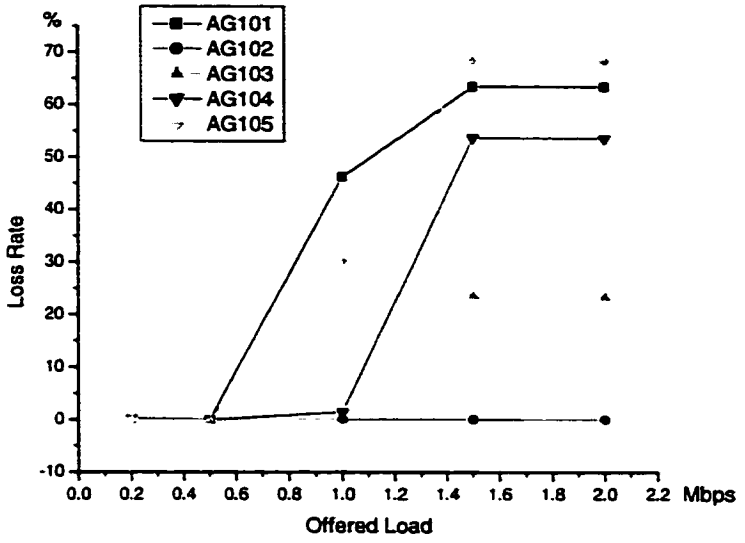


Figure 9.24 Access-List – Loss Rate

Jitter

Flows		Offered Load Per Flow (Mbps)				
No.	Group No.	0.200	0.500	1.000	1.500	2.000
1	101	0.489	0.000	0.002	0.003	0.000
2	102	0.100	0.028	0.051	0.003	0.000
3	103	0.633	0.056	0.194	0.105	0.083
4	104	0.000	0.282	0.410	0.252	0.229
5	105	0.589	0.302	0.732	0.348	0.301
6*	Other	0.500	0.800	272.800	757.000	757.000

Note: Flow 6's send rate is always 4Mbps

Table 9.14 Access List – Jitter

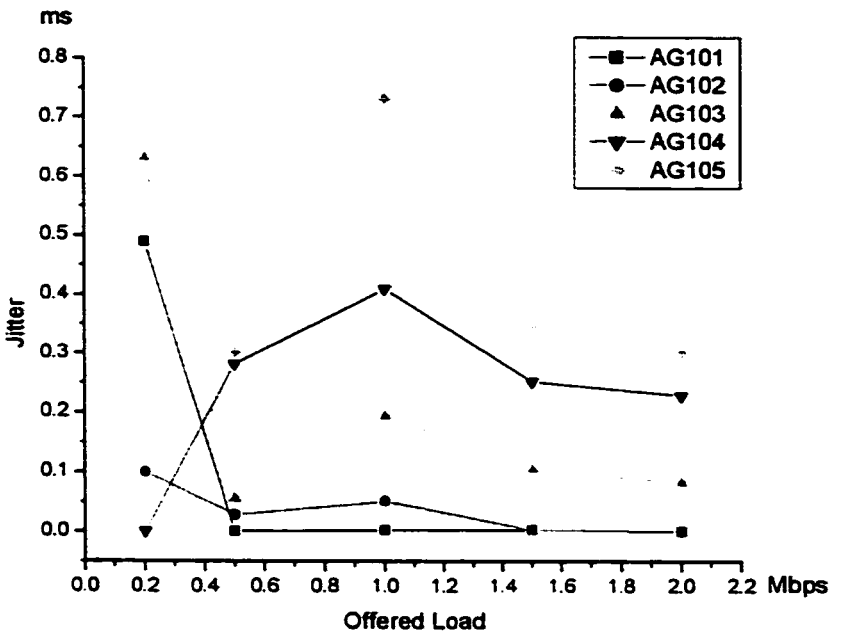


Figure 9.25 Access-List – Jitter

It is easy to see the results are very similar to the result of DiffServ network. It can be observed that whatever the classification method, DSCP value or port number, the same prioritization and queuing system is used and hence similar results are obtained.

9.3. RSVP over DiffServ Network

9.3.1. RSVP Scalability Enhancement

As introduced in 6.3.8, a trend becoming clear is that of the Internet is going to enable DiffServ or MPLS at the backbone and enable IntServ and RSVP at user network. This architecture has exceeded the theory and standard phase. Some network device vendors such as Cisco have begun to implement it, although the implementation is not standard compliant so far.

This Chapter illustrates how to implement Cisco RSVP Scalability Enhancement [120] to build a scalable fine-grained QoS network.

9.3.2. Connect the network

As shown in Figure 9.26 the topology of the network is similar to 9.2.1.2. The major difference is that the middle domain adopts Access List based Differentiated Services and the two side domains adopt RSVP flow based services. At the boundary of the domains, RSVP scalability enhancement feature is enabled to aggregate RSVP flows.

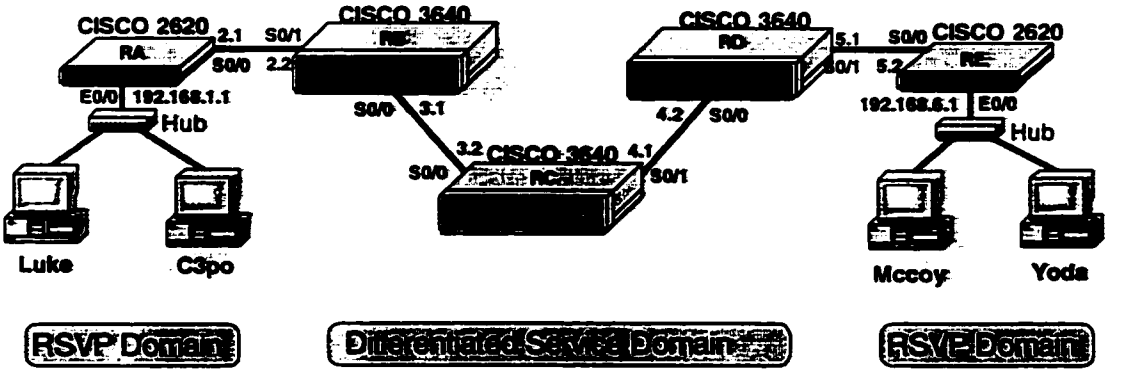


Figure 9.26 RSVP Scalability Enhancement

It is more meaningful if we enable native DiffServ in the middle domain but our routers could not support this feature well. So we use access list to classify the traffic instead of DSCP.

9.3.3. Configuring Routers

Please refer to Appendix A.5 for the detail configuration scripts.

The RA and RE are running classic RSVP, which means that RSVP is keeping a state per flow and also classifying, marking and scheduling packets on a per flow basis.

The RB and RD are running classic RSVP on the interfaces S0/1 connected to the RA and RE and running RSVP for admission control only on the interfaces S0/0 connected to core router RC by using two commands: "*ip rsvp data-packet classification none*" and "*ip rsvp resource-provider none*".

The core routers in the middle domain are not running RSVP, but are forwarding the RSVP messages to the next hop.

9.3.4. Performance Test

Pair	Send Rate Kbps	Port No	Access Group	Service Quality	Average Throughput (Mbps)	Average One-way Delay (ms)	Loss Rate	Jitter Average (ms)
1	400	6000X	101	Guaranteed*	0.406	13.589	0.000	0.018
2	400	5000X	102	Controlled Load*	0.407	14.805	0.000	0.027
3	400	4000X	103	Controlled Load*	0.406	16.004	0.000	0.118
4	400	1000X	-	Best Effort	0.371	207.879	8.519	0.988
5	400	1000X	-	Best Effort	0.371	209.490	8.464	1.022
6	400	1000X	-	Best Effort	0.371	206.192	8.477	0.989
7	400	1000X	-	Best Effort	0.371	208.574	8.536	1.001
8	1000	1000X	-	Best Effort	0.373	225.630	61.362	1.476
9	1000	1000X	-	Best Effort	0.371	231.301	61.503	1.558
10	1000	1000X	-	Best Effort	0.373	228.686	61.425	1.495

Note: token-bucket rate as 62.5Kbytes/sec and token-bucket size as 2Kbytes

Table 9.15 RSVP Scalability Enhancement

Obviously, RSVP scalability enhancement can guarantee bandwidth, delay, loss rate, jitter etc QoS characteristics as same as end-to-end RSVP scheme.

9.4. Ethernet Performance Measurement

For cost reasons, plain Ethernets without any QoS mechanisms are still widely deployed. In this section, we attempt to investigate the performance of Ethernet with regards to real-time traffic. The goal is to show the performance in the absence of QoS in order to further illustrate the importance of having QoS on Ethernets.

9.4.1. Hub based Ethernet

9.4.1.1.Connecting the Ethernet

Figure 9.27 shows eight PCs connected through a *NetGear* DS108 8 ports 10/100Mbps Hub to build an Ethernet LAN. All PCs install Windows 2000 Professional as their operating system. There is no domain controller in this Ethernet and all PCs are connected as a workgroup.

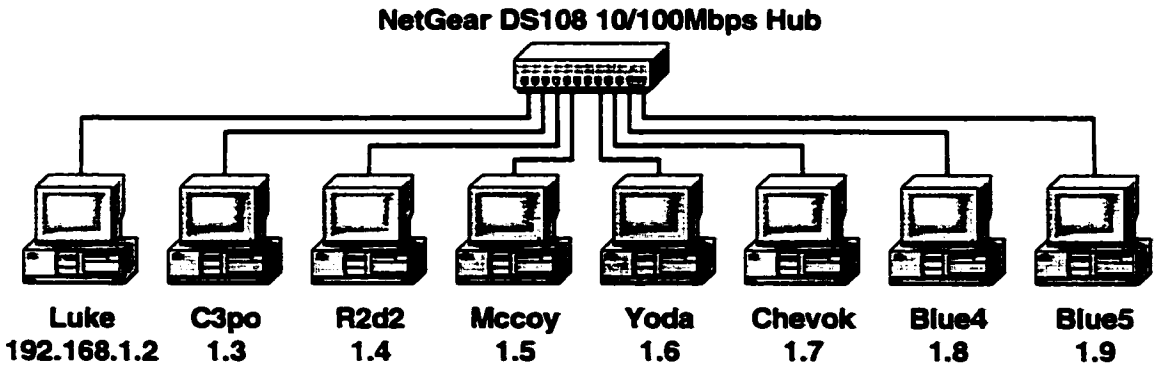


Figure 9.27 LAN Connections

9.4.1.2.Performance related parameters

A 100Mbps RTP flow is sent from 192.168.1.2(Luke) to 192.168.1.3(C3po) with different UDP packet sizes ranging from 250 bytes to 64000 bytes.

Packet Size (bytes)	Throughput (Mbps)	One-way Delay (ms)	Loss Rate (%)
250	13.424	0.027	0.000
500	27.692	0.017	0.000
1000	49.404	0.052	0.004
2000	43.447	0.001	0.000
4000	58.782	0.780	0.000
8000	67.633	0.000	0.000
16000	72.802	1.002	0.000
32000	76.122	2.965	0.000
64000	78.883	4.998	0.000

Table 9.16 Packet Size and Performance

When there is no congestion, a bigger packet size causes higher throughput and longer delay. With a maximum UDP packet size of 64Kbytes, the throughput can reach 78.833Mbps.

If there is only one flow through the Ethernet, there is no loss rate for that flow since there is no collision.

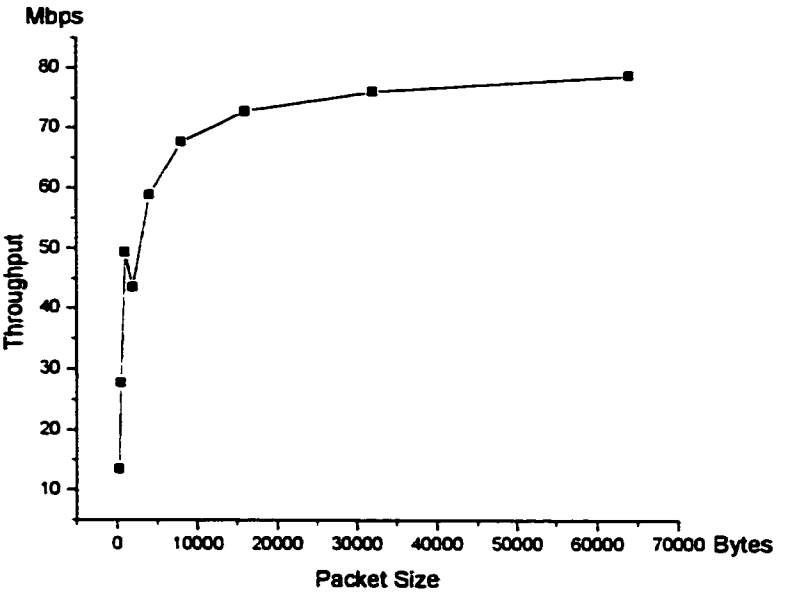


Figure 9.28 Hub: Throughput vs. UDP Packet Size

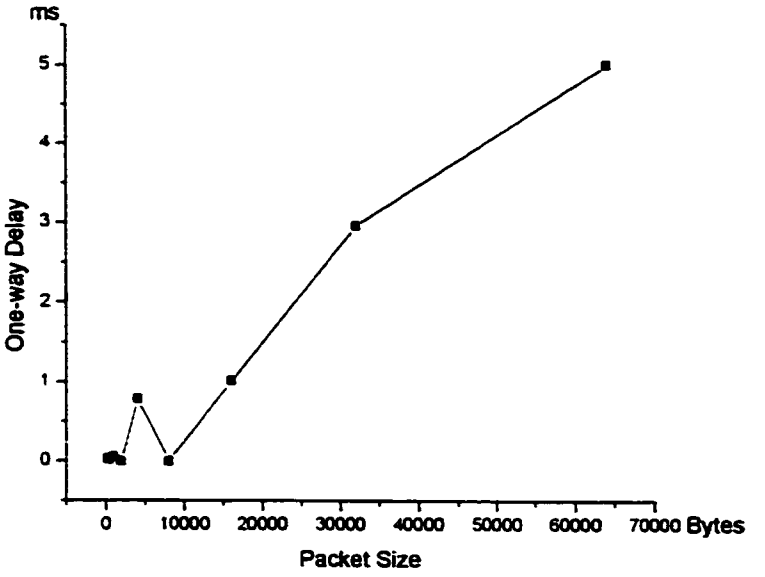


Figure 9.29 Hub: One-way Delay vs. UDP Packet Size

9.4.1.3.Congestion

I send six RTP flows among six PCs with fixed packet sizes (64000 bytes) but different send rates. In each test, the six flows have the same send rate.

Pair	Endpoint 1	Endpoint 2
1	192.168.1.2	192.168.1.3
2	192.168.1.4	192.168.1.5
3	192.168.1.6	192.168.1.7
4	192.168.1.3	192.168.1.4
5	192.168.1.5	192.168.1.6
6	192.168.1.7	192.168.1.2

Table 9.17 Flows for Hub Congestion Test

Offered Load (Mbps)	Throughput (Mbps)	Average One-way delay (ms)	Loss Rate (%)	Jitter (ms)
6x6=36	35.902	6.625	0.305	0.216
8x6=48	47.421	7.419	0.833	0.444
10x6=60	58.103	9.583	2.664	1.505
12x6=72	66.864	8.300	6.667	2.220
13x6=78	69.370	9.221	10.131	2.371
14x6=84	69.578	14.26	16.273	2.699
15x6=90	71.872	14.882	18.743	3.786
17x6=102	77.263	17.328	17.092	4.334
18x6=108	55.103	19.814	41.111	4.857
20x6=120	32.952	22.600	65.023	5.533

50x6=300	17.252	17.837	81.835	3.210
100x6=600	11.875	44.396	87.534	7.637

Table 9.18 Hub: Offered Load vs. Performance

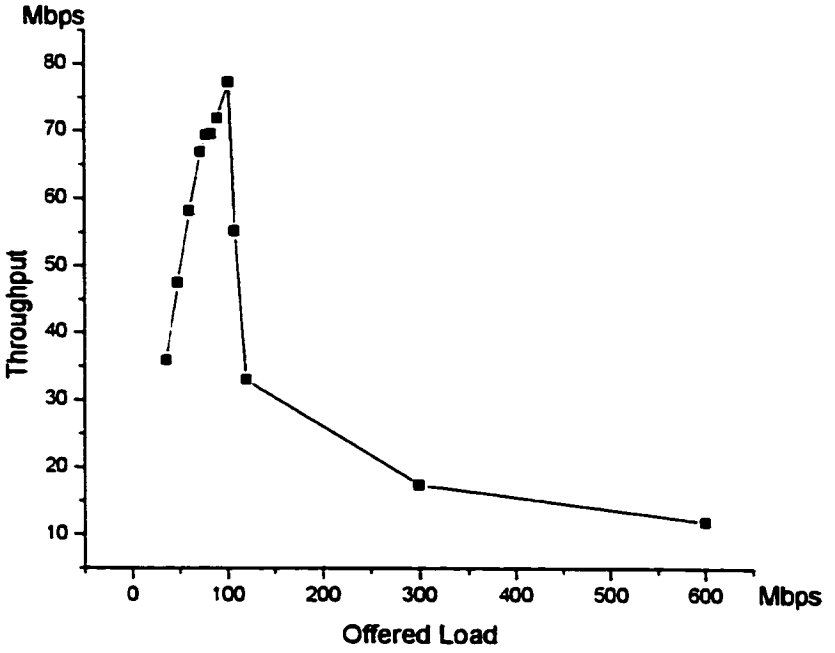


Figure 9.30 Hub: Throughput and Congestion

As the offered load increases, the throughput of the Ethernet increases until it reaches the peak throughput according to the packet size (in the case of 64Kbytes packet size, the peak is about 79Mbps). If the offered load continues to increase, the throughput is dropped due to collision followed by congestion.

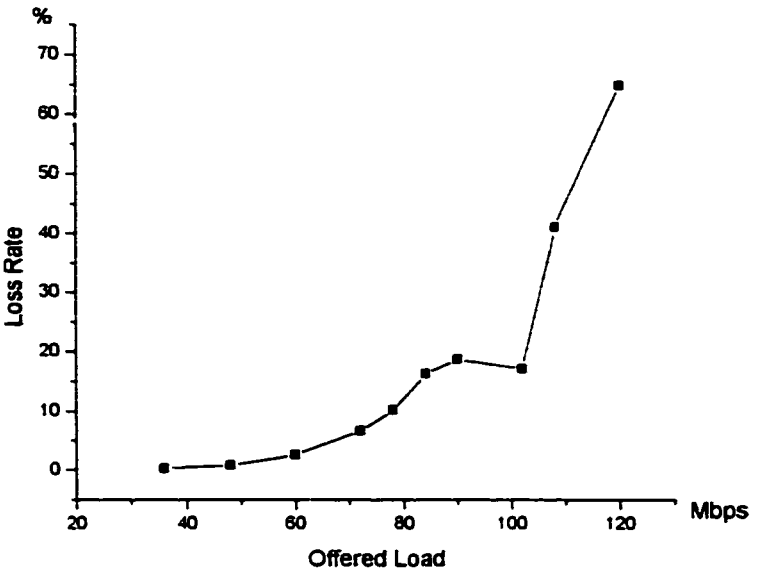


Figure 9.31 Hub: Loss Rate and Congestion

The loss rate of the network also increases with an increase in the offered load.

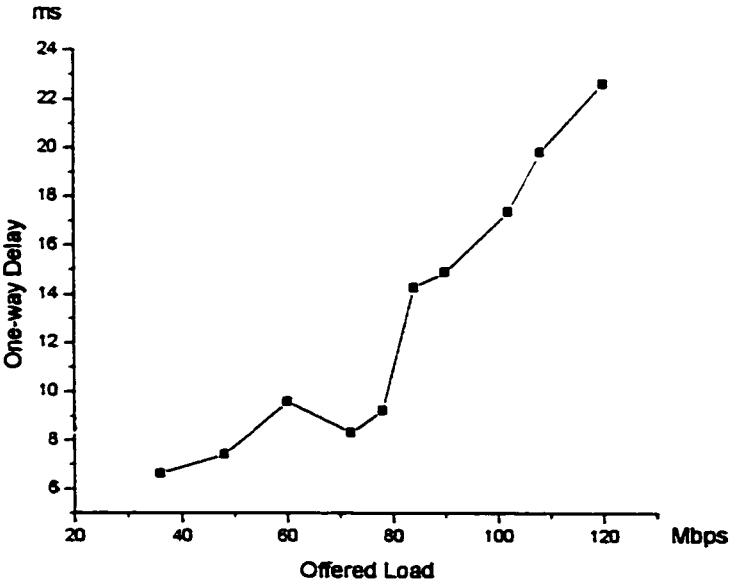


Figure 9.32 Hub: One-way Delay and Congestion

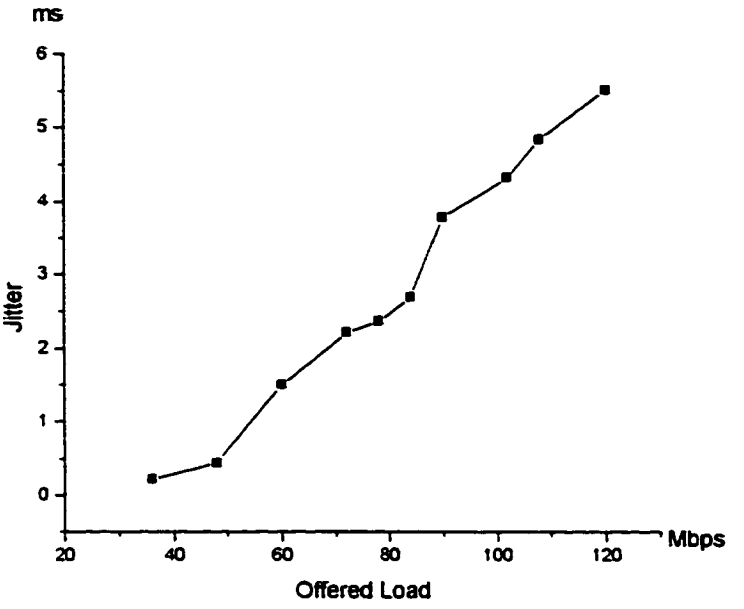


Figure 9.33 Hub: Jitter and Congestion

Additionally, delay and jitter for every individual packet increases with the increase in the offered load.

9.4.2. Switch based Ethernet

The end-to-end latency through an Ethernet network based on a thin cable or hubs depends on the network load. However, this has changed with the introduction of Ethernet switches and their new priority features.

9.4.2.1.Connecting the Ethernet

For the Ethernet connection, refer to Figure 9.34, which shows the replacement of the hub with a Cisco Catalyst 3524XL 24-ports 10/100Mbps Switch [121].

Catalyst 3524XL switch is the NetworkWorld *Reader's Choice Award* winner. It is a scalable line of stackable 10/100 and Gigabit Ethernet switches that support IEEE 802.1p and IEEE 802.3x (refer to 6.3.4).

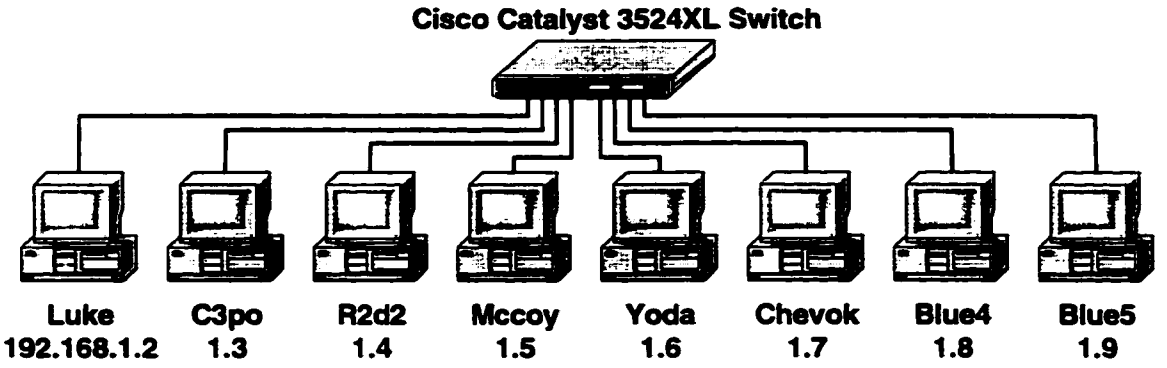


Figure 9.34 Switch connected Ethernet

9.4.2.2.Performance related parameters

Three RTP flows are sent from 192.168.1.2(Luke) to 192.168.1.3(C3po) with different UDP packet sizes ranging from 250 to 64000 bytes with unlimited send rates (>100Mbps).

Packet Size (bytes)	Throughput sum of the flow (Mbps)	One-way Delay Average (ms)	Loss Rate Average (%)	Jitter Average(ms)
1024	42.911	1.001	0.002	0.000
4000	60.278	1.004	0.004	0.000
8000	69.449	1.676	0.000	0.000
16000	76.098	2.239	0.001	0.000
32000	79.784	4.182	0.000	0.038
64000	84.456	11.180	0.000	0.001

Table 9.19 Hub: Offered Load vs. Performance

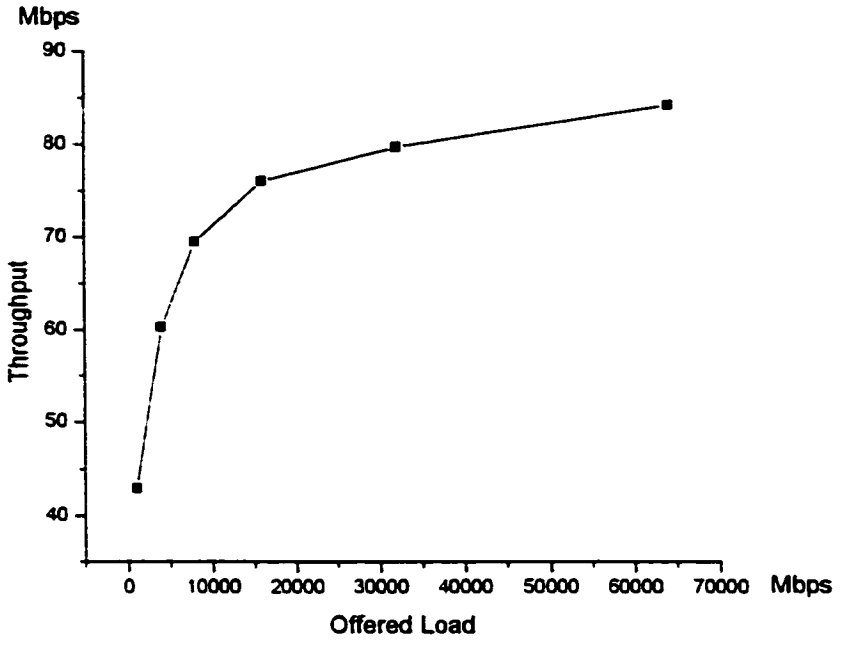


Figure 9.35 Switch: Throughput vs. UDP Packet Size

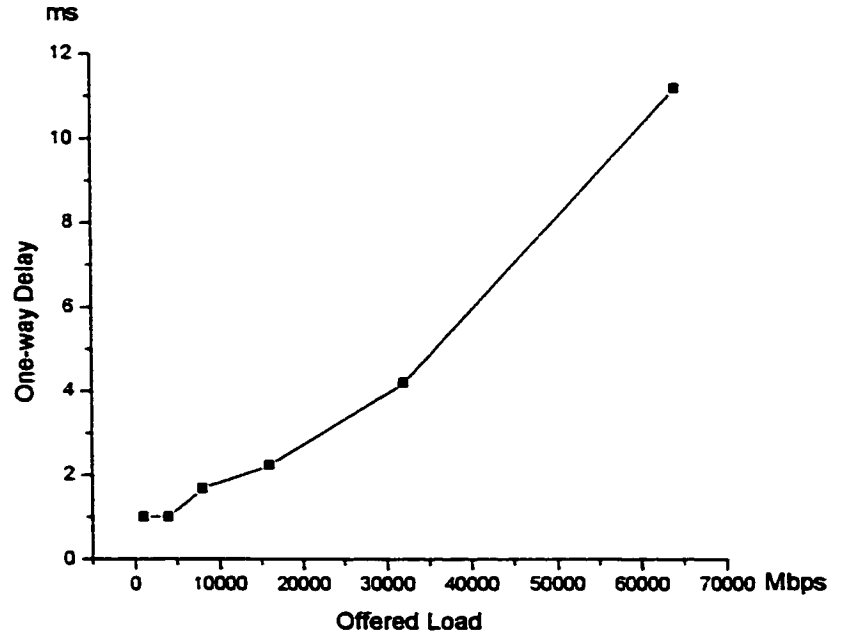


Figure 9.36 Switch: One-way Delay vs. UDP Packet Size

Similar to the Hub connected Ethernet, a bigger packet size causes higher throughput and longer delay.

When all flows are between two PCs, the throughput can reach 84.456Mbps with maximum UDP packet size of 64Kbytes.

9.4.3. Switch vs. Hub in Ethernet - Unicast

27 RTP flows are transmitted among 6 PCs with different packet sizes and different send rates in a Hub based Ethernet and a Switch based Ethernet respectively.

Pair	Endpoint1	Endpoint2	Protocol	Packet Size (bytes)
1	192.168.1.3	192.168.1.4	RTP	64000
2	192.168.1.2	192.168.1.5	RTP	
3	192.168.1.6	192.168.1.7	RTP	
4	192.168.1.3	192.168.1.4	RTP	32000
5	192.168.1.2	192.168.1.5	RTP	
6	192.168.1.6	192.168.1.7	RTP	
7	192.168.1.3	192.168.1.4	RTP	16000
8	192.168.1.2	192.168.1.5	RTP	
9	192.168.1.6	192.168.1.7	RTP	
10	192.168.1.3	192.168.1.4	RTP	8000
11	192.168.1.2	192.168.1.5	RTP	
12	192.168.1.6	192.168.1.7	RTP	
13	192.168.1.3	192.168.1.4	RTP	u[64,64000] Uniform distribution
14	192.168.1.2	192.168.1.5	RTP	

15	192.168.1.6	192.168.1.7	RTP	min. 64 bytes; max. 64000bytes
16	192.168.1.3	192.168.1.4	RTP	N[64,64000] Normal distribution min. 64 bytes; max. 64000bytes
17	192.168.1.2	192.168.1.5	RTP	
18	192.168.1.6	192.168.1.7	RTP	
19	192.168.1.3	192.168.1.4	RTP	P[64,64000] Poisson distribution min. 64 bytes; max. 64000bytes
20	192.168.1.2	192.168.1.5	RTP	
21	192.168.1.6	192.168.1.7	RTP	
22	192.168.1.3	192.168.1.4	RTP	E[64,64000] Exponential distribution. min. 64 bytes; max. 64000bytes
23	192.168.1.2	192.168.1.5	RTP	
24	192.168.1.6	192.168.1.7	RTP	
25	192.168.1.3	192.168.1.4	RTP	u[64,32000] Uniform distribution min. 64 bytes; max. 64000bytes
26	192.168.1.2	192.168.1.5	RTP	
27	192.168.1.6	192.168.1.7	RTP	

Table 9.20 Flows for Hub Switch Unicast Comparison

I change the send rate of the flows at each test and get following results shown in Table 9.21.

Send Rate	Throughput (Mbps)		One-way Delay (ms)		Loss Rate (%)		Jitter (ms)	
	Hub	Switch	Hub	Switch	Hub	Switch	Hub	Switch
1.5x27	39.934	—	4.964	—	0.267	—	0.360	—
2.2x27	58.766	—	4.842	—	0.397	—	0.525	—
2.5x27	66.430	—	5.621	—	0.698	—	0.684	—
2.7x27	64.856	—	7.360	—	10.014	—	1.106	—

3x27	40.677	—	9.040	—	49.084	—	1.620	—
4x27	17.122	106.530	29.541	3.257	81.001	0.000	5.284	0.151
8x27	6.624	209.924	47.635	7.825	92.937	0.000	10.860	0.476
20x27	12.471	263.480	—	15.223	—	0.000	—	0.608
100x27	3.631	265.614	46.011	12.920	96.128	0.000	9.164	0.620
200x27	—	265.735	—	13.187	—	0.000	—	0.616

Table 9.21 Hub vs. Switch: Unicast

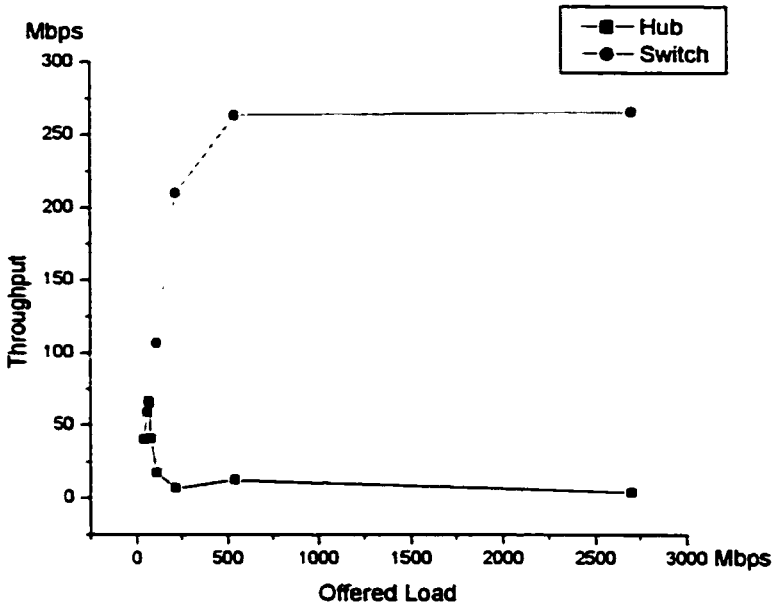


Figure 9.37 Hub vs. Switch: Unicast Throughput

100Mbps bandwidth of a Fast Ethernet Hub cannot be fully used. When the offered load exceeds its capacity,collision occurs and throughput drops quickly to almost 0.

A Fast Ethernet Switch has a bandwidth capacity that is far higher than 100Mbps (about 266Mbps in this case). When the offered load exceeds this capacity, the throughput will keep at the highest value and does not drop.

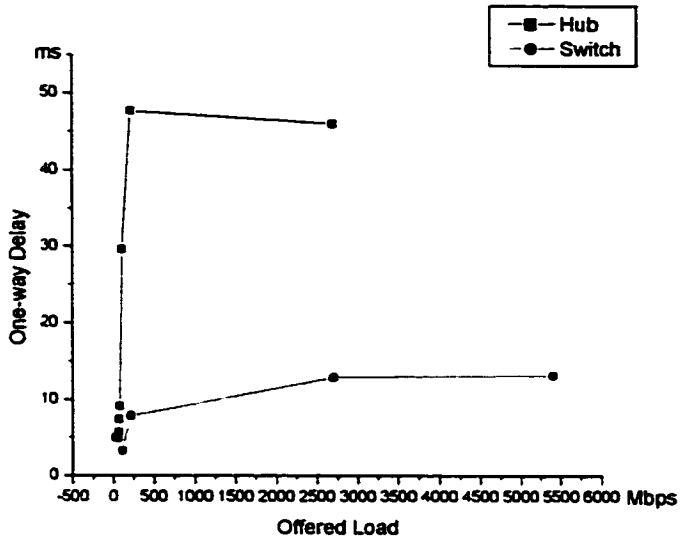


Figure 9.38 Hub vs. Switch: Unicast One-way Delay

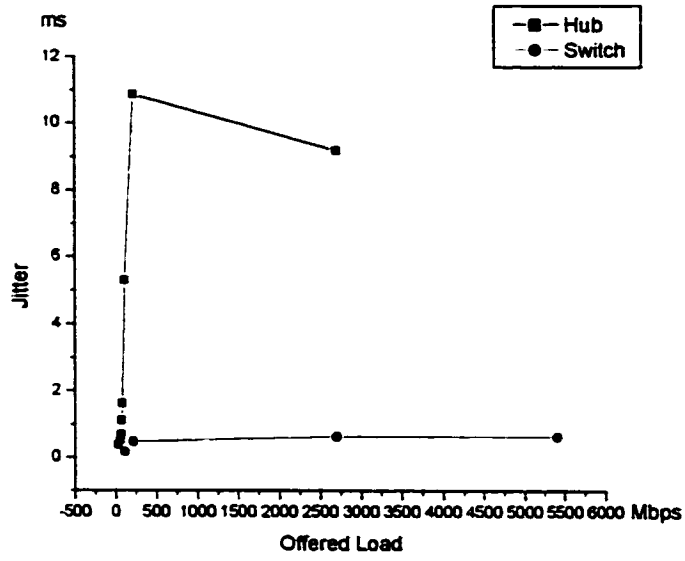


Figure 9.39 Hub vs. Switch: Unicast Jitter

Compared to the Hub, the Switch causes less delay and jitter to packets passing through it.

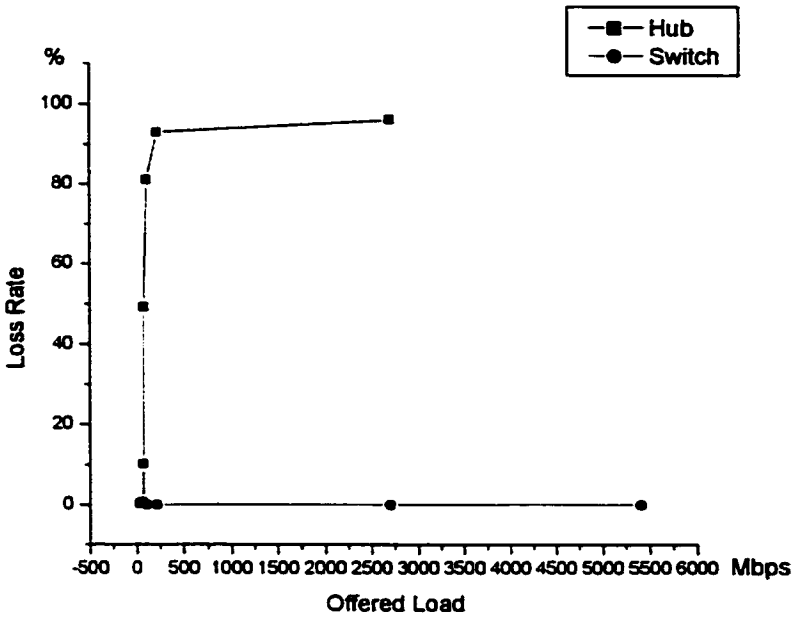


Figure 9.40 Hub vs. Switch: Unicast Loss Rate

When the offered load exceeds the hub's capacity, collision occurs and loss rate rises quickly to almost 100%.

When the offered load exceeds the switch's capacity, the loss rate is still almost 0. The IEEE 802.3x flow control feature contributes to this result. All the PCs connect to the switch directly so they work in full-duplex mode (refer to 6.3.4) The switch can send PAUSE packets to the PCs if the amount of packets received is more than the switch can handle, allowing the loss rate to be controlled.

Compared to a Hub, a switch has great advantages for high-performance network communication with high bandwidth, low delay and jitter and low loss rate requirements.

9.4.4. Switch vs. Hub in Ethernet - Multicast

Although a switch is very good at unicast communication, it still has some limitations for multicast communication. 6 multicast groups are used in this test with each group having 1 sender and 5 receivers. All flows are RTP flows and packet sizes 64000 bytes. The test is performed with different send rate in a Hub based Ethernet and a Switch based Ethernet respectively.

Multicast Address	Endpoint1	Endpoint2	Protocol	Packet Size
224.2.2.2:2222	192.168.1.2	3,4,5,6,7	RTP	64000
224.2.2.2:2224	192.168.1.3	2,4,5,6,7		
224.2.2.2:2226	192.168.1.4	2,3,5,6,7		
224.2.2.2:2228	192.168.1.5	2,3,4,6,7		
224.2.2.2:2230	192.168.1.6	2,3,4,5,7		
224.2.2.2:2232	192.168.1.7	2,3,4,5,6		

Table 9.22 Flows for Hub Switch Multicast Comparison

Offered Load (Grp)	Throughput (Mbps)		One-way Delay (ms)		Loss Rate (%)		Jitter (ms)	
	Hub	Switch	Hub	Switch	Hub	Switch	Hub	Switch
10	41.230	59.631	5.107	2.290	38.032	4.103	2.993	1.582
15	3.741	89.968	9.470	10.280	96.337	15.298	3.827	1.872
20	1.916	10.385	7.833	45.619	98.163	91.576	3.037	4.046
50	1.735	0.669	5.467	26.0	98.237	99.801	4.208	2.287
100	2.119	0.321	9.167	59.6	97.873	99.917	3.450	7.400

Table 9.23 Hub vs. Switch: Multicast

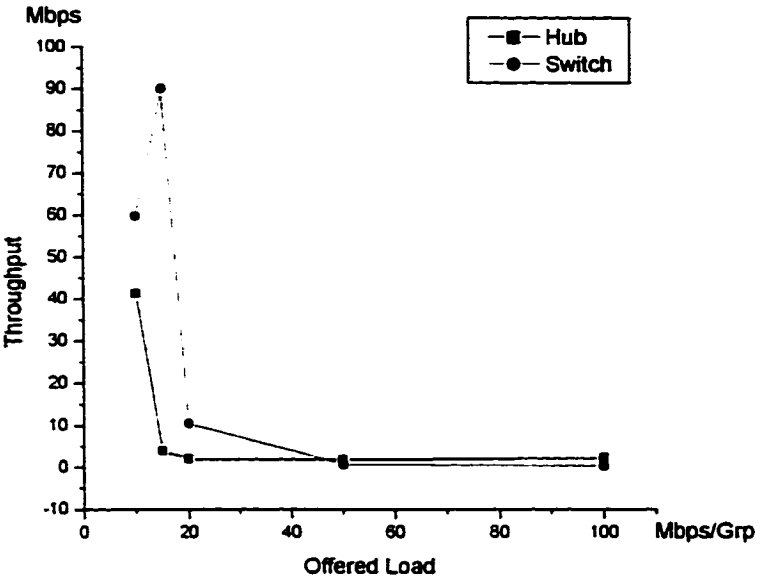


Figure 9.41 Hub vs. Switch: Multicast Throughput

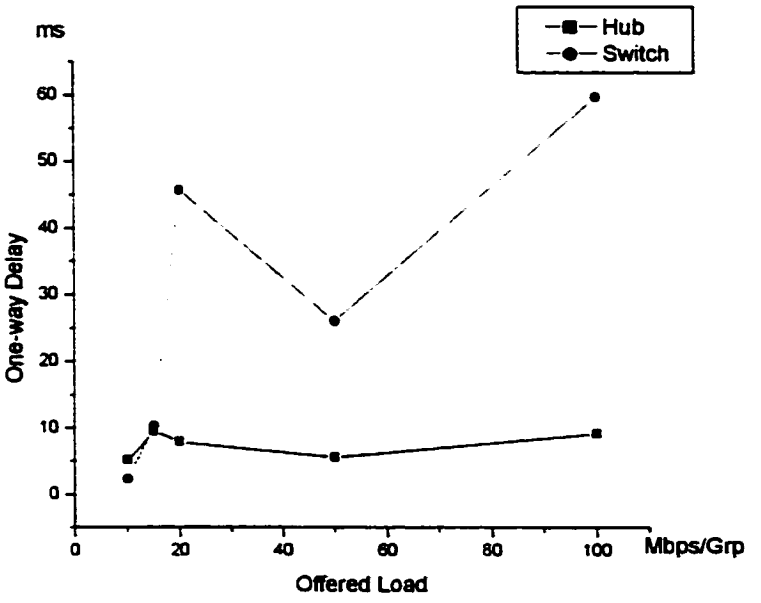


Figure 9.42 Hub vs. Switch: Multicast One-way Delay

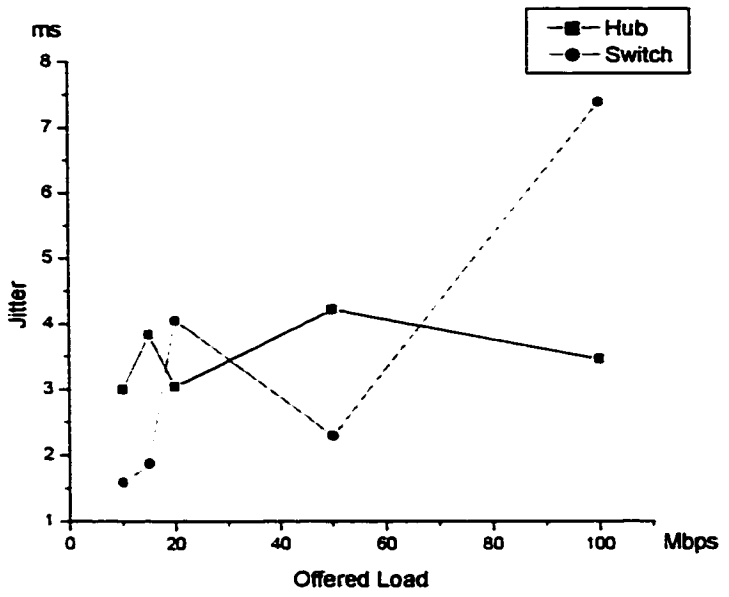


Figure 9.43 Hub vs. Switch: Multicast Jitter

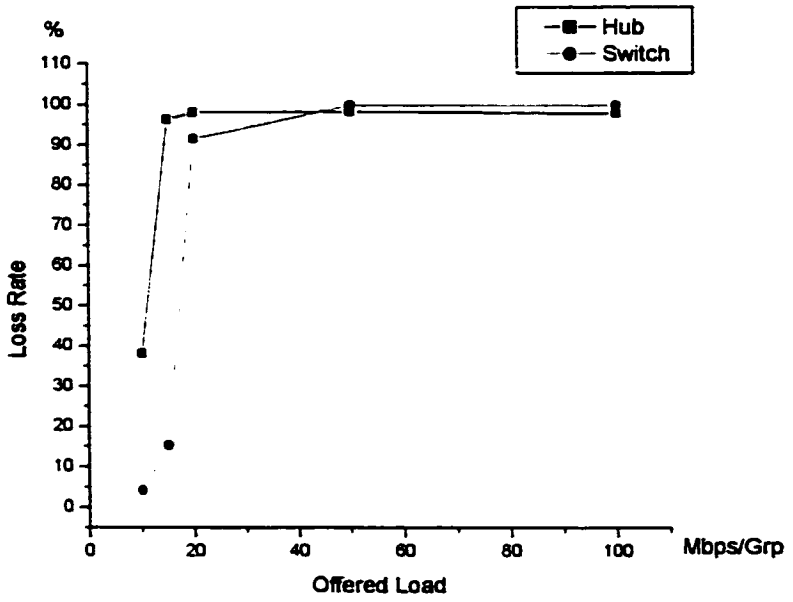


Figure 9.44 Hub vs. Switch: Multicast Loss Rate

For multicasting, the switch does not exhibit great advantages compared to the hub. Congestion occurs and loss rate reach almost 100% when offered load exceed its capacity. The reason is that the switch floods multicast traffic to all ports of the switch.

Recalling from section 6.3.4 that DMF can avoid multicast congestion, the importance of DMF becomes apparent from these measurements.

9.4.5. Switch in Ethernet – Many multicast groups

To explore the upper limits of multicasting capacity, we build a test scenario with 200 multicast groups containing 482 flows totally. (Group 1 ~ 152, each group two pairs; Group 153~192, each group three pairs; Group 193~200, each group seven pairs). All flows use 2048 byte packet sizes.

I tested with different send rates and identified the critical point of the send rate as 14Kbytes/sec. If the send rate is less than 14Kbytes/sec, flows experience the expected throughput, low loss rate, delay and jitter. If the send rate is larger than 14Kbytes/sec, the throughput drops down, while the loss rate, delay and jitter rise.

When the send rate is 14Kbytes/sec, we get the following results:

- Throughput:

  All groups have 14.8Kbytes/sec and the total overall throughput is 25.305Mbps

- One-way Delay:

  One-way delay for every flow is between 0~3.103msec (most of them (>90%) between 0~2msec)

- Lost Rate: 0.344%

Around 50 pairs of the 482 pairs have non-zero loss rate. The loss rate of these pairs is 2.553~3.191%.

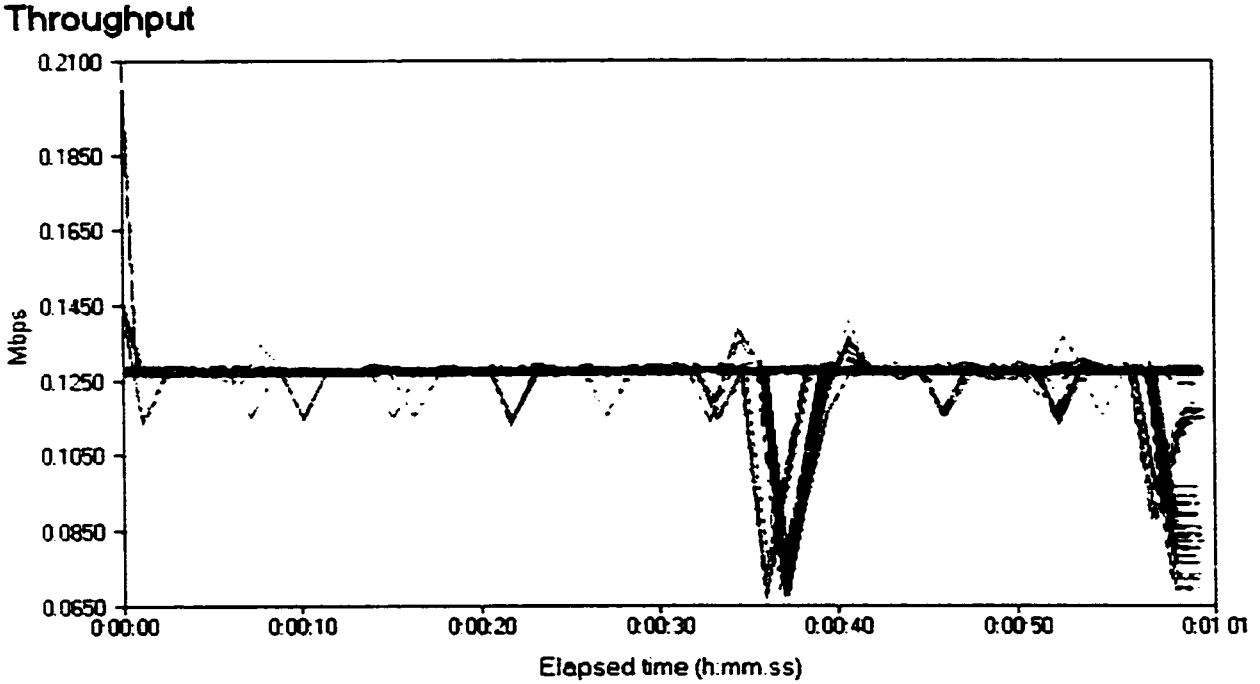


Figure 9.45 200 Multicast Groups in Switch: Throughput

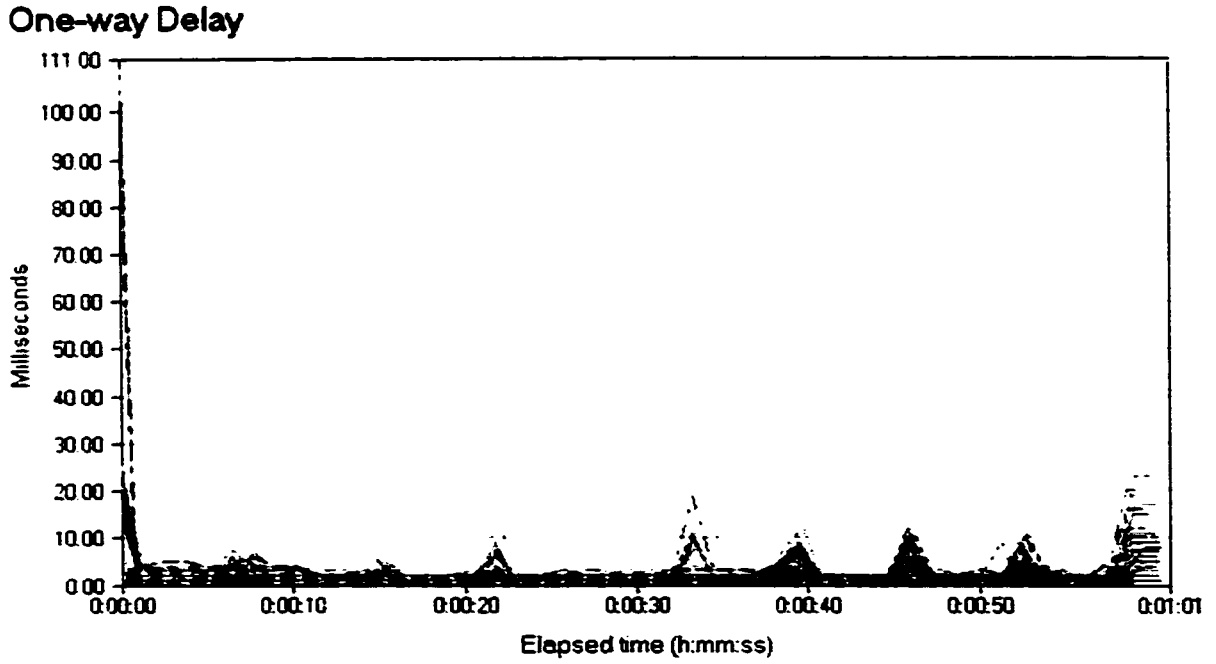


Figure 9.46 200 Multicast Groups in Switch: One-way Delay

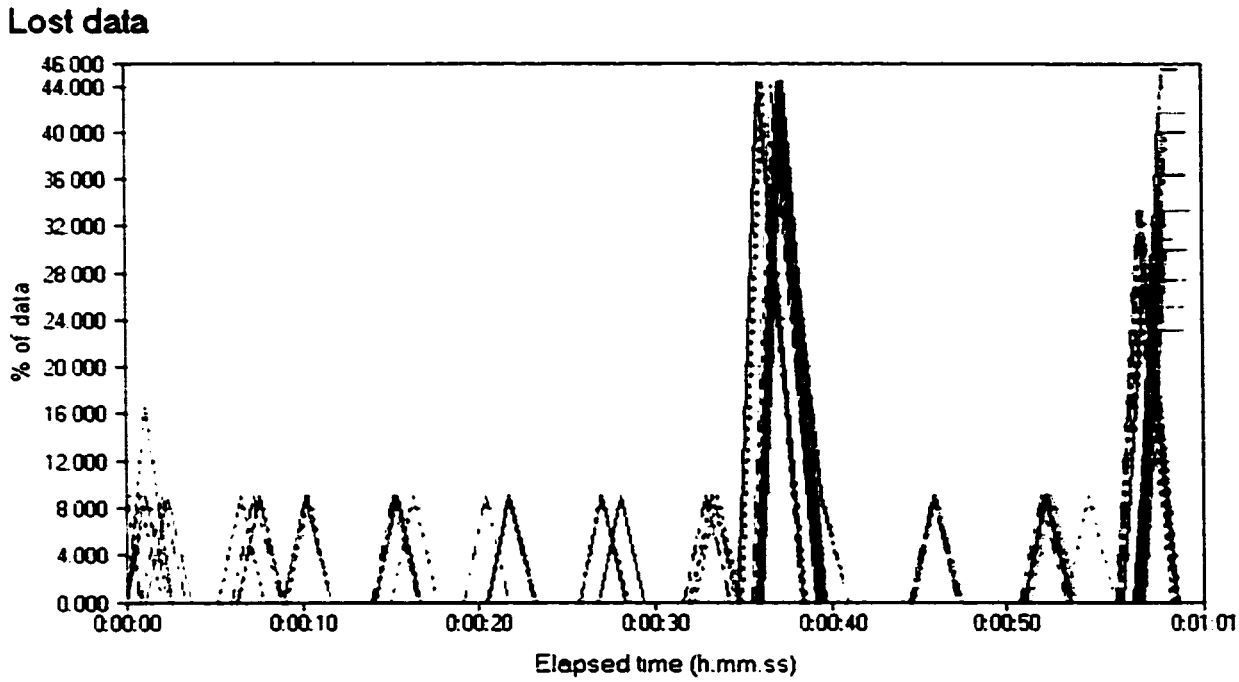


Figure 9.47 200 Multicast Groups in Switch: Loss rate

Chapter 10
Building QoS Networks for RT-DIS

10.1. Build QoS-enabled Campus Networks

10.1.1. Campus Network

A campus network is a network deployed among one or many co-located buildings but in an autonomous domain. According to this definition of campus network, Enterprise networks and Access ISP networks are also campus networks.

The difference between an access ISP and campus/enterprise network typically appears at the last mile – access method. Campus/Enterprise network users usually connect to the backbone through LAN while access ISP users connect to the backbone through dial-up connections such as a 56K modem, DSL/Cable modem, etc.

Here we focus on building a QoS-enabled Campus network that consists of many LANs. For access ISPs, some commercial QoS-enabled DSL/Cable systems have extended QoS to the last mile [122,123].

10.1.2. Cell vs. Packet

Three popular architectures were used in campus backbone design: Layer 2 FDDI backbone, Layer 3 IP Router backbone, and ATM backbone.Since multi-layer switches have erased the gap between Layer 2 devices and Layer 3 devices, it is reasonable to divide campus network solutions into two branches now: *Cell backbone* and *Packet backbone*. A cell backbone uses ATM switches as backbone nodes, while a packet backbone uses multi-layer switches as its backbone nodes.Until recently, ATM was the only switching technology able to deliver high capacity and scalable bandwidth with the promise of end-to-end QoS spanning the local area, campus, and wide area networks and to cross international boundaries seamlessly. But with the deployment of Gigabit Ethernet, 10GigaEthernet and IEEE 802.3ad Link Aggregation technology (6.3.4.3), the promise of end-to-end seamless integration with high capacity and scalable bandwidth will be possible with packet networks.

Comparison	ATM Switch	Multi-layer Switch
Bandwidth Scalability (Mbps)	1.544, 25.6, 44.736, 100, 155.52, 622.08, 2400	10, 100, 1000, 10000 with any number among them depending on 802.3ad
QoS	Excellent	Good
Cost [124]	**ATM Switch:** **155Mbps I/F:** $900/port and $3,500/uplink **622Mbps I/F:** $5,000/port and $10,000/uplink	**Multi-layer Switch:** **Gigabit Ethernet I/F:** $1,500/port Layer 2 and $3,000/port Layer 3
Management	Complex	Simple

Table 10.1 ATM Switch vs. Multi-layer Switch

With the deployment of multi-layer switches which can handle not only MAC layer QoS (section 6.3.4) but also IP layer and even higher layers QoS (section 6.3.5), the promise of end-to-end QoS will be possible with packet networks.

The analysis above also is suited for LAN systems where ATM desktop switches are compared to multi-layer desktop switches. Due to the high cost and complex management of ATM systems, a cell backbone and a cell LAN is only used in environments that demand strict QoS requirements such as safe-critical systems and applications.

For general purpose distributed real-time simulation, packet backbone and packet LAN will be the best choice.

10.1.3. Checklist for QoS Campus backbone

Figure 10.1 shows the architecture of an example of campus network built with multi-layer switches. To get the benefits of high performance, QoS and bandwidth scalability, you should choose the switches that support following features:

- Wire-speed forwarding
- Multi-Layer capability up to Layer 4
- 802.1p Traffic Class Support. (The best choose is that the switch supports eight queues for each port)
- 802.1p Dynamic Multicast Filtering
- 802.3x flow control
- 802.3ad link aggregation
- DiffServ

- RSVP
- RSVP Scalability Enhancement

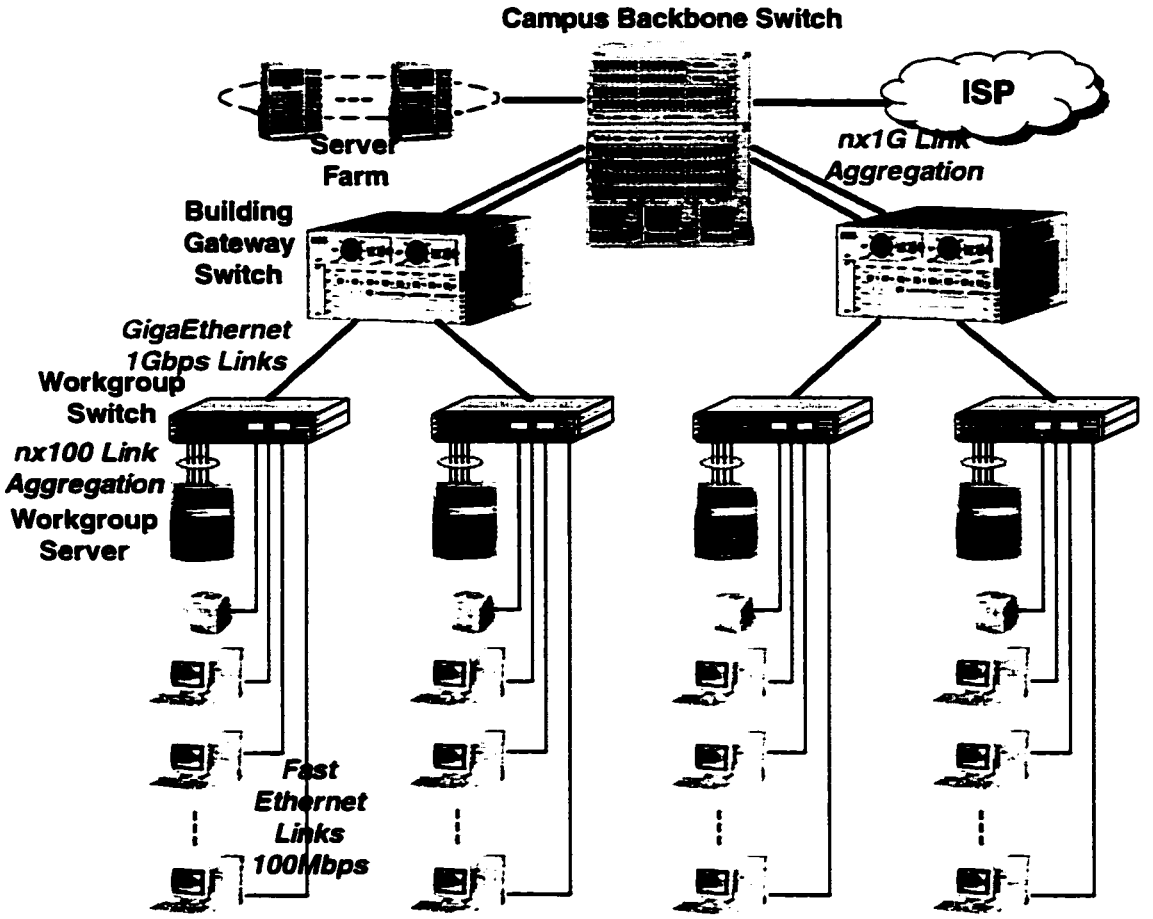


Figure 10.1 Campus Network Example

As for end systems, the following should be ensured:

- NICs are 802.1p, 802.3x compliant. For example, 3Com EtherLink 10/100 PCI NIC.
- Operating Systems support the QoS mechanisms and necessary mapping approach. For example, Microsoft Windows 2000. The Microsoft Windows 2000 series operating systems supports 802.1p, RSVP, DiffServ and Mapping approach among these different QoS mechanisms (Table 10.2) [125]:

Service Type	DSCP	802.1p
Network Control	30	7
Guaranteed Service	28	5
Controlled Load	18	3
Qualitative	0	0
All other traffic	0	0

Table 10.2 DSCP to 802.1p Mapping

10.1.4. Configuring QoS on Campus

Referring to sections 9.1, a network administrator can configure the campus network to support an end-to-end RSVP QoS scheme. If the campus network is a large one, the administrator can enable the *RSVP scalability enhancement* at workgroup switches or building gateway switches (refer to 9.2 and 9.3). Using flow-based RSVP at LANs or buildings and using aggregated RSVP at campus backbones suits for large-scale campus networks.

10.2. Backbone QoS – Will be there tomorrow

10.2.1. IP Carrier Network Schemes

It is becoming clear that two technologies are going to be deployed over the Internet Backbone – MPLS and IP over DWDM.

Almost all National ISPs including Worldcom, AT&T, Genuity, Cable & Wireless, Infonet, are deploying or have deployed MPLS over their carrier networks.

WorldCom is using MPLS for traffic engineering over its UUNET Internet backbone. WorldCom already lets users dedicate bandwidth to specific applications or user groups but is looking at using a version of DiffServ to offer users multi-vendor QoS support.

Sprint is one of the only large ISPs that is not deploying MPLS [126]. Instead, Sprint promises its network will be IP over DWDM no later than next January [126]. Sprint is also looking at technologies such as DiffServ for QoS [126] and may adopt a WDM QoS technology such as fiber delay lines (FDLs) [127], which is quiet different than current IETF QoS schemes.

10.2.2. Current SLAs

Many carriers are providing SLAs for their service, some examples are shown in Table 10.3

National ISPs	Round-trip latency	Packet loss	Network availability
Worldcom	< 65msec	< 1%	100%
Cable&Wireless	< 55msec	< 1%	100%
AT&T	< 60 msec	< 0.7%	99.99%
Sprint	< 70 msec	< 0.3%	100%
Genuity	< 55msec	< 0.5%	—

Table 10.3 Carrier ISPs' SLAs [128,129]

Although the latency in SLAs means average latency, it is a big step toward providing the latency guarantee.

10.2.3. Single-carrier QoS SLA

Carrier ISPs will soon provide QoS SLA to their access service customers.For example. Infonet will offer IP service classes in fall 2001 after it finishes building its MPLS backbone. They promise to offer four service classes, each of which carries an associated SLA for delay, packet loss and jitter [130].

10.2.4. Multi-carrier QoS SLA

Usually, an ISP network is an Autonomous System (AS) and the configuration of a QoS enabled path from end-to-end through multiple ISP networks is a big challenge.

SLAs and differentiated service classes across a multi-carrier public Internet service do not yet exist but ISPs are working towards this objective.

The IETF is working on a specification for DiffServ to let carriers set QoS levels that span multiple ISP networks. It could be the first incarnation of a multiple-network SLA [131].

Cisco has begun to provide DiffServ over MPLS solutions with its new IOS feature "DiffServ-aware MPLS Traffic Engineering (DS-TE)" [85,132].

10.3. The Coming End-to-End QoS Enabled Internet

QoS is not only a standard effort anymore but a real one. From IEEE 802 Ethernet QoS mechanisms (802.1p, 802.3ad, 802.3x) to IETF Internet QoS solutions (IntServ/RSVP, DiffServ, MPLS), all standards have been implemented in some way by some network device vendors. Based on these QoS-enabled network devices, it is not difficult to build a QoS network with guaranteed bandwidth, delay, jitter and loss rate.

The networks with heterogeneous QoS solutions are also capable of converting QoS information and keep the end-to-end QoS guarantee. Technically, the QoS is ready for deployment. It will be deployed first at campus networks since it does not face as many political issues. Carrier ISPs are also paying much attention to this field and some pioneers have started the first steps, giving rise to the idea of QoS emerging as the new revolution of the Internet.

Chapter 11
Conclusions and Future Work

11.1. Conclusions and Contribution of Thesis

As a new DIS standard that was inspired from all old ones, HLA provides a set comprehensive management functions for DIS. HLA supports efficient management of shared space of DVE through a set of functions including: federation management, declaration management, object management, ownership management, time management and data distribution management. The first part of this thesis illustrated how to manage the shared space using HLA through a distributed virtual shopping mall application.

HLA has powerful features to support a very wide range of DIS/DVE applications, but there are still some limitations for shared space management. At first, the HLA standard does not support native stream transmission and control that is critical for the immersion feeling of DVE. More importantly, HLA does not address RT-DIS and QoS issues which is very important for mission-critical and safety-critical DIS/DVE applications.

The second part of this thesis presents a comprehensive solution for stream transmission and control over HLA-RTI.

For stream transmission, depending on the system interval timer to control the schedule, RTI-API *sendInteraction()* and *receiveInteraction()* are used to transfer a data block. This approach provides a flexible way for audio/video stream retrieving and an interactive real-time audio/video application. Our implementation proves that the approach can support CD quality audio transmission with less than 250msec latency in a LAN environment.

As for stream control, a stream source publishes stream parameters using object instances so that all stream receivers can query and get these parameters at any time. Then the receivers can join the federation at any time and may receive the stream in any middle point. This solution is good at playing an audio program in a DVE application.

As the major contribution, this thesis presents a Real-Time Extension to HLA and a proposal for Real-Time RTI.

The realization of RT-DIS depends on the end-to-end predictability as a combination of the end system predictability and the network predictability. End system predictability can be satisfied with RTOS mechanisms such as preemptive priority multitask, priority message queue, etc. Network predictability can be satisfied with network QoS mechanisms such as IntServ, DiffServ and MPLS. For flexibility and scalability purposes, it is possible to use these mechanisms together.

The underlying operating system and network are ready to provide predicable services to RT-DIS applications. HLS should extend itself to standardize the way of using these advanced features in RT-DIS systems.

The major extension to HLA involves OMT specification. The attribute table and parameter table of OMT should be extended so that federates can express the QoS requirements for the attributes and parameters. The transportation table should also be extended to support more transportation types to satisfy the communication requirements of RT-DIS applications.

When we present the HLA real-time extension, we considered much about feasibility of its implementation – RT-RTI. RT-RTI demands a different architecture comparing to RTI-NG. To improve its real-time performance and scalability, RT-RTI must be fully multithreaded with a thread-pool core. At the same time, RT-RTI should take advantage of the preemptive priority mechanism of operating systems and the QoS mechanism of the IP network.

Similar to the growth of real-time CORBA after the mature based CORBA standard suite, Real-Time HLA is a natural extension following the standardization of HLA as IEEE 1516 in September of 2000.

In the fourth part, we presented performance measurements of different QoS networks and presented a practical network solution with QoS support.

This part shows that RSVP-based IntServ, DiffServ can guarantee QoS of specified network traffic even in congestion situations. The integrated solution – RSVP over DiffServ can provide excellent QoS and scalability that is the best choose for RT-DIS over a wide area.

To build a QoS-enabled network for RT-DIS, we recommend multi-layer switches with 802.1p, 802.3x, 802.3ad, DiffServ, RSVP, RSVP scalability enhancement features enabled.

11.2. Future Work

There are still some work worth to do for the HLA Real-Time Extension and Real-Time RTI:

- Comprehensive mapping between application level QoS parameters to underlying OS priority and network QoS parameters. The transition from the empirical mapping to a sophisticated mapping algorithm presents an interesting topic of research.
- Reliable multicast protocols in a QoS-enabled network are also of importance. Existing reliable multicast protocols should be simplified to just monitor and solve the packet loss due to bit error since a queuing loss can be solved by QoS mechanisms.
- Dynamic simulation load balancing is yet another area of interest. Large-scale entity-based simulations generate non-deterministic system resource consumption levels that can exceed the performance capabilities of individual simulation processing nodes or their interconnecting network communication components. Load balance approaches are worthy to explore for avoiding excessive resource consumption peaks.
- In a QoS enabled network, QoS adaptation is still required to balance user requirements and cost. How to port existing QoS adaptation approaches [100] for best-effort network QoS-enabled network is worthy of exploration as well.
- Security is a hot topic in the IP world. Security requirements of RT-DIS can be satisfied along with IP security solutions.

Appendix A. Router Configurations

1. RSVP-enabled Unicast Network

RA configuration

```
interface FastEthernet0/0
        ip address 192.168.1.1 255.255.255.0
        speed auto
        full-duplex
        fair-queue 64 64 235
        ip rsvp bandwidth 7500 7500
!
interface Serial0/0
 ip address 192.168.2.1 255.255.255.0
        fair-queue 64 64 37
        clockrate 4000000
        Ip rsvp bandwidth 1158 1158
!
Ip classless
ip route 0.0.0.0 0.0.0.0 192.168.2.2
```

RB configuration

```
interface Ethernet0/0
 ip address 192.168.7.1 255.255.255.0
        full-duplex
        fair-queue 64 64 235
        ip rsvp bandwidth 7500 7500
!
interface Serial0/0
        ip address 192.168.3.1 255.255.255.0
        fair-queue 64 64 37
```

```
        clockrate 4000000
        ip rsvp bandwidth 1158 1158
!
interface Serial0/1
        ip address 192.168.2.2 255.255.255.0
        fair-queue 64 64 37
        ip rsvp bandwidth 1158 1158
!
ip classless
ip route 192.168.1.0 255.255.255.0 192.168.2.1
ip route 192.168.4.0 255.255.255.0 192.168.3.2
ip route 192.168.5.0 255.255.255.0 192.168.3.2
ip route 192.168.6.0 255.255.255.0 192.168.3.2
ip route 192.168.8.0 255.255.255.0 192.168.3.2
```

RC Configuration

```
interface Serial0/0
        ip address 192.168.3.2 255.255.255.0
        fair-queue 64 64 37
        ip rsvp bandwidth 1158 1158
!
interface Serial0/1
        ip address 192.168.4.1 255.255.255.0
        fair-queue 64 64 37
        ip rsvp bandwidth 1158 1158
!
ip classless
ip route 192.168.1.0 255.255.255.0 192.168.3.1
ip route 192.168.2.0 255.255.255.0 192.168.3.1
ip route 192.168.5.0 255.255.255.0 192.168.4.2
ip route 192.168.6.0 255.255.255.0 192.168.4.2
ip route 192.168.7.0 255.255.255.0 192.168.3.1
ip route 192.168.8.0 255.255.255.0 192.168.4.2
```

RD Configuration

```
interface Ethernet0/0
        ip address 192.168.8.1 255.255.255.0
        full-duplex
         fair-queue 64 64 235
        ip rsvp bandwidth 7500 7500
!
interface Serial0/0
        ip address 192.168.4.2 255.255.255.0
        fair-queue 64 64 37
        clockrate 4000000
        ip rsvp bandwidth 1158 1158
!
interface Serial0/1
        ip address 192.168.5.1 255.255.255.0
        fair-queue 64 64 37
        ip rsvp bandwidth 1158 1158
!
ip classless
ip route 192.168.1.0 255.255.255.0 192.168.4.1
ip route 192.168.2.0 255.255.255.0 192.168.4.1
ip route 192.168.3.0 255.255.255.0 192.168.4.1
ip route 192.168.6.0 255.255.255.0 192.168.5.2
ip route 192.168.7.0 255.255.255.0 192.168.4.1
```

RE Configuration

```
interface FastEthernet0/0
        ip address 192.168.6.1 255.255.255.0
        speed auto
        full-duplex
```

```
        fair-queue 64 64 235
        ip rsvp bandwidth 7500 7500
!
interface Serial0/0
        ip address 192.168.5.2 255.255.255.0
        fair-queue 64 64 37
        clockrate 4000000
        ip rsvp bandwidth 1158 1158
!
ip classless
ip route 0.0.0.0 0.0.0.0 192.168.5.1
```

2. RSVP-enabled Multicast Network

RA Configuration

```
interface FastEthernet0/0
        ip address 192.168.1.1 255.255.255.0
        ip pim dense-mode
        speed auto
        full-duplex
        fair-queue 64 64 235
        ip rsvp bandwidth 7500 7500
!
interface Serial0/0
        ip address 192.168.2.1 255.255.255.0
        ip pim dense-mode
        fair-queue 64 64 37
        clockrate 4000000
        ip rsvp bandwidth 1158 1158
!
```

RB Configuration

```
interface Serial0/0
        ip address 192.168.3.1 255.255.255.0
        ip pim dense-mode
        fair-queue 64 64 37
        clockrate 4000000
        ip rsvp bandwidth 1158 1158
!
interface Serial0/1
        ip address 192.168.2.2 255.255.255.0
        ip pim dense-mode
        fair-queue 64 64 37
        ip rsvp bandwidth 1158 1158
!
interface Serial1/0
        ip address 192.168.4.1 255.255.255.0
        ip pim dense-mode
        fair-queue 64 64 48
        clockrate 4000000
        ip rsvp bandwidth 1536 1536
!
```

RC Configuration

```
interface Ethernet0/0
        ip address 192.168.7.1 255.255.255.0
        ip pim dense-mode
        full-duplex
        fair-queue 64 64 235
        ip rsvp bandwidth 7500 7500
!
interface Serial0/0
        ip address 192.168.3.2 255.255.255.0
```

```
        ip pim dense-mode
        fair-queue 64 64 37
        ip rsvp bandwidth 1158 1158
!
```

RD Configuration

```
interface Serial0/1
        ip address 192.168.5.1 255.255.255.0
        ip pim dense-mode
        fair-queue 64 64 37
        ip rsvp bandwidth 1158 1158
!
interface Serial1/0
        ip address 192.168.4.2 255.255.255.0
        ip pim dense-mode
        fair-queue 64 64 48
        ip rsvp bandwidth 1536 1536
```

RE Configuration

```
interface FastEthernet0/0
        ip address 192.168.6.1 255.255.255.0
        ip pim dense-mode
        speed auto
        full-duplex
        fair-queue 64 64 235
        ip rsvp bandwidth 7500 7500
!
interface Serial0/0
        ip address 192.168.5.2 255.255.255.0
        ip pim dense-mode
        fair-queue 64 64 37
```

```
        clockrate 4000000
        ip rsvp bandwidth 1158 1158
!
```

3. DiffServ Network

The routing parts of the configuration are the same as Appendix 1 and hence are omitted here.

RA Configuration

```
!
class-map match-all platinum
        match ip dscp 46
class-map match-all gold
        match ip dscp 10  12  14
class-map match-all silver
        match ip dscp 18  20  22
class-map match-all bronze
        match ip dscp 26  28  30
class-map match-all iron
        match ip dscp 34  36  38
!
policy-map DiffServ
        class platinum
                priority 500
        class gold
                bandwidth remaining percent 50
                random-detect dscp-based
        class silver
                bandwidth remaining percent 25
                random-detect dscp-based
        class bronze
```

```
            bandwidth remaining percent 15
            random-detect dscp-based
        class iron
            bandwidth remaining percent 10
            random-detect dscp-based
!
interface FastEthernet0/0
 ip address 192.168.1.1 255.255.255.0
      speed auto
      full-duplex
      service-policy output DiffServ
!
interface Serial0/0
      ip address 192.168.2.1 255.255.255.0
      service-policy output DiffServ
      clockrate 4000000
!
```

RB Configuration

```
!
# Same class-map and policy-map as RA Configuration
!
interface Ethernet0/0
 ip address 192.168.7.1 255.255.255.0
 full-duplex
!
interface Serial0/0
 ip address 192.168.3.1 255.255.255.0
 service-policy output DiffServ
 clockrate 4000000
!
interface Serial0/1
 ip address 192.168.2.2 255.255.255.0
```

```
 service-policy output DiffServ
!
```

RC Configuration

```
!
# Same class-map and policy-map as RA Configuration
!
interface Serial0/0
        ip address 192.168.3.2 255.255.255.0
        service-policy output DiffServ
!
interface Serial0/1
        ip address 192.168.4.1 255.255.255.0
        service-policy output DiffServ
!
```

RD Configuration

```
!
# Same class-map and policy-map as RA Configuration
!
interface Ethernet0/0
        ip address 192.168.8.1 255.255.255.0
        full-duplex
!
interface Serial0/0
        ip address 192.168.4.2 255.255.255.0
        service-policy output DiffServ
        clockrate 4000000
!
interface Serial0/1
        ip address 192.168.5.1 255.255.255.0
```

```
        service-policy output DiffServ
!
```

RE Configuration

```
!
# Same class-map and policy-map as RA Configuration
!
interface FastEthernet0/0
        ip address 192.168.6.1 255.255.255.0
        speed auto
        full-duplex
        service-policy output DiffServ
!
interface Serial0/0
        ip address 192.168.5.2 255.255.255.0
        service-policy output DiffServ
        clockrate 4000000
!
```

4. DiffServ Network via Access-List

The router configuration is the same as Appendix 3 except for the classification method. In Appendix 3, classification is implemented according to DSCP field but here the classification is replaced with "*access-list*" based method.

```
!
access-list 101 permit udp any any range 60000 65535
access-list 102 permit udp any any range 50000 59999
access-list 103 permit udp any any range 40000 49999
access-list 104 permit udp any any range 30000 39999
access-list 105 permit udp any any range 20000 29999
!
class-map match-all platinum
        match access-group 101
class-map match-all gold
        match access-group 102
class-map match-all silver
        match access-group 103
class-map match-all bronze
        match access-group 104
class-map match-all iron
        match access-group 105
!
```

5. RSVP Scalability Enhancement

RA Configuration

```
interface FastEthernet0/0
        ip address 192.168.1.1 255.255.255.0
        speed auto
```

```
        full-duplex
        fair-queue 64 256 235
        ip rsvp bandwidth 7500 7500
!
interface Serial0/0
        ip address 192.168.2.1 255.255.255.0
        fair-queue 64 256 37
        clockrate 4000000
        ip rsvp bandwidth 1158 1158
!
```

RB Configuration

```
!
access-list 101 permit udp any any range 60000 65535
access-list 102 permit udp any any range 50000 59999
access-list 103 permit udp any any range 40000 49999
access-list 104 permit udp any any range 30000 39999
access-list 105 permit udp any any range 20000 29999
!
class-map match-all platinum
        match access-group 101
class-map match-all gold
        match access-group 102
class-map match-all silver
        match access-group 103
class-map match-all bronze
        match access-group 104
class-map match-all iron
        match access-group 105
!
policy-map DiffServ
        class platinum
                priority 500
        class gold
                bandwidth remaining percent 50
```

```
        class silver
                bandwidth remaining percent 25
        class bronze
                bandwidth remaining percent 15
        class iron
                bandwidth remaining percent 10
!
interface Serial0/1
        ip address 192.168.2.2 255.255.255.0
        service-policy output DiffServ
        ip rsvp bandwidth 1158 1158
!
interface Serial0/0
        ip address 192.168.3.1 255.255.255.0
        service-policy output DiffServ
        clockrate 4000000
        ip rsvp bandwidth 1158 1158
        ip rsvp data-packet classification none
        ip rsvp resource-provider none
!
```

RC Configuration

```
!
# same access-list declaration
# same class-map definition
# same policy-map declaration
!
interface Serial0/0
        ip address 192.168.3.2 255.255.255.0
        service-policy output DiffServ
        ip rsvp bandwidth 1158 1158
        ip rsvp data-packet classification none
        ip rsvp resource-provider none
!
interface Serial0/1
```

```
        ip address 192.168.4.1 255.255.255.0
        service-policy output DiffServ
        ip rsvp bandwidth 1158 1158
        ip rsvp data-packet classification none
        ip rsvp resource-provider none
!
```

RD Configuration

```
!
# same access-list declaration
# same class-map definition
# same policy-map declaration
!
interface Serial0/0
        ip address 192.168.4.2 255.255.255.0
        service-policy output DiffServ
       clockrate 4000000
       ip rsvp bandwidth 1158 1158
       ip rsvp data-packet classification none
       ip rsvp resource-provider none
!
interface Serial0/1
       ip address 192.168.5.1 255.255.255.0
       service-policy output DiffServ
       ip rsvp bandwidth 1158 1158
!
```

RE Configuration

```
interface FastEthernet0/0
 ip address 192.168.6.1 255.255.255.0
 speed auto
 full-duplex
 fair-queue 64 256 235
 ip rsvp bandwidth 7500 7500
!
```

```
interface Serial0/0
 ip address 192.168.5.2 255.255.255.0
 fair-queue 64 256 37
 clockrate 4000000
 ip rsvp bandwidth 1158 1158
!
```

References

[1] K. L. Morse, "Data Distribution Management Migration from DoD 1.3 to IEEE 1516", *Proc. IEEE Fifth International Workshop on Distributed Simulation and Real Time Applications (D S - R T 2001)*, Cincinnati, August 2001.

[2] N. D. Georganas, "Advanced Distributed Simulation Technology and Collaborative Virtual Environment", CRC Report, Industry Canada, December 1997.

[3] J. I. Cleveland, and S. S. Herndon, "Real-Time Simulation User's Guide" http://www.cs.mcgill.ca/~nader/Red_Book.html

[4] U.S. Army STRICOM, "SIMNET Software Summary", http://www.stricom.army.mil/STRICOM/DRSTRICOM/SOFTWARE/SUMMARIES/simtem.html

[5] ALSP Joint Training Confederation, http://alsp.ie.org/alsp/biblio/mors_96_miller/mors_96.html

[6] E. Berglund and H. Eriksson, "Distributed Interactive Simulation for Group Distance Exercises on the Web", Linkoping University, Sweden, October 1998, http://www.ida.liu.se

[7] U.S. Army STRICOM, "Advanced Distributed Simulation", http://www.stricom.army.mil/STRICOM/E-DIR/ES/ADS

[8] ADS & T&E, "Gateway to the Virtual Test Environment", http://www.jads.abq.com/html/ads/ads.htm

[9] SPLINE homepage, http://www.merl.com/projects/spline/

[10] R. M. Weatherly, A. L. Wilson, B. S. Canova, E. H. Page, A. A. Zabek. "ADS through the ALSP". http://ms.ie.org/page/papers/hicss-29/camera.html

[11] OMG Special Interest Group in Simulation. http://cgi.omg.org/techprocess/xsigs.html#simsig

[12] Approved IEEE Draft, "1516-2000 IEEE Standard for Modeling and Simulation

(M&S) High Level Architecture (HLA) - Framework and Rules". http://standards.ieee.org/catalog/simint.html

[13] Approved IEEE Draft, "1516.1-2000 IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Federate Interface Specification". http://standards.ieee.org/catalog/simint.html

[14] Approved IEEE Draft, "1516.2-2000 IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Object Model Template (OMT) Specification" http://standards.ieee.org/catalog/simint.html

[15] HLA-RTI software distribute center, "HLA-RTI Programmer's Guide", http://sdc.dmso.mil/

[16] "Vendors of HLA Run Time Infrastructure", http://hla.dmso.mil/index.php?page=74

[17] S. L. Grinaway, A. Lebioda, U.S. Army TARDEC, "The Integration of Man-in-the-Loop" *Proc. of Distributed Simulation with Motion Simulation Interoperability Workshop*, Orlando, Spring, 1998.

[18] R. D. Smith, Technical Director, BTG Inc. http://www.modelbenders.com/bio/smithr.html

[19] Fourth IEEE International Workshop on Distributed Simulation and Real Time Application (D S – R T 2000), http://www.cs.unt.edu/~boukerch/DS-RT2000

[20] X. Shen, R. Hage and N. D. Georganas, "Agent-aided Collaborative Virtual Environments over HLA/RTI", *Proc. IEEE/ACM Third International Workshop on Distributed Interactive Simulation and Real Time Applications (D I S - R T ' 99)*, Greenbelt MD, October 1999.

[21] J. C. Oliveira, X. Shen and N. D. Georganas , "Collaborative Virtual Environment for Industrial Training and e-Commerce", *Proc. Workshop on Application of Virtual Reality Technologies for Future Telecommunication Systems, IEEE Globecom'2000 Conference*, San Francisco, November 2000.

[22] Simulation Interoperability Workshop http://www.sisostds.org/siw/

[23] S. Singhal, and Zyda, "Networked Virtual Environment: Design and

Implementation", Addison-Wesley Pub Co, ISBN: 0201325578, September 1999.

[24] DoD Modeling and Simulation Executive Agent for Terrain, "Modeling and Simulation Terrain Execution Plan", March 1996. http://www.tmpo.nima.mil/new_tep/tep_anx4.html

[25] L. Granowetter, MÄK Technologies, Inc. "Solving the FOM-Independence Problem", http://www.mak.com/tech/fomagile.html

[26] A. D. White. (1994). "The Impact of Cue Fidelity on Pilot Behavior and Performance", *Proc. of I/ITSEC*, Orlando, November 1994.

[27] V. Hardman, M. A. Sasse, M. Handley, and A. Watson. "Reliable audio for use over the Internet". *International Networking Conference (INET)*, Honolulu, Hawaii, June 1995.

[28] MSDN online library – Platform SDK – Graphics and Multimedia Services – Windows Multimedia – Multimedia Input – Multimedia Timers

[29] http://www.nd.edu/~lemmon/courses/UNIX/ - Signals and Timers - POSIX Interval Timers

[30] I. Kouvelas, V. Hardman and A. Watson, "Lip Synchronization for use over the Internet: Analysis and Implementation". *Proc. of IEEE Globecom '96*, London, November 1996.

[31] MSDN help of "waveOutGetPosition()". http://msdn.microsoft.com/library/psdk/multimed/mmfunc_665r.htm

[32] S. McCanne and V. Jacobson, "vic: A Flexible Framework for Packet Video", *Proc. of the Third ACM International Conference on Multimedia*, San Francisco, CA, November, 1995..

[33] IEEE Spectrum, Volume37, Number7, Jul2000. http://www.spectrum.ieee.org

[34] "Virtual Environment for Reconstructive Surgery (VERS)", http://biocomp.arc.nasa.gov/vers/

[35] LaRC's Simulation Facility Components in the system development branch of NASA's Aerospace Technology Enterprise. http://www.aero-space.nasa.gov/ http://www-sdb.larc.nasa.gov/Simulations/simulations.html

[36] The Information Directorate of the Air Force Research Laboratory, http://www.rl.af.mil/tech/thrusts/ct27_rt_people_equip.htm

[37] Center for Human Modeling and Simulation, University of Pennsylvania. http://www.cis.upenn.edu/~hms

[38] C. L. Liu, J. W. Layland, "Scheduling algorithms for multiprogramming in a hard real-time environment", JACM, Vol.20, No. 1, pages 46-61, 1973.

[39] Real-Time Consult, "RTOS Evaluation Report – Windows NT Workstation 4.0", Real-Time Consult, December 1998. http://www.realtime-info.be

[40] J. Liu, "Real-Time Systems", Section 6.8.4. Prentice Hall, ISBN 0-13-099651-3. 1999.

[41] "VxWorks5.4 Programmer's Guide", WindRiver Systems, Inc. May 1999. http://www.windriver.com/

[42] T. Modi, "Why Thread Pools are Important in Java", October 2000. http://www.devx.com/upload/free/features /javapro/2000/10oct00/tm0010/tm0010.asp

[43] G.Travis, "Using Thread Pools to Increase Threading Efficiency", June 1998. http://www.zdnet.com/devhead /stories/articles/0,4413,1600297,00.html,

[44] JTC1/SC22/WG15 – POSIX, http://anubis.dkuug.dk/JTC1/SC22/WG15/

[45] http://www.UNIX-systems.org/

[46] IEEE. The Portable Application Standards Committee, http://www.pasc.orsg/abstracts/posix.htm

[47] Real-Time PSIG, "Real-Time CORBA Specification", October 2000, http://www.omg.org/homepages/realtime/

[48] Real-Time Consult, "What makes a good RTOS", Real-Time Consult, February 2000, http://www.realtime-info.be

[49] Real-Time Consult, "The RTOS buyer's guide", Dedicated Systems Maganize. http://www.dedicated-systems.com/encyc/buyersguide/rtos /Dir228.html

[50] Wind River, http://www.windriver.com/

[51] QNX Software Systems, http://www.qnx.com/

[52] D. Hildebrand, "An Architectural Overview of QNX", http://www.qnx.com/literature/whitepapers/archoverview.html

[53] SUN software white paper, "Scalable Real-Time Computing in the Solaris™ Operating Environment", http://www.sun.com/software/white-papers /wp-realtime/

[54] DIAPM, "RTAI Programming Guide 1.0", http://server.aero.polimi.it/projects/rtai/

[55] SGI, "SGI-IRIX6.5 REACT Real-Time Programmer's Guide", http://techpubs.sgi.com/

[56] PC Engines, "About UNIX and real-time scheduling", http://www.pcengines.com/schedule.htm

[57] QNX Software Systems, "QNX4 Real-Time OS Introduction and System Architecture", http://www.qnx.com/products/os/qnxrtos.html

[58] B. Johnson, SGI, "Windows NT® or IRIX® in Real-Time Simulation: Does It Make a Difference?" February 1999. http://www.sgi.com/VisSim/Technical/nt_irix.html

[59] Dedicated Systems Magazine, "RTOS Evaluation Reports", http://www.dedicated-systems.com/encyc/buyersguide /rtos/evaluations/docs.asp

[60] Encore ports its real-time environment to Solaris, March 2000. http://www.sun.com/smi/Press/sunflash/2000-03/sunflash.20000310.1.html

[61] Real-Time Magazine 1999, "Windows NT Real-Time Extension". http://www.dedicated-systems.com/magazine/ 97q2/index972.htm

[62] G. Huston, "ISP Survival Guide: Strategies for Running a Competitive ISP". Wiley & Sons, Inc. 1999. ISBN 0-471-31499-4

[63] M. E. Flannagan, "Administering Cisco QoS in IP Networks", ISBN 1-928994-21-0, 2001.

[64] IEEE Standard. LAN/MAN Standards Committee of the IEEE Computer Society, "IEEE Standard 802.1D, 1998 Edition. Part 3: Media Access Control (MAC)

Bridges", http://standards.ieee.org/catalog/olis/lanman.html

[65] White paper, Nortel Networks, "QoS in Frame-Switched Networks", http://www.nortelnetworks.com/solutions/lan/collateral/ppqoswp.pdf

[66] Cisco, "Multicast in a Campus Network: CGMP and IGMP Snooping", http://www.cisco.com/warp/public/473/22.html

[67] LAN MAN Standards Committee of the IEEE Computer Society, "IEEE Standard 802.3x-1997: Specification for 802.3 Full Duplex Operation and Physical Layer Specification for 100 Mb/s Operation on Two Pairs of Category 3 or Better Balanced Twisted Pair Cable (100BASE-T2)", Approved 20 March 1997, http://standards.ieee.org/getieee802

[68] LAN/MAN Standards Committee of the IEEE Computer Society, "IEEE 802.3, 2000 Edition (ISO/IEC 8802-3: 2000) (E), Part 3: CSMA/CD Access Method and Physical Layer Specifications", http://standards.ieee.org/catalog/olis/lanman.html

[69] IETF Integrated Services (IntServ) Working Group, http://www.ietf.org/html.charters/intserv-charter

[70] IETF Differentiation Service Working Group, http://www.ietf.org/html.charters/diffserv-charter

[71] IETF Multiprotocol Label Switching (MPLS) Working Group, http://www.ietf.org/html.charters/mpls-charter

[72] IETF RFC2212, S. Shenker, C. Partridge, R. Guerin, "Sepcification of Guaranteed Quality of Service", September 1997. http://www.ietf.org/rfc/rfc2212.txt

[73] IETF RFC2211, J. Wroclawski, "Sepcification of the Controlled-load Network Element Service", September 1997. http://www.ietf.org/rfc/rfc2211.txt

[74] IETF RFC2205, R. Braden, L. Zhang, S. Berson, S. Herzog, S. Jamin, "RSVP – Version 1 Functional Specification", September 1997. http://www.ietf.org/rfc/rfc2205.txt

[75] Microsoft, "QoS protocols & architectures", http://www.winsock2.com/qos/whitepapers/protocols.htm

[76] White paper. Cisco, "Quality of Service (QoS) Networking", June1999. http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/qos.htm

[77] G. Gaines, and M. Festa, "A Survey of RSVP/QoS Implementations", July 1998. http://ai.iit.nrc.ca/IETF/RSVP_survey/

[78] The Open Group, "Technical Standard: RAPI", http://www.opengroup.org/onlinepubs/9619099

[79] "Intel® PC-RSVP", http://developer.intel.com/ial/rsvp and "Microsoft Winsock Generic QoS Mapping", September 1998.

[80] IETF RFC 2814. R. Yavatkar, D. Hoffman, Y. Bernet, F. Baker, M. Speer, "SBM (Subnet Bandwidth Manager): A Protocol for RSVP-based Admission Control over IEEE 802-style networks", May 2000.

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

[82] IETF RFC3175, F. Baker, C. Iturralde, F. Le Faucheur, B. Davie, "Aggregation of RSVP for IPv4 and IPv6 Reservations", September 2001.

[83] Cisco IOS feature, "RSVP Scalability Enhancement", http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122newft/122t/122t2/rsvps cal.htm

[84] IETF Internet Draft, F. Faucheur, L. Wu, B. Davie, "MPLS Support of Differentiated Services", April 2001. http://www.ietf.org/internet-drafts/draft-ietf-mpls-diff-ext-09.txt

[85] Cisco IOS feature, "Diff-Serv-aware MPLS Traffic Engineering (DS-TE)", http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122newft/122t/122t4/ft_ds_te.htm

[86] Abilene Network, http://www.internet2.edu/abilene/

[87] Very High Performance Backbone Network Service, http://www.vbns.net/

[88] The Quantum Project, http://www.dante.net/quantum/

[89] IETF RFC1667, S. Symington, D. Wood, M. Pullen, "Modeling and Simulation

Requirements for Ipng", August 1994.

[90] IETF RFC2502, M. Pullen, M. Myjak, C. Bouwens, "Limitations of Internet Protocol Suite for Distributed Simulation in the Large Multicast Environment", February 1999.

[91] C. Chassot, A. Lozes, F. Garcia, M. Diaz, L. Dairaine, L. Rojas, "QoS required by a distributed interactive simulation in a new generation Internet". *Proc. IDMS (Interactive Distributed Multimedia Systems)*, Toulouse, France, October, 1999.

[92] L. Ferral, G. Shackelford, "Network Resource Reservation in a Distributed Simulation Environment". *Proc. Simulation Interoperability Workshop*, Orlando, Florida, USA, Spring 1998.

[93] J. Calvin, C. Chiang, S. McGarry, S. Rak, D. Hook, "Design, Implementation, and Performance of the STOW RTI Prototype (RTI-s)", *Proc. Simulation Interoperability Workshop*, Orlando, Florida, USA, March 1997.

[94] M. Myjak, S. Sharp, "Java Real-Time RTI", *Proc. Simulation Interoperability Workshop*, Orlando, Florida, USA, Fall, 1998.

[95] C. Greenhalgh, S. Benford, G. Reynard, "A QoS Architecture for Collaborative Virtual Environments", *Proc. ACM Multimedia Press 1999*, Orlando, Florida, USA, 1999. http://www.crg.cs.nott.ac.uk/research/publications/

[96] L.Ferral, G.Shackelford, Network Resource Reservation in a Distributed Simulation Environment. *Proc. Simulation Interoperability Workshop*, 1998.

[97] IETF RFC2210, J. Wroclawski, "The Use of RSVP with IETF Integrated Services", September 1997. http://www.ietf.org/rfc/rfc2210.txt

[98] J.Pullen, "Reliable Multicast Network Transport for Distributed Virtual Simulation", *Proc. IEEE/ACM Third International Workshop on Distributed Interactive Simulation and Real Time Applications (DIS-RT'99)*, Greenbelt MD, October, 1999.

[99] M.Nilsson, D.Dalby, J.Donnell, "Layered Audiovisual Coding for Multicast Distribution on IP networks", *Proc. 6th IEEE International Conference on Multimedia Computing and Systems*, Centro Affari, Florence, ITALY, June 1999.

http://www.labs.bt.com/projects/mware.htm

[100] X. Wang, H. Schulzrinne, "Comparison of adaptive Internet multimedia applications", *IEICE Transactions on Communications*, pp.806-818, June 1999. http://www.cs.columbia.edu/~xinwang/public/paper/

[101] OMG Standard. Real-Time PSIG, "Real-Time CORBA Specification", October 2000. http://www.omg.org/homepages/realtime/

[102] J. Calvin, C. Chiang, S. McGarry, S. Rak, D. Hook, "Design, Design, Implementation, and Performance of the STOW RTI Prototype (RTI-s)", *Proc. Simulation Interoperability Workshop*, Orlando, Florida, USA, 1997.

[103] J. Calvin, R. Weatherly, "An Introduction to the HLA RTI", *Proc. Simulation Interoperability Workshop*, Orlando, Florida, USA, 1996.

[104] S. McGarry, P. DiCaprio, A. Wilson, R. Weatherly, "Design Issues for the HLA RTI Prototype V0.2", *Proc. Simulation Interoperability Workshop*, Orlando, Florida, USA, 1996.

[105] R. Briggs, R. Weatherly, R. Richardson, J. Olszewski, T. Stark, "RTI F.0 Integrated Product Team", *Proc. Simulation Interoperability Workshop*, Orlando, Florida, USA, Spring 1997.

[106] MÄK Technologies, http://www.mak.com/rti.htm

[107] J. Pullen, N. Kakarlamudi, "Performance Issues for the Lightweight RTI", *Proc. Simulation Interoperability Workshop*, Orlando, Florida, USA, Fall 1998.

[108] G. Blank, "Design and Implementation of the High Performance Computing RTI for the HLA in SPEEDES0.81", *Proc. Simulation Interoperability Workshop*, Orlando, Florida, USA, Spring 2000.

[109] SAIC RTI help desk, http://helpdesk.dctd.saic.com/

[110] B. Givens, "Positions For and Against and Open-Source RTI", *Proc. Simulation Interoperability Workshop*, Orlando, Florida, USA, Spring 2000.

[111] S. Bachinsky, J. Noseworthy, F. Hodum, "Implementation of the Next Generation RTI", *Proc. Simulation Interoperability Workshop*, Orlando, Florida, USA, Spring, 1999.

[112] S. Bachinsky, L. Mellon, "RTI2.0 Architecture", *Proc. Simulation Interoperability Workshop*, Orlando, Florida, USA, Spring 1998.

[113] S. Pounds, J. Simmers, "Scalability and RTI-NG", *Proc. Simulation Interoperability Workshop*, Orlando, Florida, USA, Fall 2000.

[114] B. Murray, S. Monson, "Analysis of a Real-Time HLA Distributed Mission Training Federation", *Proc. Simulation Interoperability Workshop*, Orlando, Florida, USA, Spring 2000.

[115] T. Modi, "Why Thread Pools are Important in Java", October 2000. http://www.devx.com/upload/free/features/javapro/2000/10oct00/tm0010/tm0010.asp

[116] A. Schultz, "Multi-threading in a CORBA ORB", January 2001. http://www.cs.uni-magdeburg.de/~aschultz/mico/mt/dist/ thesis.pdf

[117] "The Adaptive Communication Environment", http://www.cs.wustl.edu/~schmidt/ ACE.html

[118] C. Ryan, D. Levine, D. Schmidt, J. Noseworthy, "Applying a Scalable CORBA Event Service to Large-scale DIS". *Proc. of the Fifth International Workshop on Object-Oriented Real-Time Dependable Systems*, Monterey, California, USA, November 1999.

[119] http://msdn.microsoft.com > MSDN Code Center > Code Examples > Windows Development > QoS Samples

[120] Cisco IOS 12.2(T), "RSVP Scalability Enhancement", http://www.cisco.com

[121] Cisco, Catalyst 3500 Series XL Switches, http://www.cisco.com/warp/public/cc/pd/si/casi/ca3500xl/

[122] Lucent, "Aggregating DSL Services", http://www.lucent.com/products

[123] Motorola, The Broadband Services Router 64000, http://www.gi.com/noflash/ipns_solutions.html

[124] M. James, A. Bubke, and M. Sedgwick, "Gigabit Ethernet – Accelerating the standard for speed", http://www.biz.uiowa.edu/class/6k180_park/Student-Reports/abubke

[125] White paper, Microsoft, "The Microsoft QoS Components", http://www.microsoft.com/windows2000 /docs/QoSComp.doc

[126] D. Pappalardo, Network World, "ISPs continue to improve Internet access SLAs", February 2001. http://www.itworld.com/Net/2608/NWW117115_02-19-2001/

[127] M. Yoo, C. Qiao, S. Dixit, "QoS Performance in IP over WDM Networks", http://www.cs.buffalo.edu/~qiao/jsac-qos.pdf

[128] UUNet SLA, http://www.uu.net/customer/sla/

[129] D. Pappalardo, "SLA competition heightens", Network World Internet Services Newsletter, July 2001.

[130] J. Wexler, Campbell, CA. "The QoS Conundrum", April 2001. http://www.bcr.com/bcrmag/2001/04/p48.asp

[131] IETF Internet Traffic Engineering Working Group (tewg), http://www.ietf.org/html.charters/tewg-charter.html

[132] V. Alwayn, "Advanced MPLS Design and Implementation", ISBN 1-58705-020-x, 2002, Cisco press.