uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Diabi Abdelfettah

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.C.S. (Master of Computer Science)

GRADE / DEGREE

School of Information Technology and Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

A Hybrid Application-layer Multicasting Protocol for Distributed Simulations on the Internet

TITRE DE LA THÈSE / TITLE OF THESIS

Shervin Shirmohammadi

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

Jiying Zhao

Peter X. Liu

Gary W. Slater

Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

A Hybrid Application-Layer Multicasting Protocol for Distributed Simulations on the Internet

By

Abdelfettah Diabi

A thesis submitted to the

Faculty of Graduate and Postdoctoral Studies

In partial fulfillment of the requirements for the degree

Master in Computer Science

Ottawa-Carleton Institute for Computer Science

School of Information Technology and Engineering

University of Ottawa

Your file Votre référence
ISBN: 978-0-494-18386-1
Our file Notre référence
ISBN: 978-0-494-18386-1

Abstract

IP multicast is typically used to support message transmission among entities in collaborative virtual simulations running on Intranets, especially military training simulations. But, IP Multicast is mostly not available on the Internet because it is not deployable by ISP's due to undefined billing at the source and unlimited amount of overlay topologies for data transmission. Alternatively, researches have in recent years proposed the use of Application Layer Multicasting techniques (ALM) to alleviate this problem and to allow a somewhat scalable message passing among peers in a group of users on the Internet. As the number of users grows in the session, efficient handling of resources and scalable communication between users becomes critical due to network traffic and computation required to track all objects in the session. In this thesis, peer-to-peer communication architecture for distributed simulation is proposed. The architecture uses both proxies and end-systems to provide a number of communication services: 1) Best effort LAN multicast 2) Timely-reliable LAN multicast 3) Best effort peer to peer delivery on the Internet 4) Timely-reliable peer to peer delivery on the Internet, and 5) Any combination of the above, including LAN to Internet translation and vice versa. The main achievement of this research could be summarized as the design, justification, and implementation of an architecture that, in comparison to others, has higher efficiency and guaranteed reliability for performing tightly synchronous collaborative tasks in Collaborative Virtual Environments (CVEs).

Acknowledgements

At the end of this fruitful journey, I would like to praise and thank my Lord for bestowing huge bounties on me and for supplying me with tremendous power and energy to successfully complete my research with excellent results.

I would also like to thank my Supervisor, Dr. Shervin Shirmohammdi for his continuous help, support and guidance. My gratitude also goes to Mr. Pascal Lacombe for participating in the development of the graphics applications to test my work.

Finally, I dedicate this thesis to my Parents who have provided everything possible to make my life smooth and to help me achieve my objectives.

Table of Content

List of Figures

List of Tables

List of Abbreviations

ALM	Application-Layer Multicasting
AoIM	Area of Interest Management
CVE	Collaborative Virtual Environments
DIS	Distributed Interactive Simulation
DIVE	Distributed Interactive Virtual Environment
DoD	Department of Defense
HDSP	Hybrid Distributed Simulation Protocol
HLA	High Level Architecture
IGMP	Internet Group Management Protocol
IP	Internet Protocol
ITU	International Telecommunication Union
LAN	Local Area Network
MSN	Multicast Service Nodes
NAT	Network Address Translator
OMNI	Overlay Multicast Network Infrastructure
P2P	Peer to Peer
PIM	Protocol Independent Multicast
Probass	Proxy-based Single Source ALM Protocol
QoS	Quality of Service
RTT	Round-Trip Time
SCTP	Synchronous Collaborative Transport Protocol
UDP	User Datagram Protocol
UML	Unified Modeling Language

Chapter 1

Introduction

1.1 Motivation

Distributed simulation is the process of executing simulations on geographically distributed computers interconnected via local area network or Internet to accomplish common tasks [9]. The fundamental advantage of distributed simulation is the reduction of model execution time by a factor up to the number of entities (end-hosts, servers, etc.) participating in the session [9] (Figure 1).

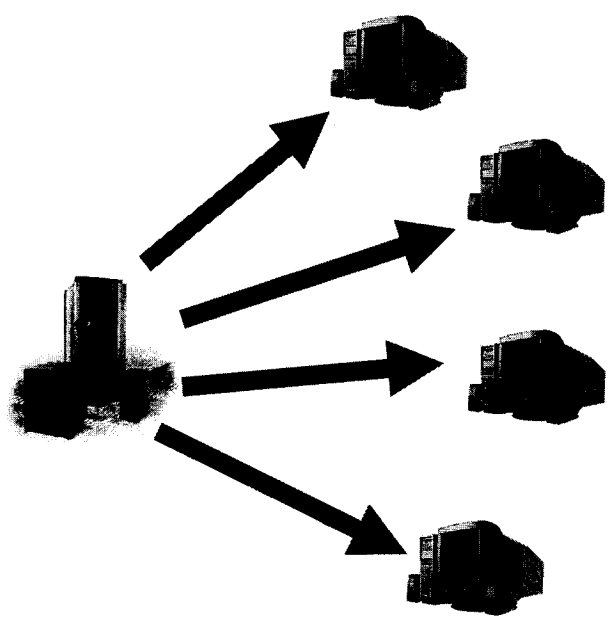

Figure 1: End-hosts executing parts of a large program to achieve a common task

Distributed simulation covers a wide range of applications especially those applications that require intensive resource usage and demand many geographically distributed users

to participate in the simulation. Examples of distributed simulation include military simulation, grid computing, collaboration in virtual environments, multiplayer online games, etc. Figure 2 portraits a scenario of gaming application where users have a common vision of the simulation environment while the computation of the whole game state is distributed among those users.

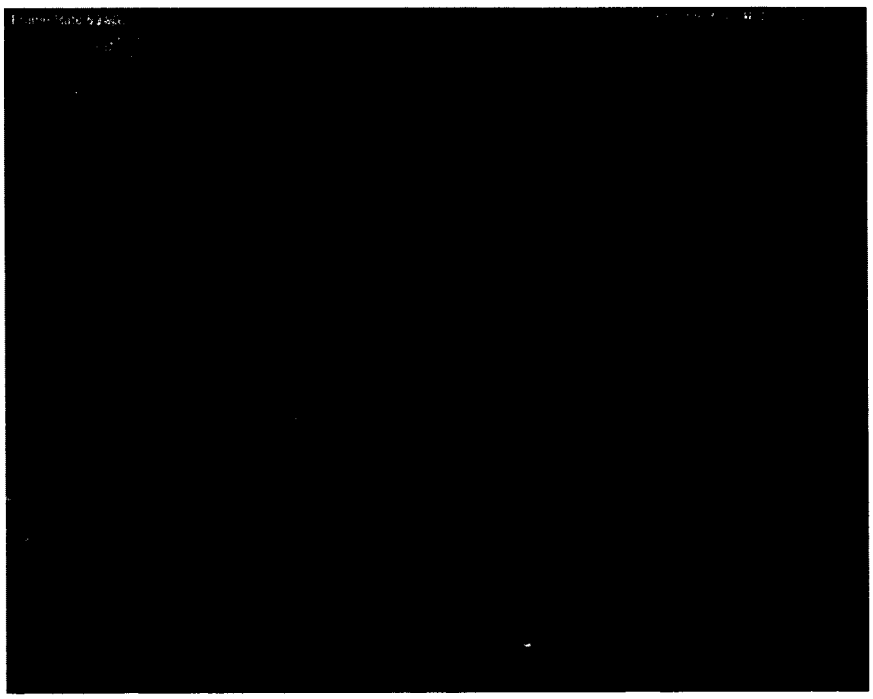

Figure 2: Distributed simulation example: online gaming

Distributed simulations in collaborative virtual environments require the transport of update messages between entities participating in the simulation. The primary reason for exchanging messages between entities is to ensure that all participants have the same synchronized view of shared objects and of one another. Although the nature of collaboration is determined by the context and the situation where objects interact, it can be classified into two types of collaboration: combat-type simulations and closely coupled collaboration. The former uses dead-reckoning techniques and assumes that the loss of one update message is compensated by the next [4], while the latter is unpredictable and requires reliable delivery of certain "key" updates [5][8]. There can also be scenarios where communications are performed only between two peers.

Since entities participating in the simulation send and receive update messages to each other, great attention should be given to transport mechanisms of those update messages as the number of entities grow in the session. The naïve approach to handle the transport of messages is the client-server architecture, which is bandwidth intensive and the server in this approach is the bottleneck. A better approach to handle massive multi-user simulation is IP-multicasting, however, IP-multicasting is not deployable on the Internet by ISP's due to undefined billing at the source and because it requires manual configuration at routers [32].

To alleviate the problem caused by IP-multicasting, researches have in recent years proposed the use of Application-Layer Multicasting techniques (ALM) to allow a somewhat scalable message passing among participants in the simulation environment. The concept of ALM is simply the implementation of multicasting functionality as an application service instead of a network service. Figure 3 below represents a group of entities organized in a typical ALM configuration. Entities form a multicasting tree (logical topology) among themselves. Here, the multicasting tree has been built at the application layer as seen in the figure. Using only the unicasting capability of the network, the source (S) sends two packets, one to D1 and one to D2, each of which in turn send the packet to D4 and D3, respectively.

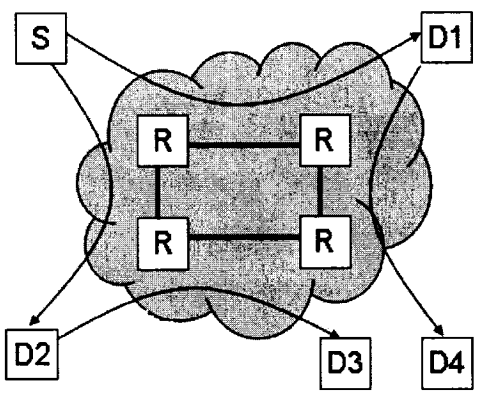

Figure 3: Application Layer Multicasting

While existing ALM protocols have addressed many transport issues related to distributed simulations in general [1], [2], [3], [4], [6], they fail to fulfill collaboration needs mainly because they do not consider the properties of collaboration data. Synchronous collaboration in virtual reality spaces has specific requirements that differ from those of other application data, leading to a different communication approach [5], [7], and [8].

In this thesis it is believed that a framework on top of which simulation applications are written will provide transport services suitable for all kinds of simulation data including collaboration specific data.

1.2 Research Problem

One of the main problems that have been studied and addressed to some extent in recent years is network lag in multi-user collaborative environments. Network lag adversely affect networked Collaborative Virtual Environments (CVEs). When a user participates in such environment, his/her interactions with other users and/or the environment must be sent to other participants over the network such that all entities involved are updated with that user's latest state. Because of network limitations and traffic conditions, some of these "updates" are lost or delayed. In fact, network lag is present in any distributed application such as web browsing, email, and audio/video streaming. However, due to its requirements for highly interactive operations, networked games are especially susceptible to packet loss and delay.

Besides network lag, collaboration in distributed simulation introduces a new problem that deals with the nature of data being exchanged. The general assumption in distributed simulations is that objects transmit update messages often, and that the latest state of things can be determined by techniques such as dead-reckoning algorithms because they are somewhat predictable. Experience has shown that these assumptions work very well for scenarios such as simulation of battlefields or multi-user avatar-based games. In fact most of these systems have been specifically designed for either military purposes or

games and they do a good job at that. In "shoot-em-up" games or battlefield scenarios; people, tanks, planes, and other war machines are almost constantly moving in a short-term predictable manner. A plane's course of flight can be extrapolated from its position and velocity vector. Also, a lost update message is usually followed by many other update messages, or keep-alive messages. However, these assumptions fail for Collaborative Virtual Environments (CVEs). The conditions in CVEs can be quite the opposite: shared objects often do not send continuous update messages, and when they do it is not necessarily in a predictable manner. When coordinating closely coupled collaborative tasks in CVEs, there is no room for "guessing" the state of a shared object. All participants must reliably receive the most current state or collaboration might fail. Dead reckoning and similar algorithms do not guarantee that all hosts share identical state about a shared entity and they require hosts to tolerate discrepancies. Moreover, those algorithms lose efficiency in situations when precision and consistency are vital [31], such as collaborative situations.

1.3 Research Objective

The research conducted in this thesis is motivated by a real world problem: How can entities in collaborative environment simulations interact and exchange update messages with the least bandwidth usage, in a timely manner while taking into account the nature of collaboration data.

Researchers in recent years proposed Application-Layer multicasting to overcome the deployment difficulties that face IP-multicasting in the Internet. Although many ALM protocols have been developed to allow more scalability and efficient message passing among entities in the simulation environments they do not put into account collaboration data requirements.

The research objective in this thesis is to design, deploy, and test an ALM based protocol that can support multi-user collaborative environments on the Intranet and on the Internet simultaneously. This protocol must support all kind of simulation data and must meet

collaboration-specific requirements of tightly coupled tasks, which we shall see in the coming chapters.

1.4 Research Contributions and New Ideas

This thesis introduces new ideas and a number of contributions including:

- Proposal of a hybrid peer-to-peer communication architecture for distributed simulations, which is more efficient than existing ones in the context of supporting collaborative update messages.
- Design and implementation of different transport services using the ALM architecture, including:
  - Best-effort LAN
  - Timely-reliable LAN
  - Best-effort peer-to-peer delivery on the Internet
  - Timely-reliable peer-to-peer delivery on the Internet
  - Combination of the above services
- The designed protocol is able to provide guaranteed reliability for "key" updates.
- Prototyping and performance evaluation of the proposed approach.
- Peer-reviewed publications:
  - A.Diabi, S. Shirmohammadi, A. Gillmore, P. Lacombe, J.C. Oliveira, "Internet-based Collaborative Virtual Simulations with Area of Interest Management", Proc. International Symposium on Collaborative Technologies and Systems (CTS '06), Las Vegas, U.S.A, May 2006.
  - S. Shirmohammadi, A. Diabi, P. Lacombe "An Application-Layer Multicasting Protocol for Distributed Collaborations", Proc. IEEE Workshop on Haptic Audio Visual Environments and their Applications, Ottawa, ON, Canada, October 2005.
  - S. Shirmohammadi, A. Diabi, P. Lacombe, "A Peer-to-Peer Communication Architecture for Networked Games", Proc. Future Play 2005, Michigan State U., USA, October 2005

1.5 Organization of the Thesis

The remaining part of this thesis is organized as follows:

Chapter 2 provides background knowledge of distributed simulation, Synchronous Collaborative Transport Protocol (SCTP), an overview of ALM protocols with their advantages and limitations, and a brief description of multi-user networked games. **Chapter 3** presents the underlying algorithm and the rational behind ALM-tree formation for the proposed protocol, Hybrid Distributed Simulation Protocol (HDSP). In **Chapter 4**, a detailed discussion of the design and communication framework for HDSP is presented. More specifically, the main classes and components of the system are illustrated using Unified Modeling Language (UML). **Chapter 5** presents the simulation prototype, performance and subjective evaluation setup and test results. Finally, **Chapter 6** summarizes and concludes the thesis with future work recommendations.

Chapter 2

Background

The main objective of this thesis is the design, simulation and implementation of an ALM architecture that supports the transmission of all types of simulation data. Prior to engaging in the detailed explanation of the proposed architecture, we shall present a brief overview of the related work.

2.1 Distributed Simulation

Distributed simulation is the process of executing simulations on geographically distributed computers interconnected via local area network or Internet [9], [16]. In CVE technologies, distributed simulation plays an important role. This is true because of the nature of data exchanged between entities. Moreover, entities in CVEs are scattered in different locations, which imply that large programs' execution can be done efficiently if it is divided and shared among participating entities. This is achieved through exchanging update messages between entities/users in CVEs (Figure 4).

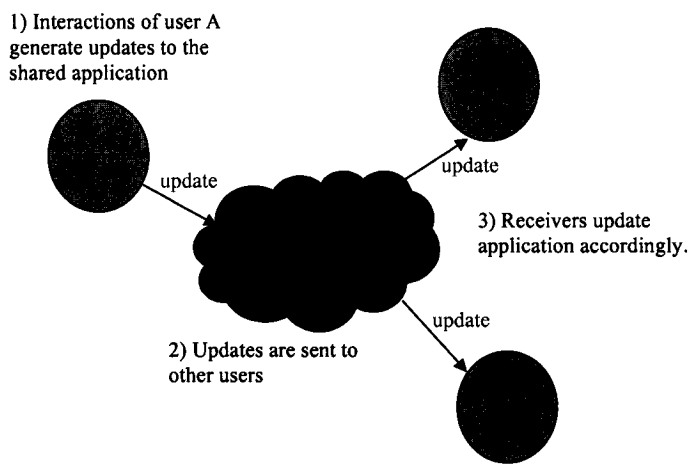

Figure 4: Exchange of update messages in collaboration systems

In the following few sections we will briefly discuss the advantages and types of distributed simulation.

2.1.1 Advantages of Distributed Simulations [9]

The following are some of the advantages introduced by distributed simulation:

- **Reducing model execution time:**

  This is achieved when the entities in the session run the divided portions of the program. This process can significantly reduce the execution time by a factor equal to the number of entities participating in that simulation. This advantage can be quite beneficial if the number of entities is huge, i.e. (100's or 1000's of nodes) which is the case in massive multi-player online games. As the number of users grows in the session, execution time of large-scale programs could be handled efficiently through using entities resources.

- **Geographical distribution of users and/or resources**

  As the name (Geographical distribution of users and/or resources) suggests, users and/or resources are geographically scattered. The travel expenses can be alleviated through making virtual worlds with multiple entities communicating from different sites.

- **Fault tolerance**

  One fundamental advantage of distributed simulation is that in case of node failure, other nodes can continue to operate and do the simulation, provided that the failure of that node does not cause critical damage to the simulation and the functionalities done by that node could be replaced by other nodes.

2.1.2 Distributed Virtual Environment

In this section a brief overview of some distributed virtual environments is presented. The reader is referred to [10], [16] for more details.

2.1.2.1 Distributed Interactive Simulation (DIS) [10]

In order to conduct real-time war games across multiple host computers, DIS [11] is considered as a standard. DIS was proposed as a consequence of the U.S. government's initiatives for defining architecture to make the virtual world of battlefield by linking scattered entities of various types. DIS uses are targeted to support a mixture of virtual entities with computer-controlled behavior, virtual entities with live operators, live entities, and constructive entities. DIS draws heavily from the experience of the Advanced Research Projects Agency (ARPA) who developed the Simulator Networking (SIMNET) program by adopting many of SIMNET's basic concepts and heeding lessons learned. In a nutshell, DIS provides for the capability to set up battlefield scenarios, with both friendly and opposition forces on a specified terrain.

The mechanism of propagating the update messages is one of the interests of this research and in DIS terminology; it is called Protocol Data Units (PDU). It uses multicast packets over User Datagram Protocol (UDP) to propagate state information about an entity among hosts. Clearly the network traffic is greatly reduced by using multicast as the sender transmits only one packet to the whole multicast group. However, further studies have shown that network traffic is still high with many simultaneous users in one simulation.

DIS uses *Dead Reckoning* to overcome this deficiency. The idea behind dead reckoning is that instead of just sending an entity's location, a host sends a message that contains the entity's location, a time stamp, and a velocity vector. Each host in the network can predict the entity's future locations based on the velocity without additional updates. The prediction becomes erroneous when the velocity vector changes. DIS figures out this issue by having each entity run the dead reckoning algorithm for its own object. As soon as the difference between an actual location and a predicted location exceeds a certain threshold, the entity re-transmits an update message with the current position/velocity. When objects are short-term predictable, this technique reduces the network traffic considerably.

Moreover, DIS entities send a periodic update message. These update messages are called *keep-alive* messages. If these messages are not received from an entity within a predetermined duration, it is assumed that the entity's existence in the simulation has been terminated. If an entity enters the simulation after it has started, it will receive every entity's state within a few seconds through the keep-alive messages.

Despite all these techniques, experiences have shown that large simulations still generate a bandwidth problem. In addition, the processing of update messages for a large number of entities overwhelms the hosts. This is due to the fact that even if some hosts are not interested in some entities (because they are too far apart); all entities in the simulation transmit update messages to all hosts. Hence, DIS introduces another technique, called *filtering*. Filtering causes unwanted update messages to be dropped at the host without handling, while all the update messages are still transmitted throughout the network.

2.1.2.2 Distributed Interactive Virtual Environment (DIVE) [10]

The Distributed Interactive Virtual Environment (DIVE) is an internet-based multi-user virtual reality system, which was developed at the Swedish Institute of Computer Science [12]. The participants of a DIVE session usually navigate in a 3D space and collaborate with other users and applications. DIVE is an experimental stage for the development of virtual environments where user interfaces and applications are based on shared 3D synthetic environments. Several networked participants interact over a network that is especially set to multi-user applications.

DIVE is based on a peer-to-peer method. It works in a decentralized way where peers communicate by reliable and non-reliable multicast. It is possible to form different multicast groups and exchange data among members in a group.

2.1.2.3 High Level Architecture (HLA) [13]

HLA is architecture for interoperability and general reuse. The HLA was developed under the leadership of the Defense Modeling and Simulation Office (DMSO) to support reuse and interoperability across large number of different types of simulations developed

and maintained by the US Department of Defense (DoD). HLA provides a generic environment that can be attached to any virtual object in order to participate in the simulation. In the Rules' document of HLA, one can find the general principles that define HLA and delineate ten basic rules which apply to HLA federations and federates. The HLA Interface Specification defines the functional interface between federates and the Runtime Infrastructure (RTI).

The Runtime Infrastructure contains the communications entities that propagate update messages among entities in the virtual environment. Although HLA does not specify the exact underlying mechanism for communications, it does specify certain requirements that it must meet. HLA specifies that RTI must support both best effort and reliable communication service. UDP multicast and TCP channels are implementations for RTI.

2.2 Synchronous Collaboration in Virtual Environment [14]

In CVEs, many types of data are exchanged depending on the need and service requirements. Each data type has a preferred transport mechanism. In this section we focus on a transport mechanism for "collaborative data". We will be covering two important concepts that have been thoroughly investigated [14]. These are:

1. "Interaction Stream Model" for translating the user' collaborative actions into an application- level model.
2. Transport protocol for efficient transmission of this stream among a large number of participants, taking in consideration collaborative data requirements.

2.2.1 Interaction Stream

A series of update messages generate a sequence of interactions of a user with a shared object grouped into one *stream*. The generic *interaction stream is depicted* in Figure 5.

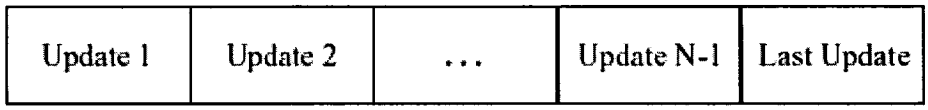

Figure 5: Generic interaction stream

Update messages are categorized according to how critical they are. For some of those updates reliability is crucial while other updates just need best effort. Hence, update messages are categorized as *key* update messages, and regular update messages. After this categorization, the interaction stream will look different than the generic one, as shown in Figure 6. We will see next what makes an update to be a "key".

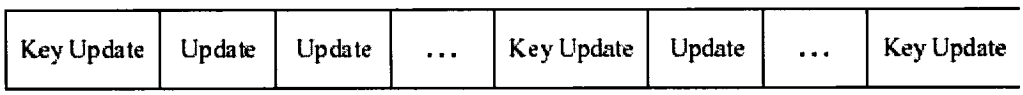

Figure 6: *Interaction Stream.*

Notice that the last update message will be a key message because it is the most important message, and if it is lost a potential collaboration failure occurs since different participants will perceive the last position of the object differently.

One way to save the bandwidth is through the use of "differential messages". The basic idea here is that instead of sending the entire new state of the object being interacted with, it is enough to send the difference between the current state and the previous state(s). Consider the example of an object moving from location (12245, 156, -1233) to (12243, 155, -1230) and then to (12241, 153, -1229). Figure 7 shows two types of interaction streams; one is the regular one and another one uses differential messages.

Regular update messages

(12245, 156, -1233)	(12243, 155, -1230)	(12241, 153, -1229)

Differential update messages

(12245, 156, -1233)	(-2, -1, 3)	(-4, -3, 4)

Figure 7: Regular and differential update messages

Applying the proposed architecture of *key* and *non-key* updates to differential messages, Reference updates are sent as key messages while differential updates are sent by best-effort transport. Figure 8 shows how differential messages can be sent using the proposed architecture.

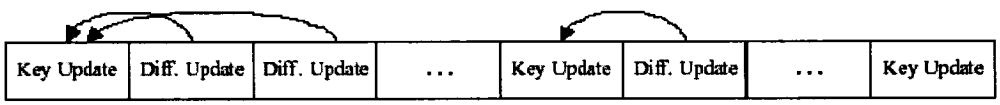

Figure 8: Differential update messages

2.2.2 SCTP: the Synchronous Collaborative Transport Protocol

SCTP is designed based on the interaction stream described above and it takes collaboration data properties into account. SCTP is a host-to-host layer protocol and it is encapsulated into UDP packets. It is based on IP multicast. The main advantage of SCTP is the support for collaborative data and more specifically handling *key* update messages.

2.2.2.1 Reliability for Key Updates

ACK-based (sender-initiated) and NACK-based (receiver-initiated) are the two main approaches for achieving reliability. SCTP uses ACK-based approach instead of NACK-based approach. Although NACK-based approach uses Automatic Repeat Request (ARQ), Forward Error Correction (FEC), or combinations of both, it is not utilized by SCTP, simply because it is not efficient for the Interaction Stream. ARQ-

based scheme only detects a loss when it receives a newer update message with a larger time stamp/sequence number. As we can see, this is not CVE compliant due to the fact that a newer message might be transmitted late or might not be transmitted at all leading to a collaboration failure. ACK-based approach is utilized instead with slight modification [15] to avoid the ACK implosion problem.

2.3 Overlay Networks

Overlay network constructs a virtual distributed network on top of a physical network. There are many overlay networks currently available and deployed for various purposes. This concept is not new. Peer-to-peer is an example of overlay network. In peer-to-peer, peers are loosely coupled with one other, play an equal role in the system, independent of each other, and can join or quit the system at any time, but their main objective is to share the content. So it is quite clear that overlay networks are designed and implemented targeting to provide some services by exploiting underlying network infrastructure.

The layout of overlay network is quite different from the underlying network. Usually the nodes that use the service of overlay network are only the member of this virtual network. Figure 9 depicts a snapshot of overlay network. This topology may be a star or may be ring or may be anything. It may happen that any two nodes that are the neighbor of each other in the overlay network but physically they are long away.

A number of overlay networks have been developed. These systems or architectures are designed to provide functionalities or services that cannot be easily or quickly provided by the IP network. But what are the reasons to design overlay networks? There many reasons behind the development of overlay network. Some of them are mentioned below,

- To provide robust end-to-end connectivity
- To protect against denial of service attacks
- Multicast connectivity to end users
- Self organization and resilient data distribution

- Security issues, etc.

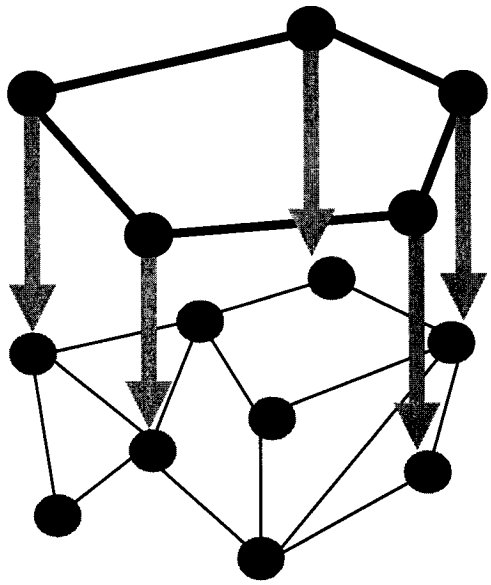

Figure 9: An overlay network. Overlay network on top of physical network

2.3.1 Applications of Overlay Networks

Overlay networks consist of routing software installed at the selected sites to communicate with each other. They exchange their own control packets for the management of overlay network. The concept of overlay system is already deployed in the IP network, namely multicast backbone (MBone), IPv6 backbone (6Bone), virtual network system (VNS), peer-to-peer network, content addressable network (CDN), and application layer multicast (ALM), etc. In the following sub sections a brief explanation of some overlays networks with their working principles is presented.

Multicast Backbone or IPv6 Backbone

In order to support multicasting across the network multicast routing protocols have to be deployed all the routers on Internet. But this is quite infeasible. One possible alternative solution is to use tunneling concept. Tunneling is a technique which can be used for transporting new — not universally supported — protocols or services, such as IPv6 or multicasting, over the Internet. The unsupported packets are "wrapped" in standard IP packets and transported across the Internet between routers that do support the protocol.

Figure 10 is an example that illustrates the multicast backbone concept. Shared communications such as videoconferencing, collaborative workspaces are currently using MBone. It is often connected to universities and research institutions but not generally connected to Internet Service Providers [35].

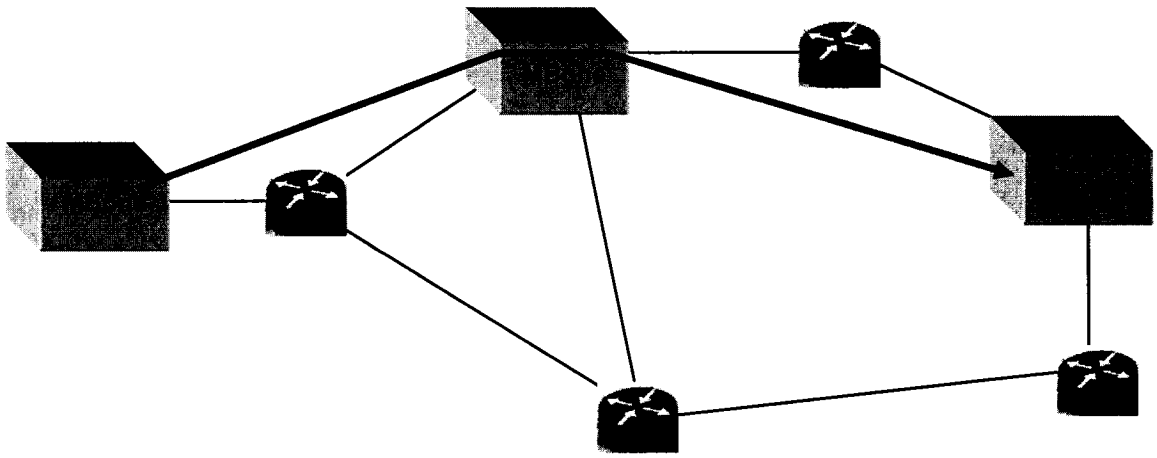

Figure 10: Multicast backbone

So, with the help of this tunneling concept any new protocol data can be transported using the overlay networking nodes. But in doing so, necessary software have to be installed and configured on the nodes.

Peer-to-peer Networks

Peer-to-Peer is a distributed network architecture where each node shares its recourses such as processing power, bandwidth, storage capacity, etc. Usually nodes in peer-to-peer system are loosely coupled with each other and play an equal role (client/server) in the system. Moreover nodes are autonomous which makes it difficult to construct a robust peer-to-peer system. In a pure peer-to-peer network, there is no notion of clients or servers, but only equal peer nodes that simultaneously function as both "clients" and "servers" to the other nodes on the network [36]. This model of network architecture differs from the client-server architecture where communication is usually to and from a server. FTP is an example of non peer-to-peer file transfer system where the client and

server programs are quite distinct, and the clients initiate the download/uploads and the servers react to and satisfy these requests.

Another very important application of overlay networks is Application-Layer multicasting which we will be covered in the next section.

2.4 Application Layer Multicasting

Much research has been done to compensate for network lag in order to provide better quality for distributed simulations. Some of these studies provide receiver-initiated and selectively reliable transport protocols [17] that can be used to deliver important messages with a high degree of reliability, while others use sender-initiated approaches to transmit *key updates* with guaranteed reliability [8]. The IEEE DIS standard [18] has also been successfully used in controlled environments with vast resources, mostly for military simulations. These approaches; however, are all based on IP multicasting and although they achieve good results in Intranet environment, they are not readily deployable on the Internet.

The lack of applicability of IP multicasting on the Internet has been well documented [19] [20]. Reasons include scalability; the fact that IP Multicasting is designed for a hierarchical routing infrastructure and does not scale well in terms of supporting large number of concurrent groups; the deployment hurdles caused by manual configuration at routers and Internet Service Providers' unwillingness to implement IP multicasting, and marketing reasons due to the undefined billing at the source (content provider) and receivers.

An alternative has therefore been proposed to shift multicast support from the networking layer to end systems. This is Application Layer Multicasting (ALM). In ALM, data packets are replicated at end-hosts instead of at routers. The end-hosts form an overlay network, and the goal of ALM is to construct and maintain an efficient overlay for data

transmission. This is demonstrated in Figure 9. Since the routing information is maintained by the application, it is more scalable than IP multicasting since it can support large number of concurrent groups. Also, because ALM needs no infrastructure support, it is fully deployable on the Internet. In theory, Content Providers can deliver bandwidth-intensive contents such as TV programs and interactive networked games to vast number of clients via the Internet by using ALM. This was impractical before because the bottleneck bandwidth between content providers and consumers is considerably less than the natural consumption rate of such media.

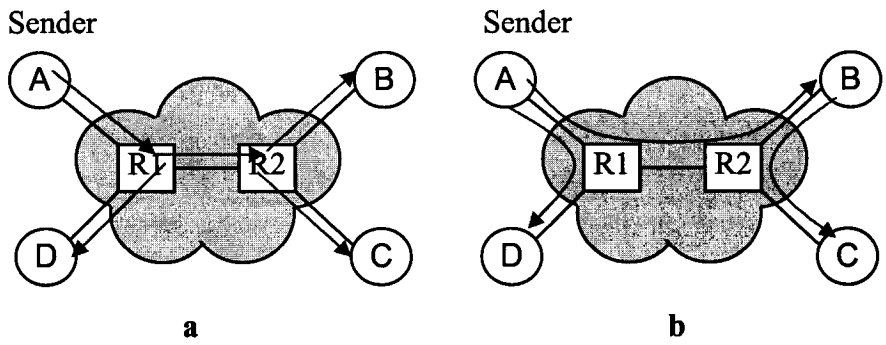

Figure 11: Network layer multicasting (a) versus application layer multicasting (b): square nodes are routers and circular nodes are end-hosts.

However, ALM does come with a trade-off: more bandwidth and delay (compared to IP multicasting) for the sake of supporting more users and scalability. But it has been shown that ALM-based algorithms can have "acceptable" performance penalties with respect to IP Multicasting, and compared to other practical solutions [21].

2.4.1 Application Domains

Perhaps the most crucial feature of an ALM protocol and one that affects most of its resulting characteristics is its targeting application. The application domain determines the number of users the protocol must support, the data type(s) that a protocol's delivery tree must accommodate and the metrics that such a tree attempts to optimize. Inspired by the categorization of Diot [22], application domains can be categorized as follows:

1. Multiplayer network games: protocols for this application must take into consideration the high level of interactivity between players. The primary metric is latency and bandwidth to a lesser extent.
2. Audio/video streaming: usually involves a single source distributing media to a large number of receivers. Examples include live streaming of a sporting event, or streaming of pre-recorded news. The primary metric is bandwidth and latency to a lesser extent.
3. Audio/video conferencing [33]: these involve small to medium size groups interacting in a multi-party conferencing session. The difference with the previous category is that group size is smaller but with higher degree of interactivity and the existence of multiple sources. Both bandwidth and latency are important metrics.
4. Generic multicast service: protocols falling into this application domain category try to create a generic multicast service based on specific metrics that can affect a variety of applications.
5. Reliable data broadcast and file transfer: reliable transfer and distribution of (usually large) files (e.g. distributed databases and file sharing), with Bandwidth being the only metric.

As can be seen, the different classes of applications have different sets of requirements regarding reliability, latency and bandwidth. Such requirements in turn determine the design choices of any ALM protocol regarding the group management mechanism it employs.

2.4.2 Deployment Levels

A key factor to determine the set of assumptions of ALM protocol operations is based on the level the protocol is expected to be deployed at. ALM protocol can be deployed at the proxy-level or end-system level. Proxy-based ALM protocols require the deployment of dedicated servers/proxies on the Internet where they self-organize into an overlay network and provide a transparent multicast service to the end-user. On the other hand, end-system level protocols utilize only a unicast service from the infrastructure and

expect the end-system hosts to participate in providing the multicasting functionality by taking on some of the forwarding responsibility (figure 10).

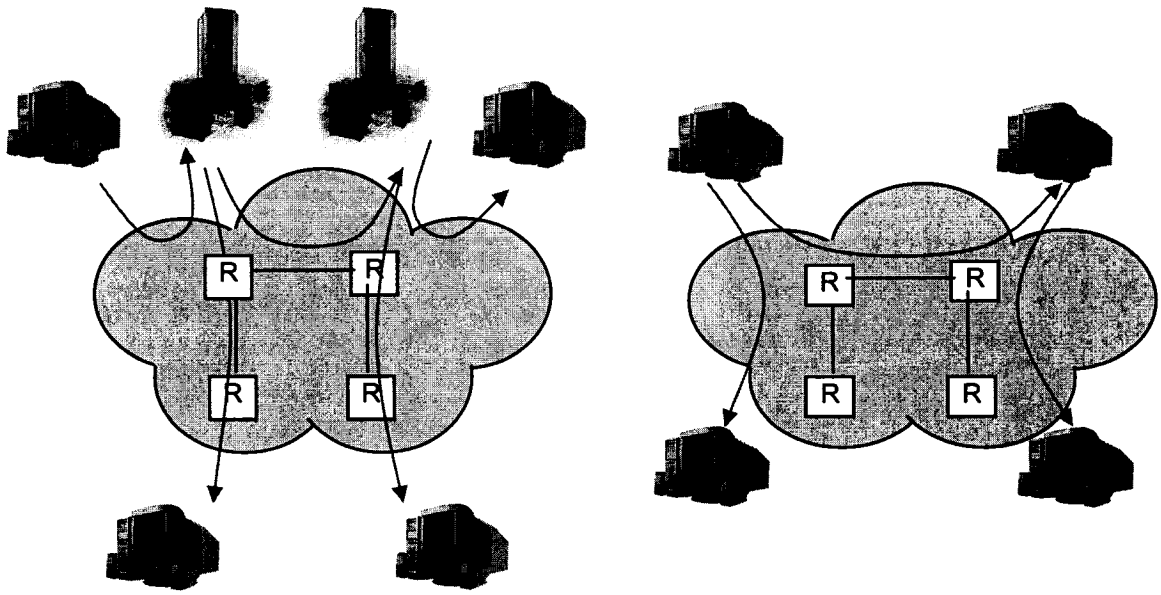

Figure 12: Proxy-based (left) and End-system (right) deployment of ALM

The choice between selecting an infrastructure level or an end-system level ALM protocol is perhaps driven as much by business and marketing issues as does the technological ones. End systems sharing the forwarding load of a multicast session use the existing Internet infrastructure available to them and may not be expected to pay more for participating in the multicast session, as illustrated by the free nature of peer-to-peer file-transfer applications. An infrastructure of dedicated proxies deployed over the Internet that offer multicasting services however are more likely to expect a service charge. In terms of technological consequences, Proxy-based ALM protocols can take advantage of existing IP Multicast 'islands' and therefore increase their efficiency, assume greater bandwidth availability to the proxy nodes, and assume longer life cycle of overlay nodes. The major disadvantage of this approach is the need for the deployment of dedicated proxies over the inter-network and so incurring the cost associated with acquiring and deploying them. End system ALM protocols enjoy more flexibility, adaptability to specific application domains, and immediate deployment over the Internet

but may not scale well to large number of users or large number of simultaneous sessions, and must deal with limited bandwidth of user machines. This will also require the end-system to take on some of the forwarding responsibility and therefore increase application software development complexity.

2.4.3 ALM Protocols

Application layer multicasting protocols have been extensively investigated in the past few years. A thorough coverage for those protocols is beyond the scope of this thesis. The reader is referred to [29] and Appendix A for more detailed information about existing ALM protocols and their different classifications. In the following section we will briefly cover three ALM protocols that have some relevance to our research.

2.4.3.1 A Generic Proxy System for Networked Computer Games [23]

The generic proxy system (Figure 11) combines the advantages of centralized and distributed approach. The objective of this architecture is to provide congestion control, to achieve robustness, to minimize the impact of network delay, and provide fairness.

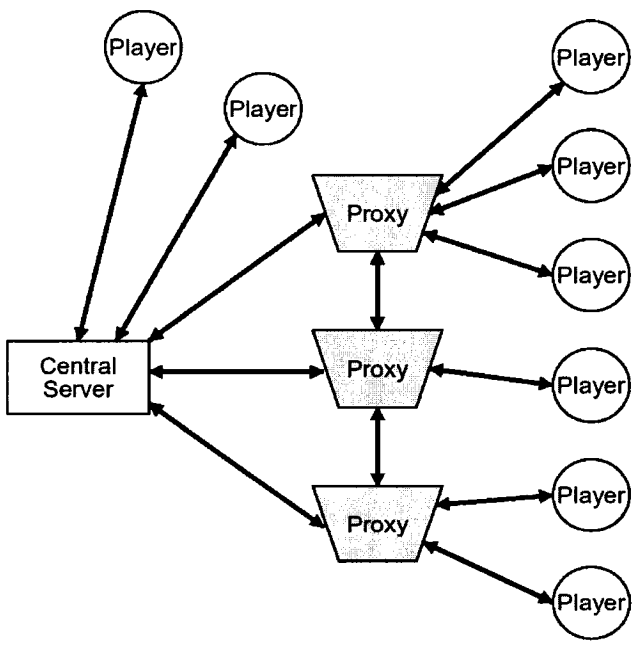

Figure 13: Generic proxy architecture for computer games

Mauve [23] argues that this can be achieved by moving server functionality to the boundaries of the network. Moreover, by placing proxies close to the clients, it will be quite possible to move functionalities from the clients to the proxies to gain more control over the players' actions. This will more reliably prevent cheating, and improve the overall game performance.

2.4.3.2 Game State and Event Distribution using Proxy Technology and ALM

This is part of ongoing research by Knut-Helge Vik [24] at the University of Oslo. The protocol focuses mostly on proxy technology rather than client/server or P2P technologies. Vik [24] argues that in client/server approach the server is a bottleneck because it is a centralized architecture. Although peer-to-peer is distributed as proxy technology they have discarded it simply because it is very hard to administrate the game state and make it consistent between loosely coupled hosts. Data exchanged among participants is classified into three categories:

- Requiring varying degree of consistency. For this data, proxy technology is used.
- Does not affect the game state. For this data peer-to-peer is used.
- High consistency requirements. Client/server is utilized in this case.

2.4.3.3 Application Layer Multicast for Efficient Peer-to-Peer Application [25]

A new algorithm, called Fastcast, is introduced. It is root based, online, and topology-aware ALM, Fastcast is controlled by a parameter, and by changing this parameter it is possible to control the trade-off between used traffic and the worst-case length of the application layer path. ALM algorithms for peer-to-peer applications can benefit from the possibility of limiting the number of children in the ALM tree, since many nodes could be connected by slow modem connections.

2.4.3.4 BMP: an Efficient and Scalable Multicast Protocol [26]

BMP stands for Branching based multicast protocol. Branching point approaches provide high degrees of stability, incremental deployment and require low memory. Branching based multicast protocol (BMP) is also a BP based protocol, where many joint processes

can be done simultaneously and the process is localized. Two main assumptions are set a priori. Firstly, BP must be aware of its children and of its parent BP in the reduced tree. The second assumption is that every none-BP must be aware of the next BP in the tree. The BMP packet forwarding method has very little impact on unicast packet forwarding. It avoids packet duplications in other BP based approaches which occurs in the presence of network asymmetry.

2.4.3.5 Construction of an Efficient Overlay Multicast Infrastructure for Real-time Applications [27]

In this approach overlay architecture is considered, where service providers deploy a set of service nodes called Multicast Service Nodes (MSN). This MSN deployment in the network is used to efficiently implement media streaming applications. The role of these MSNs is to forward data for a set of clients based on application layer multicasting scheme. The following are some characteristics of the proposed approach:

1. It's optimization criteria is "degree-constrained minimum average latency problem"
2. MSN is dynamically prioritized based on the size of its service set.
3. The overlay tree is iteratively modified using localized transformations to adapt with the changing distribution of MSN clients, as well as network conditions.

2.4.3.6 Approximation and Heuristic Algorithms for Minimum Delay Application-Layer Multicast Trees [28]

These algorithms can be effectively used to implement centralized overlay systems, such as peer-to-peer and server based systems. They show how to directly map the node load to the delay penalty at the application host, and create a new model that captures the trade offs between the desire to select shortest path trees and the need to constrain the load on the hosts. Two delay measures are considered. These are processing delay and communication delay.

2.4.3.7 ProBass [29]

ProBass is a proxy based single source ALM protocol. ProBass targets mainly media streaming applications. While dedicated proxy servers are self-organized into an overlay backbone and they provide multicast services to end-systems, end-systems' responsibility is to forward the data to the other end-systems via ALM tree links.

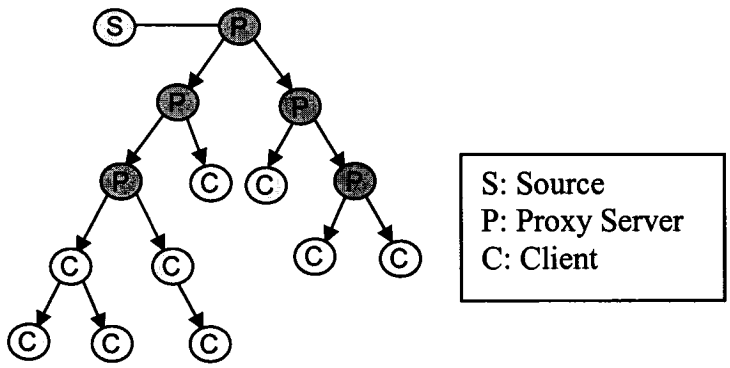

Figure 14: Overlay multicast architecture of ProBass

ProBass considers two metrics for building the tree: bandwidth and end-to-end delay. Out degree and end-to-end delay reflect bandwidth and end-to-end distance respectively. Proxy servers are the entry point and the exit point of the system. Users joining or leaving the session must go through the proxy server. ProBass provides an efficient mechanism for handling users leaving the session early.

2.4.4 ALM Protocols and Distributed Collaboration

Distributed simulation in collaborative environments has special requirements, and communication between entities participating in the simulation should be addressed differently than communication in another context. From the above-discussed protocols it can be clearly seen that none of them address the issues of distributed collaboration on the Internet. The most important metrics covered by those protocols are bandwidth and end-to-end delay. ALM protocols for Distributed simulation in CVEs need to put into consideration the nature of collaboration data. Moreover, for closely coupled tasks, there is an important metric, which is the reliability of certain data and synchronization between entities in a timely manner.

In this thesis, it is believed that the emergence of a hybrid ALM protocol that handles different communication approaches, such that it is suitable for different platforms and takes into account the nature of collaboration data, is of imminent need. In chapter 3 a detailed explanation of this hybrid protocol algorithm is presented along with theoretical evaluation.

Chapter 3

The Proposed Protocol: HDSP

In this section we propose a multi-source ALM protocol. The protocol developed and implemented is referred to as HDSP: Hybrid distributed simulation protocol. HDSP is developed to support all kinds of simulation data. More specifically, it takes into consideration data that have tight collaboration requirements. The protocol supports media streaming and tightly collaborative network games etc. Lag and data loss are more significant in tightly coupled network games than other applications, therefore, as a proof of concept, HDSP has been integrated with a collaborative network gaming application. HDSP incorporates ALM technology when users are on the Internet. Users in the LAN are treated differently as it will be seen in Chapter 4. In the following section, discussion of HDSP-ALM algorithm is presented.

3.1 HDSP-Algorithm

As mentioned previously, HDSP is a multi-source ALM protocol. This means every node in the tree can send and receive data. Nodes in the tree are end-systems. The overlay multicast architecture for the ALM tree is shown in Figure 13.

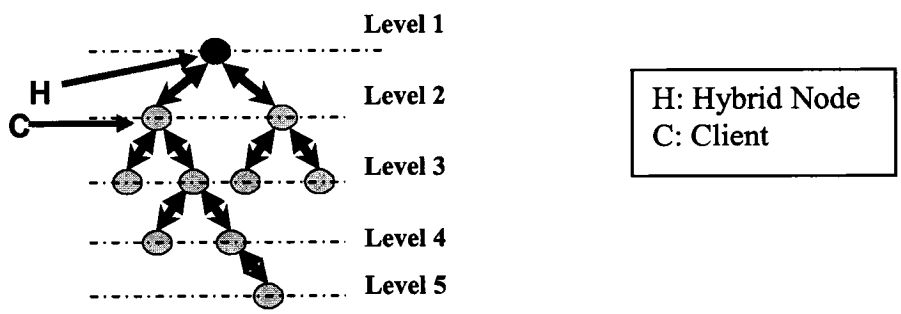

Figure 15: Overlay multicast architecture of HDSP-ALM

The hybrid node (Figure 14) resides in the Local Area Network where it acts as a mediator between the LAN and Internet users. Besides mediation, the hybrid node is responsible for building and maintaining the ALM tree. It directs new joiners to their potential parents and also reconstructs the ALM tree dynamically in case one node crashes or leaves the session early. Other nodes in the tree are responsible for sending data due to changes made by users or forwarding received data from other end-systems. It is assumed that the hybrid node and the end-hosts have already been organized into an overlay tree before data delivery commences.

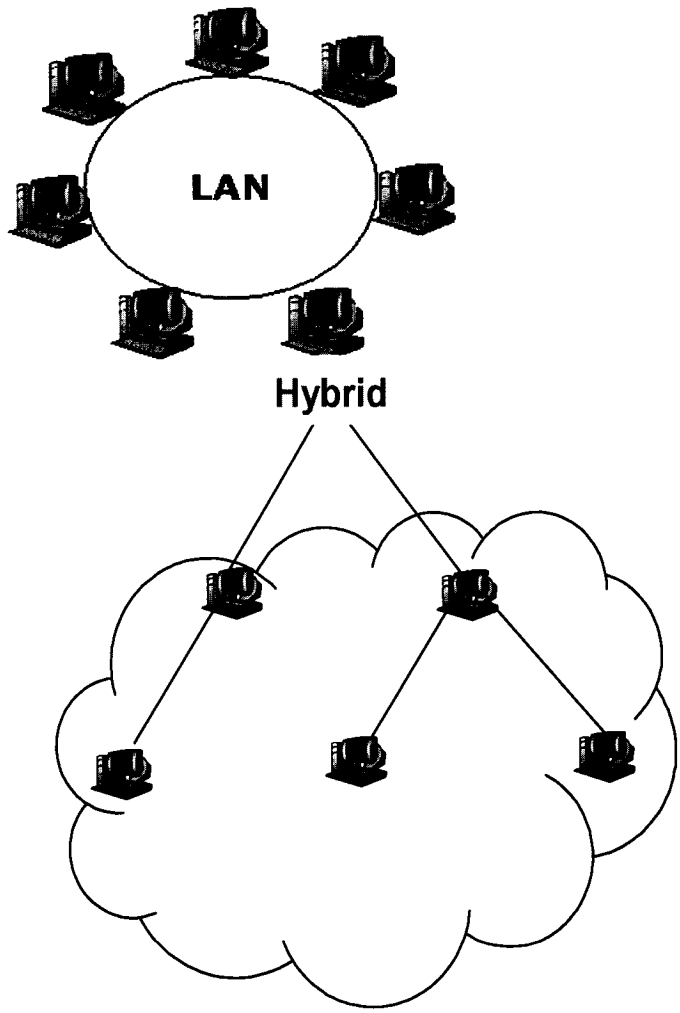

Figure 16: Hybrid node mediating between LAN and Internet users

3.1.1 Rationale

HDSP considers three main metrics in building the ALM tree. Two of the metrics are in-line with collaboration data requirements. The first metric is the bandwidth which can be controlled by the out degree parameter. Each node in the tree has an out degree bound. The out degree reflects the maximum bandwidth each node can provide. Since HDSP supports frequent short messages transmitted between users, large number of users can connect to each level of the tree. The second metric is the end-to-end distance. The node-to-node round trip time (RTT) can serve as the distance metric in building the tree. Since nodes are assumed to be in the same vicinity, the second parameter, which is the distance, is not as critical as the first parameter, which is the bandwidth. The third metric HDSP considers is the reliability of "key" updates. This is achieved by implementing SCTP to guarantee the delivery of critical messages. Considering the third metric along with supporting many communication modes makes HDSP significantly different than other ALM protocols.

3.1.2 Member Join

We assume in our algorithm that the hybrid node is always alive and waiting for clients' requests. The hybrid node is the entry point of the system. When a new node wants to join the tree it must query the hybrid node such that the new node will know its potential parent. The potential parent is either the hybrid node itself or another node in the ALM-tree. After the hybrid node receives a request it performs a check on its children to see if the current number of children are greater than or equal to the predefined out degree. If the hybrid node can accommodate more immediate children then it will accept the new node as a child and send an acceptance message to it. Otherwise, the hybrid node will direct this new node to another node in the tree. In the later case, the hybrid node will also send a message to the potential parent with the new node ID. In both cases, the potential parent (hybrid or tree-node) will increase its number of children counter, after accepting a node to be a child. Also the hybrid node will increase the total number of nodes when a new joiner is accepted. There is another rare scenario that could happen when the total number of nodes in a tree exceeds the upper bound of the number of nodes.

In this case the hybrid node will suspend the request of this node for a limited number of seconds. The new joiner will be pending for some time before it resends another request. If the hybrid node receives three consecutive joint requests from the same node and the number of nodes are still equal to the upper bound then it will reject this request. As mentioned earlier, the hybrid node has full information about the structure of the ALM tree. The hybrid node keeps record of the ALM-tree state and updates it based on the new information it has from new joiners or nodes leaving the tree. Figure 15 illustrates the join process.

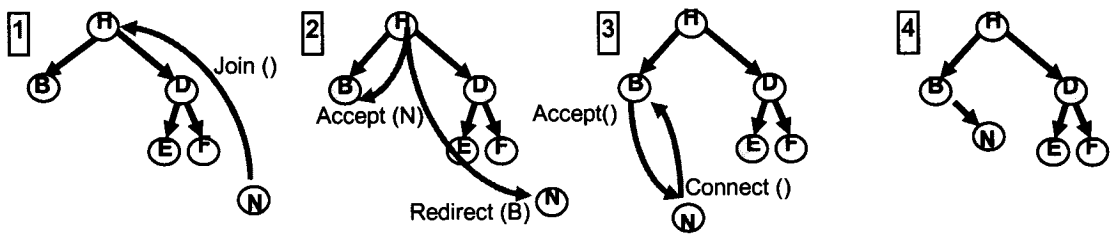

Figure 17: Member join algorithm

The following UML diagram (Figure 16) depicts the process of join, acceptance and redirection. Those operations are fundamental for the ALM-tree construction and update:

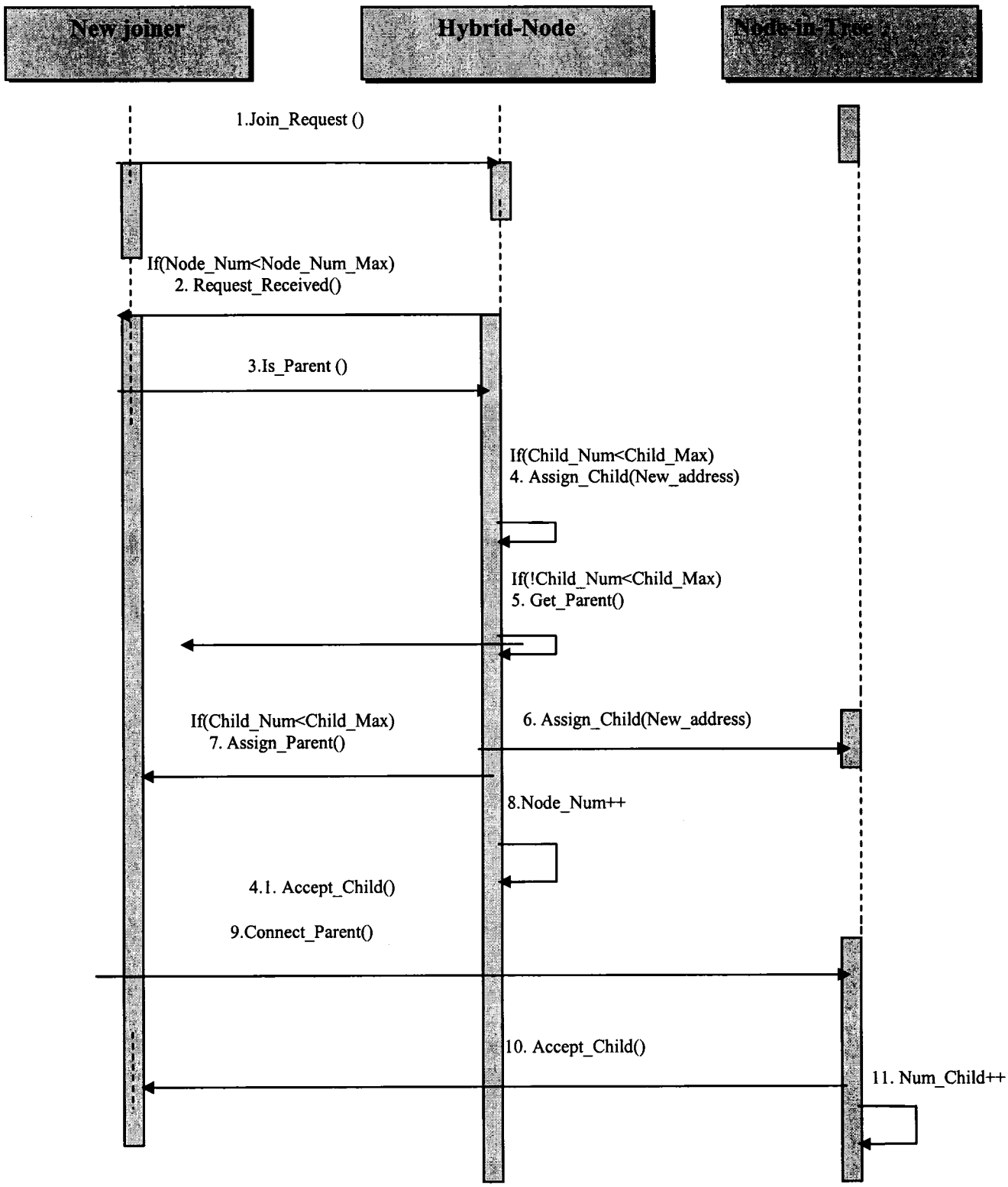

Figure 18: ALM-tree construction

3.1.3 Member Leave

HDSP allows two types of leave: early leave and unexpected leave. The latter case happens when one node crashes or leaves without informing the group. In the former case the leaving node sends a leave messages to the hybrid node, to its parent and to its children. The parent of the leaving node will decrease the number of children variable. Children of the leaving node have to request the hybrid node for reassignment of a new parent. Figure 17 illustrates the first case of member leave algorithm.

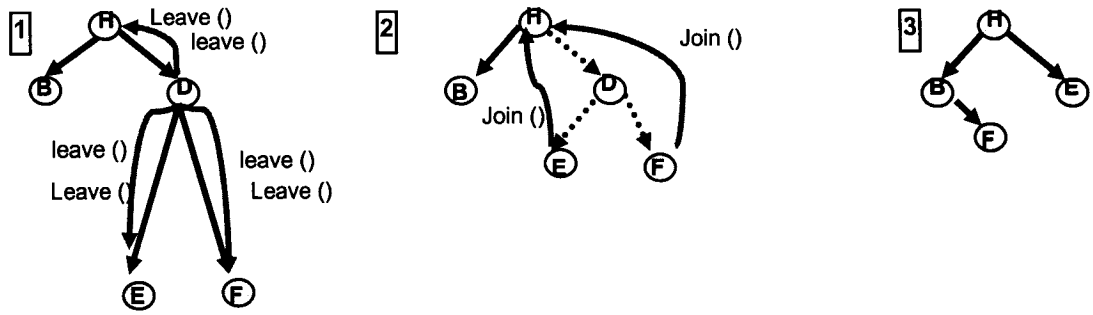

Figure 19: Member leave algorithm

The second case of member leave happens when one node crashes or leaves without informing the group. To handle this case HDSP make use of keep-alive messages between adjacent nodes. Each node sends keep-alive messages to its parent and children. If a node does not respond to these messages then the neighbours assumes its death and a join request will be sent from children to hybrid node. The hybrid node in this case verifies if the reported node is still alive; and based on that it reconstructs the tree.

3.1.4 Routing Packets

HDSP is a multi-source protocol. This means each node can send or receive data to or from other adjacent nodes. If a node decides to send a packet then it should send the data to all adjacent nodes (parent and children). When a node receives a packet it immediately forwards it to other adjacent nodes. Data sent or received will be categorized as key or non-key updates. Based on this categorization, HDSP uses simple UDP protocol or SCTP protocol according to the type of data exchanged. UDP protocol will be used for data that

does not require reliability, i.e. non-key update while SCTP is used for data that require high level of reliability, i.e. key update. When a node sends a key update then it will expect an acknowledgment from destination. The node will retransmit the update if one or more receivers do not acknowledge back. Simple UDP protocol is used to transmit regular update.

3.2 Theoretical Evaluation

In the following three subsections a theoretical evaluation of HDSP architecture and traffic management is presented.

3.2.1 Architecture

HDSP uses peer-to-peer architecture with the possibility of incorporating proxies to ensure fairness and even distribution of session states among users. Peer-to-peer architecture was chosen instead of client/server architecture because in the latter architecture the server is a bottleneck because it is a centralized architecture. Since peer-to-peer architecture is distributed it is very hard to administrate the session state and make it consistent between loosely coupled hosts. HDSP overcomes this difficulty by making the hybrid node, which is a simple host, administer of the session. Moreover, the hybrid node is responsible for global session state through restructuring the tree in case of other nodes leaving or crashing.

3.2.2 Traffic Management

HDSP classifies the data into key and non-key updates. Through this classification HDSP can handle any type of simulation data with efficient bandwidth usage in a timely manner and with guaranteed reliability for important data. Instead of acknowledging all messages, HDSP requires only key update messages to be acknowledged. Beside UDP, HDSP makes use of SCTP to handle collaboration of specific data.

3.2.3 ALM-tree Infrastructure

In building the tree, there are several possible scenarios that could affect the state of the ALM-tree. These scenarios include early and unexpected member leave. HDSP allows the tree to be rebuilt in case some nodes leave the session. Moreover, HDSP can detect the absence of one member due to a failure and report that to the hybrid node which will reconstruct the ALM-tree. Detecting the absence is achieved through sending keep-alive messages in a periodical manner.

Chapter 4

Communication Framework and System Design

4.1 Communication framework

The proposed Hybrid Distributed Simulation Protocol (HDSP) is tailored to provide transport services suitable for all types of simulation data: both tightly synchronous collaboration data and generic simulation data. HDSP is a multi-source protocol that provides best effort LAN multicast, timely reliable LAN multicast, best effort peer-to-peer delivery on the Internet, timely reliable peer-to-peer delivery on the Internet, and any combination of the above. Hence, it is called a hybrid protocol.

4.1.1 Best Effort LAN Multicast

For data transmitted on the LAN, HDSP uses best effort LAN multicast mode if transmitted data does not require reliability, such as continuous avatar updates. If one or many users in the LAN do not receive data packets then retransmission is not enforced. For this, HDSP simply implements normal UDP protocol. Every node in the LAN joins the session through a multicast socket. Updates are sent and received from the multicast socket. In each end-system there is a thread running and listening through the multicast socket. Notice that the entry point for a new joiner is the multicast socket and not the hybrid node. A node will join the session by sending a "hello" message to the multicast socket. Each member will reply back with another "hello" message. When the hybrid node receives a "hello" message it will update the session state and send it to the new node such that the new user has the same view of the session as the other users. The hybrid node will increase the counter of the users by one. Figure 18 shows the system architecture for Best Effort LAN Multicast.

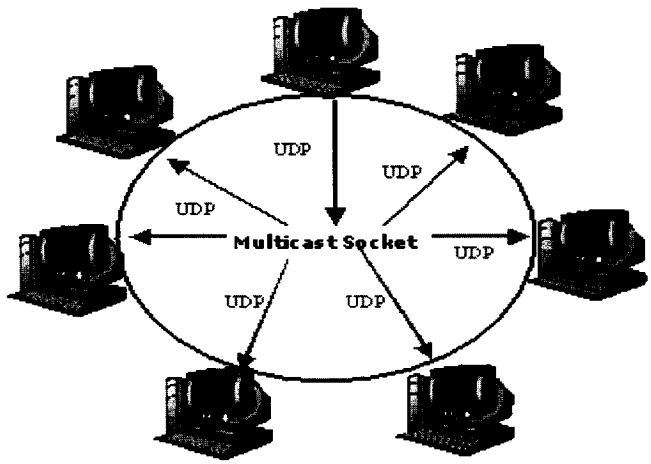

Figure 20: System architecture for Best Effort LAN Multicast

4.1.2. Timely Reliable LAN Multicast

HDSP uses Timely reliable LAN multicast for data that must be received reliably and in a timely manner by all users. For this mode, an ACK-based reliable multicast protocol must be used to guarantee data delivery. SCTP packets are used to transmit "key" updates. This approach has been shown to support tightly coupled collaboration between end-systems [8]. When a packet is sent the sender invokes a timer. Before the timer expires the sender must receive acknowledgements equal to the number of receivers; otherwise the packet will be retransmitted again. Figure 19 shows the system architecture for timely reliable LAN multicast.

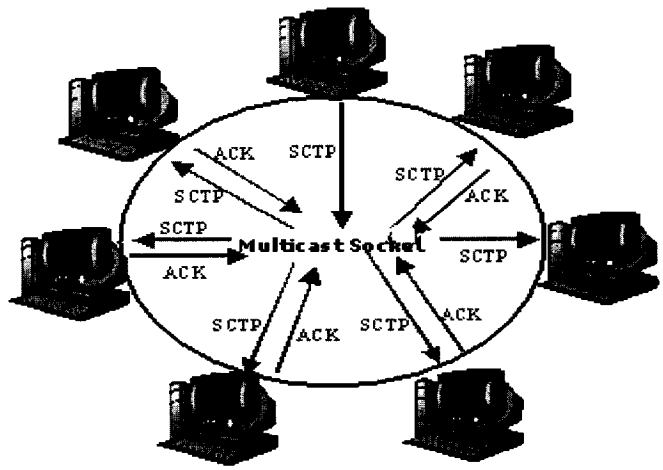

Figure 21: System architecture for Timely-reliable LAN Multicast

4.1.3. Best Effort Peer-to-Peer Delivery on the Internet

HDSP utilizes best effort peer-to-peer delivery on the Internet for generic simulation data that must be received by members in the group. Data transmitted does not require reliability; therefore no retransmission is handled in this mode. When a node receives data it makes the necessary state update and immediately relays it to the other adjacent members. Figure 20 depicts the system architecture for Best effort peer-to-peer delivery on the Internet. One can see that Node A sends a packet to Node B only. The packets is replicated from Node B to H which relays it to E. Node E forward that packet to Node C and D. Node A sees only Node B but the data sent reaches all members in the ALM tree.

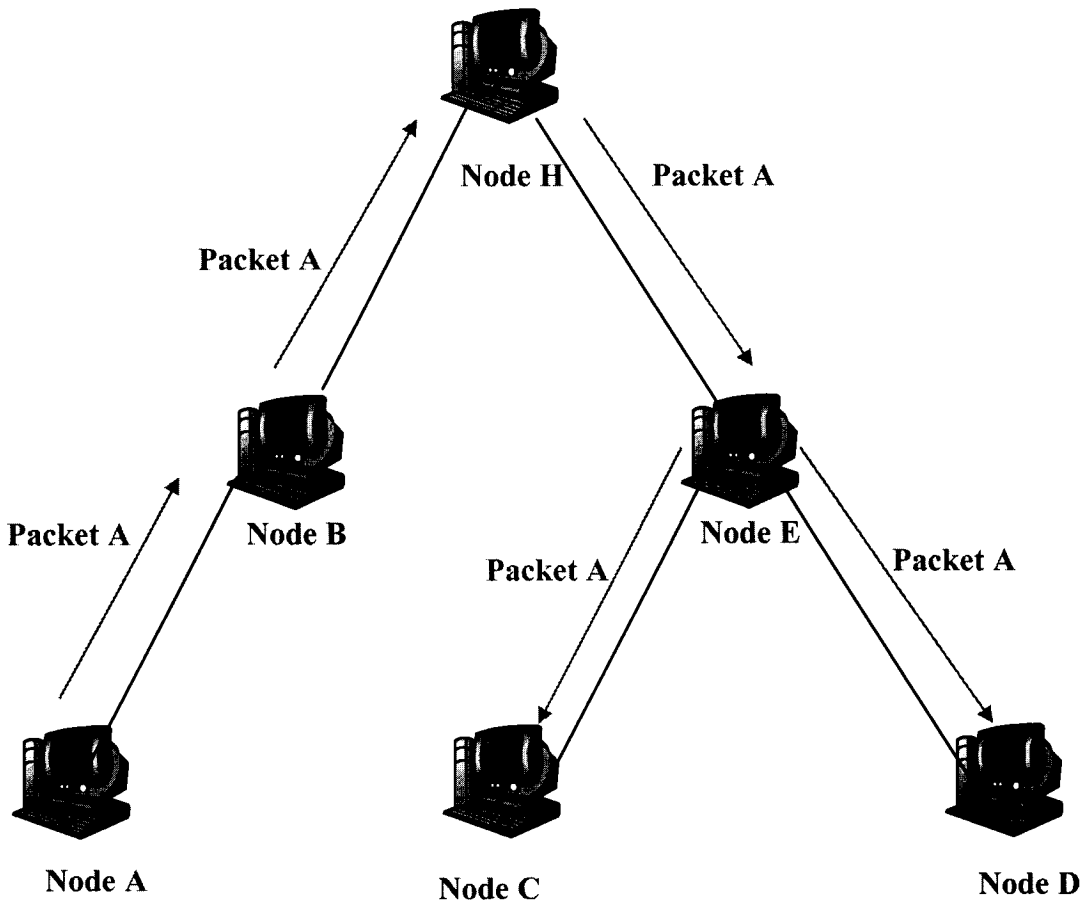

Figure 22: Best Effort Peer-to-Peer delivery on the Internet

4.1.4 Timely Reliable Peer-to-Peer Delivery on the Internet

Similar to 4.1.2, this mode is used for tightly synchronous data, except that data is sent on the Internet. For this purpose, HDSP utilizes timely reliable peer-to-peer delivery to transmit "key" updates. The data transmitted requires high degree of reliability; therefore retransmission is handled by this protocol in case there is lost or damaged packet. For this mode, an ACK-based reliable protocol (SCTP) is utilized. When a key update is sent, a timer is invoked. The sender should receive acknowledgements equal to the number of recipients. The number of recipients could be equal to the number of adjacent nodes or to the number of adjacent nodes –1. The former case is when the sending node originates the update while the latter case occurs when the sending node is relaying data. Notice that acknowledgments are not relayed. This mode also uses the same peer-to-peer distribution tree described in chapter 3. Figure 21 depicts the system architecture for Timely-reliable Peer-to-Peer delivery on the Internet.

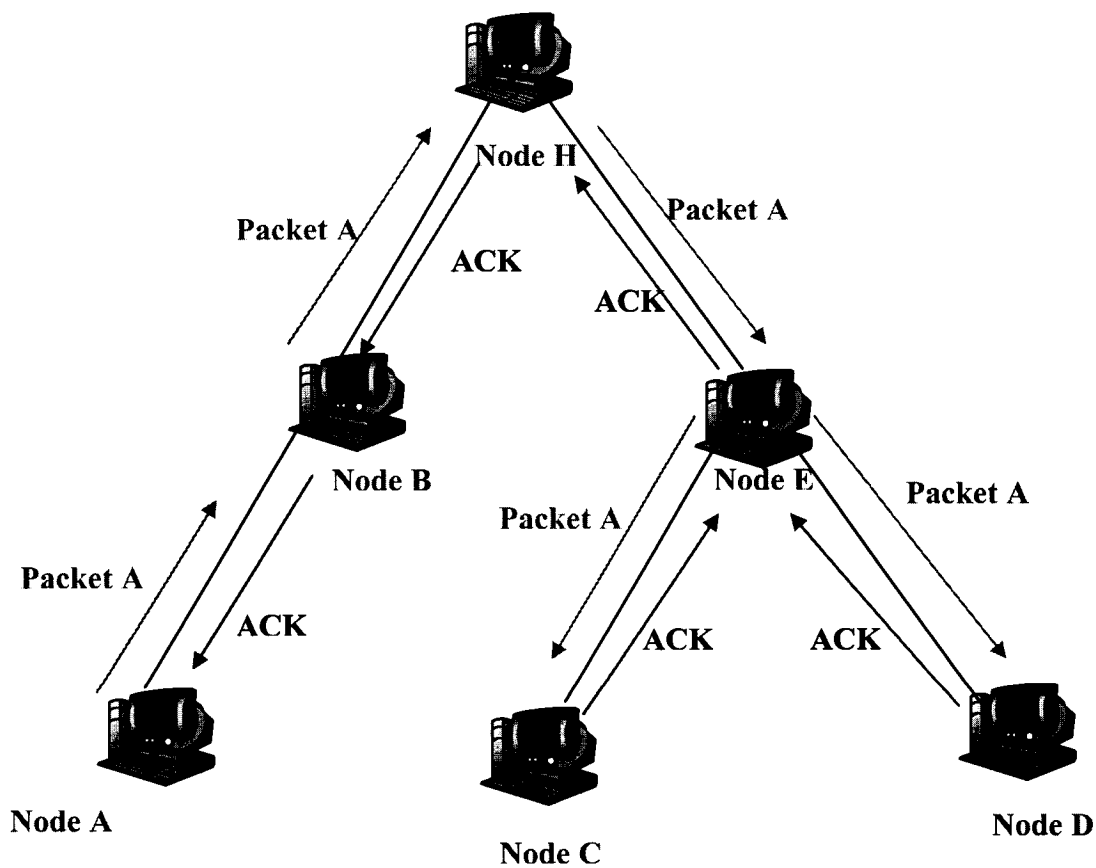

Figure 23: Timely-reliable Peer-to-Peer delivery on the Internet

4.1.5 Combination of all Services

HDSP is able to combine any of the above services at the same time and hence support LAN users and Internet users simultaneously collaborating in the same virtual simulation. Figure 22 shows System architecture.

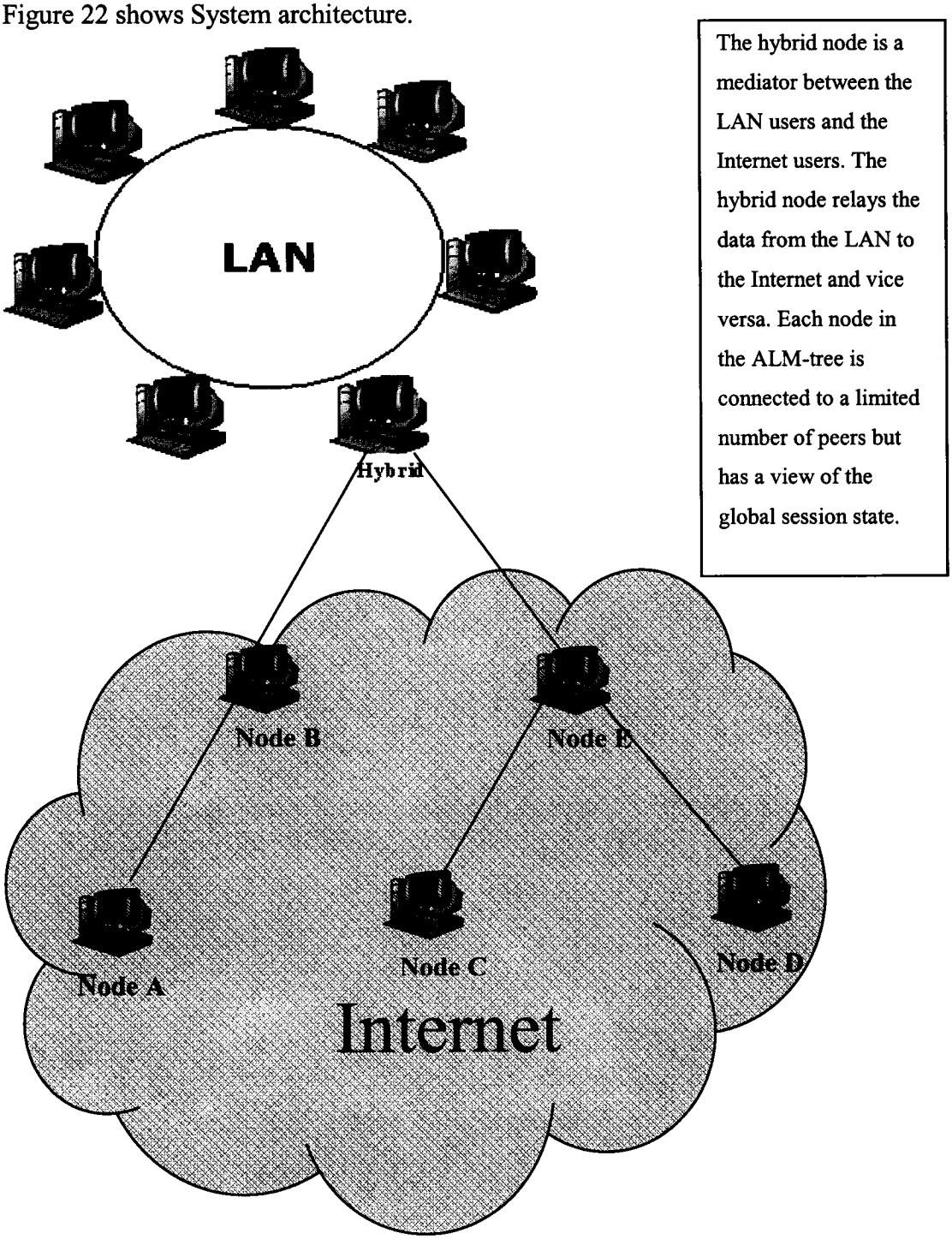

Figure 24: HDSP system architecture

The same procedure for sending and forwarding updates will be applied here except that the hybrid node will relay the updates received from the Internet to the LAN and vice versa.

4.2 Use Cases

In this section we present our system design utilizing use cases and class diagrams. The diagrams are drawn using the UML tool Rational Rose Enterprise Edition. Use case diagrams are shown in Figure 23 - Figure 27 along with their descriptions.

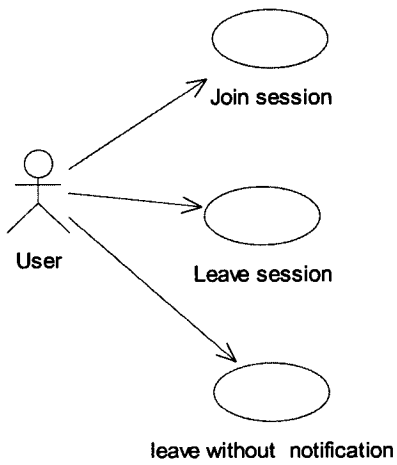

Figure 25: Use case Diagrams 1

Join session:

Actor: User

Description:

1. The Internet user joins the session by sending "hello" message to the hybrid node.
2. The LAN user joins the session by sending "hello" message to multicast socket.
3. The Hybrid node will check the availability in the ALM-tree and reply back with "accept" or "redirect" message to Internet users.
4. The Hybrid node will increase the counter of the LAN users by 1 after a new member joins.
5. Potential parent will be informed of the new joiner address and information.

6. The new joiner will query the potential parent after receiving its address.

Leave session:

Actor: Internet user

Description:

1. The user sends a "leave" message to its parent, children and hybrid node.
2. The parent of the leaving node will decrease the number of children counter by 1.
3. The children of the leaving node will send a join request to the hybrid node.
4. The hybrid node will rearrange the ALM-tree while relocating the orphan children.
5. The hybrid node will decrease the ALM-tree members by 1.

Leave without notification:

Actor: Internet user

Description:

1. The user leaves the session without sending "leave" message.
2. Adjacent neighbors detect the absence of the leaving node.
3. The parent of the leaving node will decrease the number of children counter by 1.
4. Children of leaving node send a join request to the hybrid node.
5. Hybrid node relocates the orphan children and restructures the ALM-tree.
6. The hybrid node will decrease the ALM-tree member counter by 1.

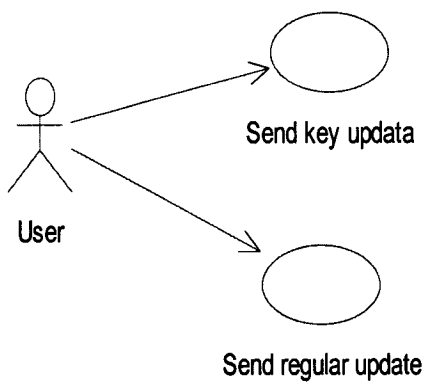

Figure 26: Use case Diagrams 2

Send key update:

Actor: user

Description:

1. The user sends a "key" update for critical data.
2. The sending node invokes a timer and waits for acknowledgements from its neighbors.
3. Sending node will retransmit message if one or more acknowledgements are not received after the timer expires.

Send regular update:

Actor: user

Description:

1. The user sends a "non-key" update for regular data.
2. No timer is invoked and the sender expects no acknowledgements to be received.

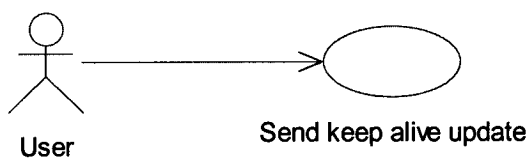

Figure 27: Use case Diagrams 3

Send keep-alive update:

Actor: user

Description:

1. The user sends keep-alive messages to its neighbors periodically.
2. Recipients of keep-alive messages must acknowledge back.
3. The user will assume the death of its neighbor if the neighbor does not reply after the sending of three consecutive keep-alive messages.

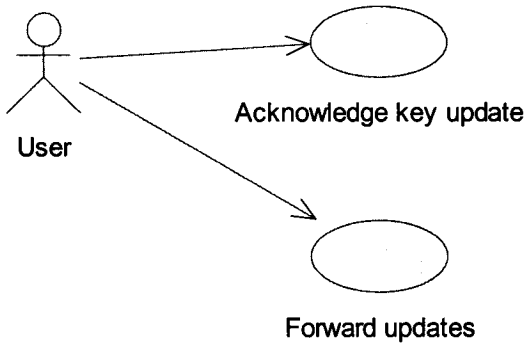

Figure 28: Use case Diagrams 4

Acknowledge key update:

Actor: user

Description:

1. If the received message is a "key" message then acknowledge back the sender if receiver is in the Internet. If receiver is in the LAN, send acknowledgement to multicast socket

Forward update:

Actor: user

Description:

1. All received messages (key or non-key) are broadcasted to all adjacent neighbors except the sender.
2. Acknowledgements are not to be relayed.

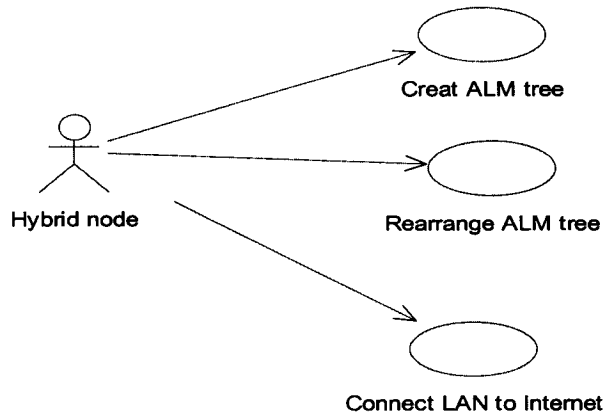

Figure 29: Use case Diagrams 5

Create ALM-tree:

Actor: Hybrid node

Description:

1. Hybrid node accepts new joiners as children or redirects them to potential parents.
2. Hybrid node keeps record of the ALM-tree state and structure.

Rearrange ALM-tree:

Actor: Hybrid node

Description:

1. Hybrid node redirects orphan children to new parents.
2. Hybrid node updates tree after new member joins or one member leaves.

Connect LAN to Internet:

Actor: Hybrid node

Description:

1. Hybrid node relays data from LAN to Internet and vice versa.

4.3 Class and Sequence Diagrams

HDSP framework classes are shown in Figure 28. Class attributes and operations are

shown in details in Figure 29 - Figure 34. Sequence diagram describing the process of joining the ALM tree is portrayed in Figure 35.

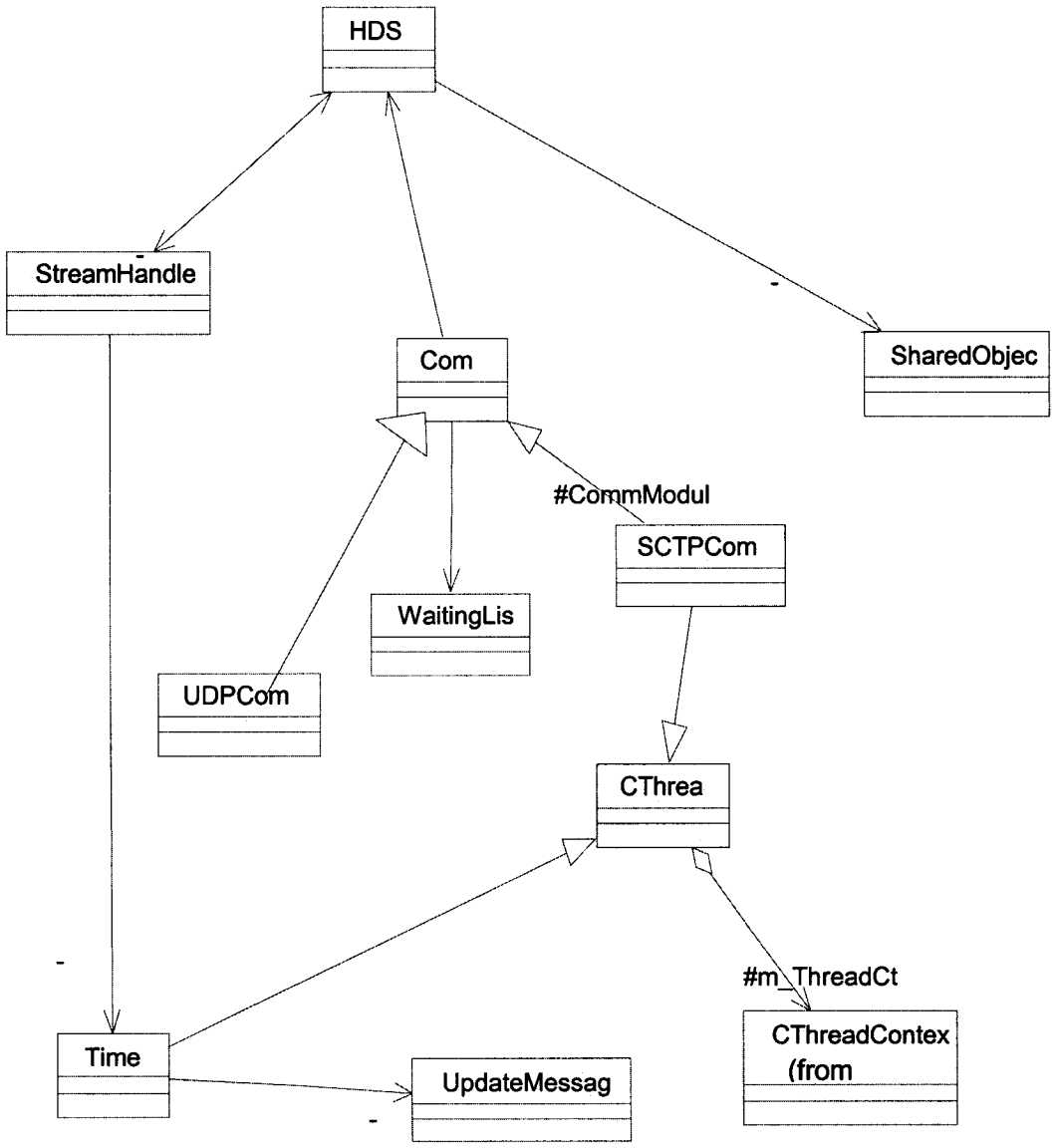

Figure 30: Class diagram for HDSP framework

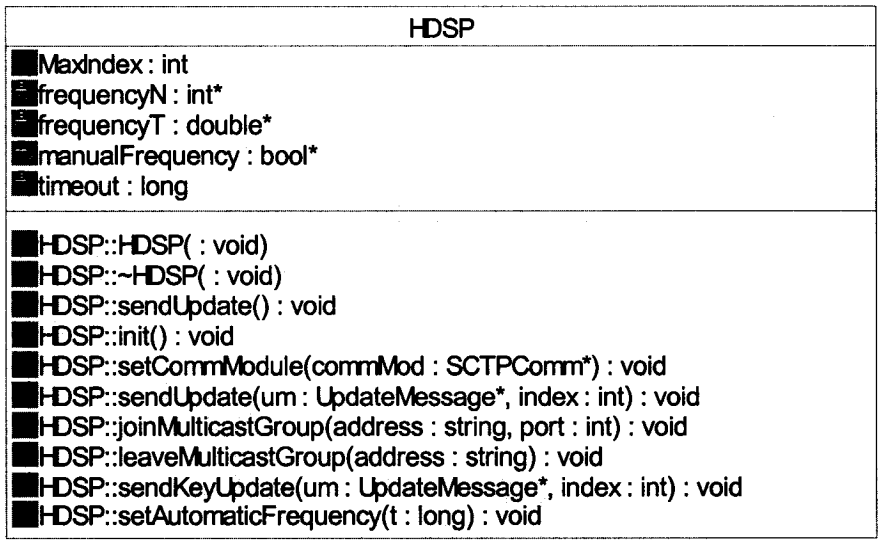

Figure 31: HDSP class diagram

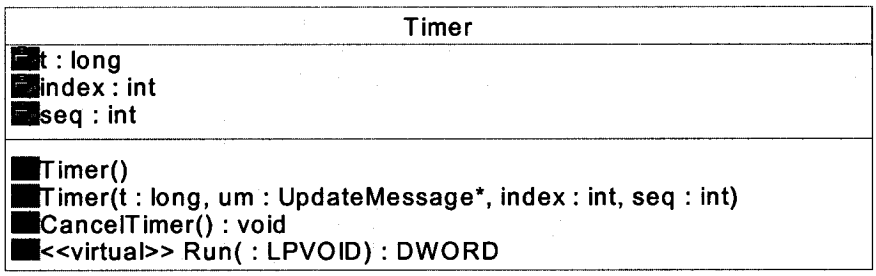

Figure 32: Timer class diagram

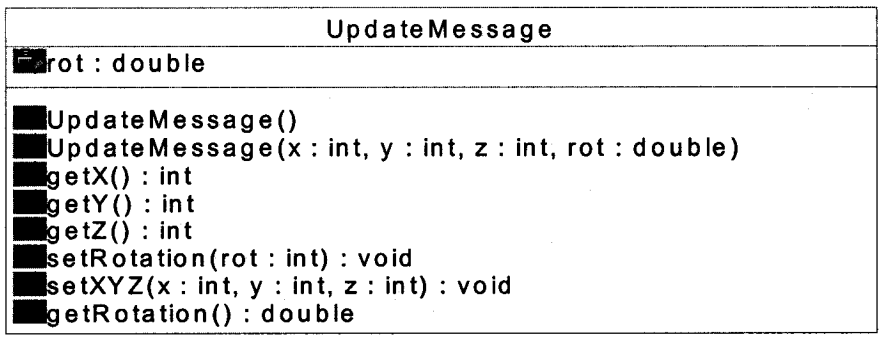

Figure 33: UpdateMessage class diagram

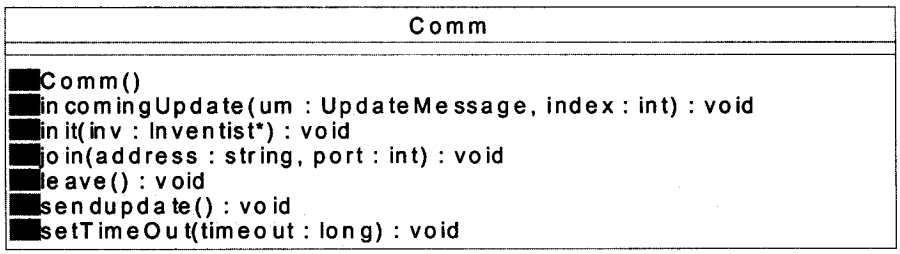

Figure 34: Comm class diagram

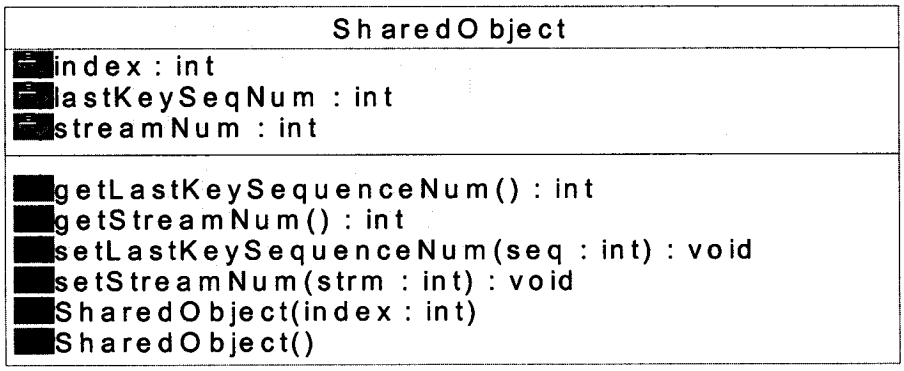

Figure 35: SharedObject class diagram

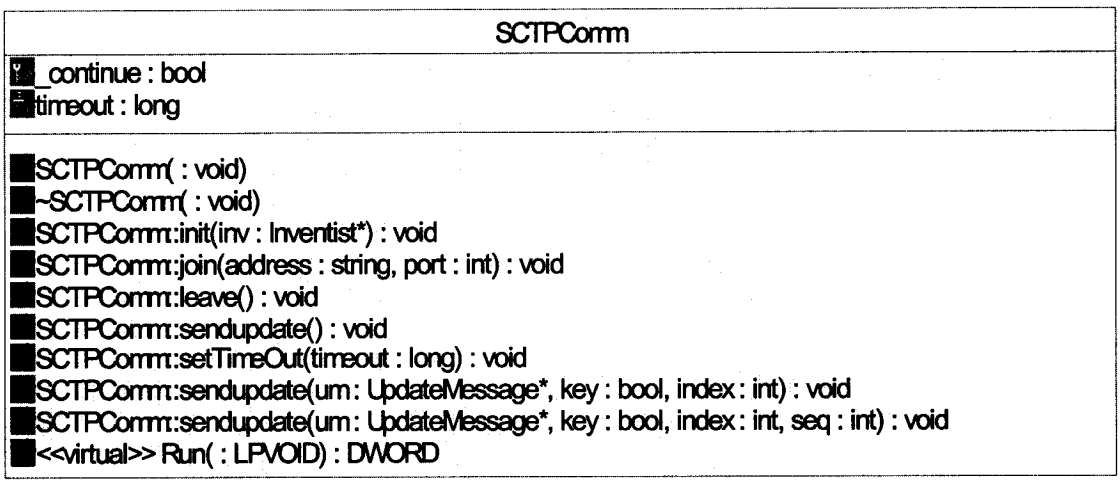

Figure 36: SCTP class diagram

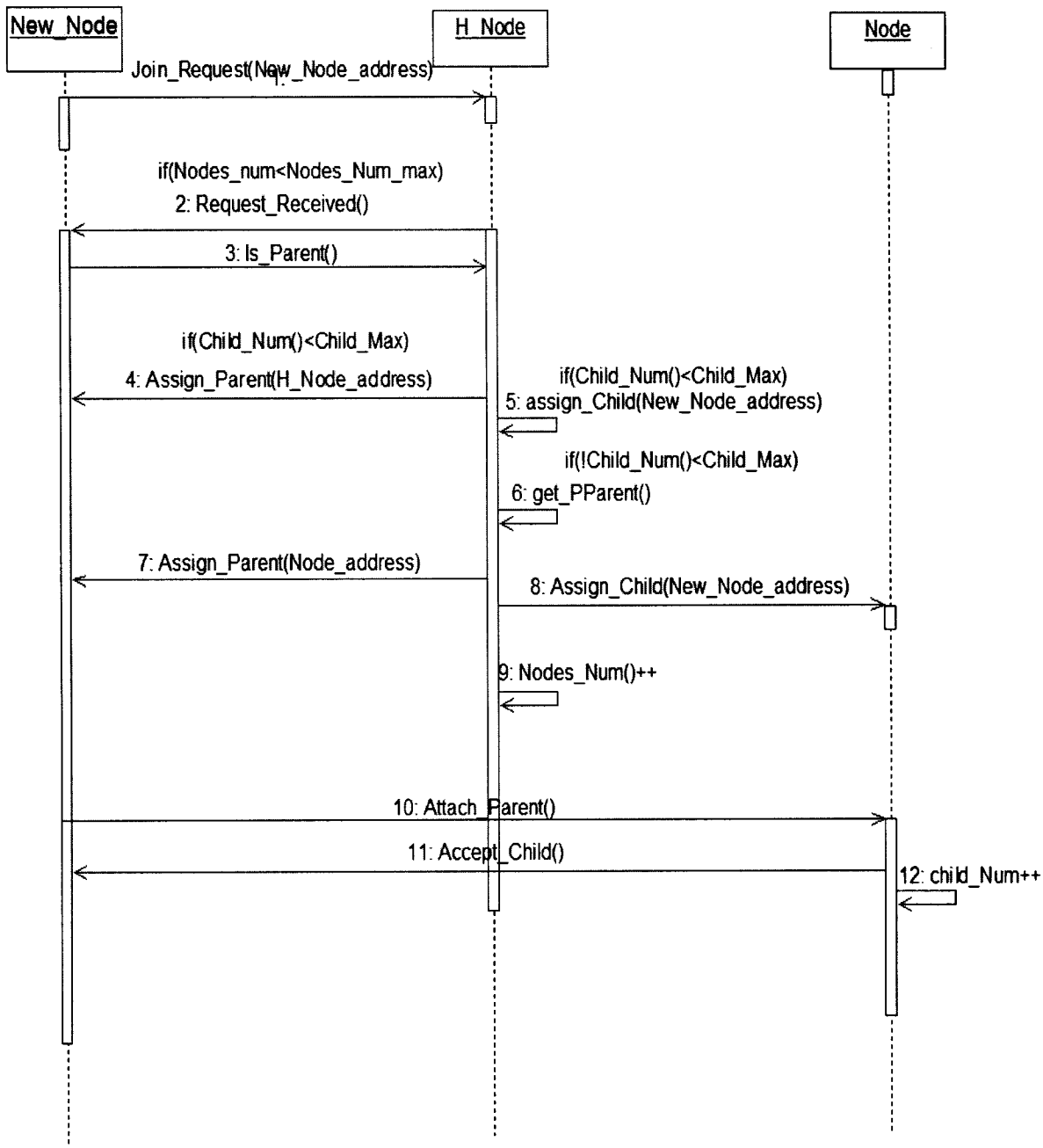

Figure 37: Member-join Sequence diagrams

Chapter 5

Prototype and Performance Measurement

In this chapter the details of the simulation prototype and measures taken to test HDSP protocol are discussed. The testing includes performance as well as a subjective evaluation.

5.1 Simulation Prototype

HDSP is a multi purpose protocol for collaborative virtual environments. To show the usability and strength of HDSP it has been integrated with two multi-player gaming applications. Notice that HDSP framework and those two gaming applications are independent from each other. A coop student [30] has designed these two applications, namely "Panzer Blaster" and "Civilian". While "Panzer Blaster" is a tank simulation application, "Civilian" is a game where users move a sensitive shared object collaboratively. The latter application demonstrates how HDSP handles tightly synchronous tasks in a timely manner. Most of the updates sent and received are key updates. The former application is a simulation of pure multiplayer game environment where both key and non-key messages have the same weighted probability of occurring. Since synchronization between users is critical for a decent experience in real-time distributed applications, HDSP would face an important test in terms of end-to-end delay. We chose OpenSceneGraph (OSG) [34], an object orientated OpenGL graphics library for C++, as our rendering toolkit since it is very well designed and uses a scene graph approach. For "Panzer Blaster" application the scene contains four basic elements: a HUD display, a terrain with vegetation, a sky and a list of users. Each user object contains a tank, a number of bullets and a list of variables needed to create a player.

Both "Panzer Blaster" and "Civilian" simulations run as a separate thread from the HDSP framework. They share global buffers for receiving and sending messages. Those buffers are the middleware of communications between the game application and HDSP framework. Consider "Panzer Blaster" application, when a user wants to connect to a session, a key update message is sent to retrieve a user id from the Hybrid node. This ID is then used to create and identify the user object for that specific player. The messages sent to move and connect a user are encoded bitwise into a string. Movement messages contain XYZ position, direction, speed and angle of the tank's moving parts, while the shoot messages contain the XYZ origin and direction. All these messages also contain the type of message and the user id so each client knows which user's tank to update. These messages are then put in the sending buffer and are specified as key or none-key messages depending on the situation. As depicted in Figure 36 when a bullet is shot, a key message will be triggered and immediately sent to adjacent users.

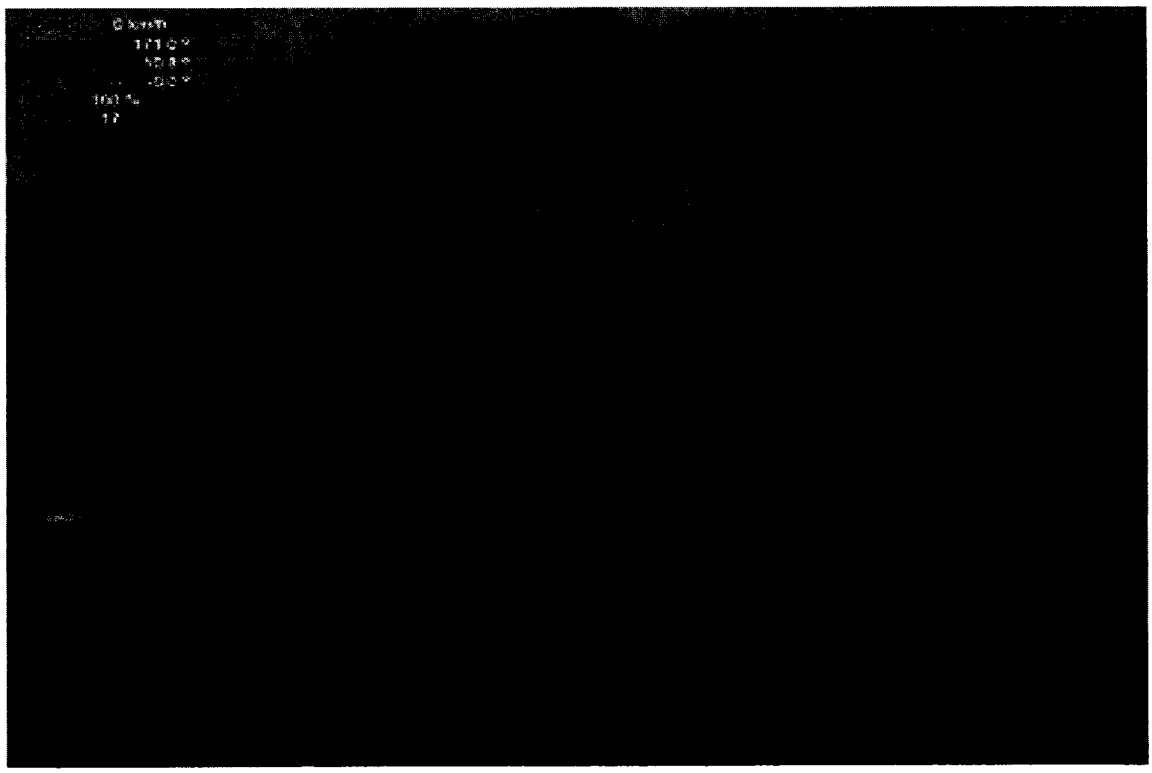

Figure 38: A bullet is shot, this triggers a key message

Figure 37 shows a scenario of movement messages being sent to adjacent users. The receivers in this case will predict the next movement of the tank. The HDSP framework does the rest of the work for sending and relaying the messages. The receiving buffer is looked at for new messages. Messages are decoded and applied to the scene as soon as they are received. Every tank has a controller and a callback node to make it move. The callback nodes are called on a per frame basis during the traversal of the scene graph. The callback node then calls an update method in the controller to update the tank's position if needed. The same principal is applied to move a bullet after its initial position is calculated. When a message to move a tank is received, it is only a matter of assigning its new position, speed and direction and the tank will follow this route until the next message is sent.

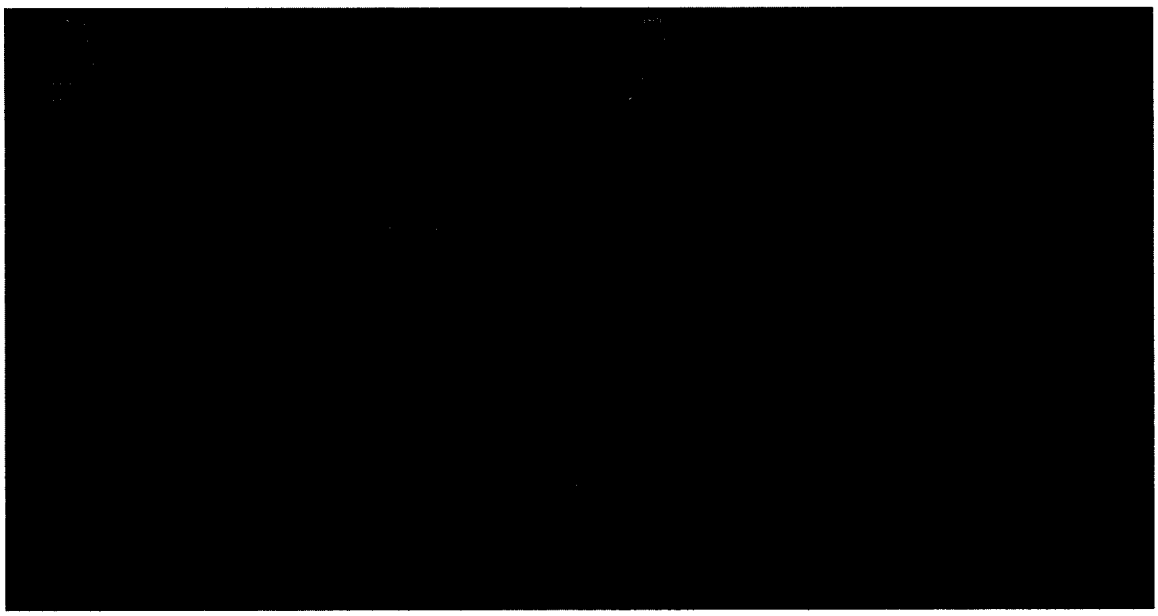

Figure 39: Scenario of movement messages being sent to adjacent users

To predict the tank's movements, the previous message is stored. It is assumed that the tank is either accelerating or stopping in a specific direction. If the message rate is higher than three (3) messages per second then the prediction will be very accurate and is still reasonably good if the rate is one (1) message per second.

Messages are sent to adjacent nodes only when the tank is moving or shooting, eliminating the need for continuous bandwidth usage when objects are idle. During critical events, key messages are sent. Other events are regarded as regular update messages and are simply propagated using UDP. For the developed application, key messages are used when one of the following events happens: stopping, shooting and when moving from a stopped position. These messages are vital to reliably reproduce the movements of the tank on client machines. If these key messages are completely disabled, most of the time the tanks would continue moving forever until the next message is received. Sometimes the tank stops approximately in the right spot but this was due to movement prediction. As expected, some of the shooting messages were not received when using only non-key updates resulting in a conflicting view between players where one player assumes the tank was shot while the other player sees his tank moving.

Figures 38 and Figure 39 are snapshots from the "Civilian" application interface. The shared object was moving due to players' interactions. The rate of messages sent was around 30 messages per second. Notice that in this application, movements cannot be predicted. So, dead reckoning technique was not of great help due to the high frequency of messages and the unpredictability of movements. "Civilian" application was a great scrutiny for HDSP in terms of handling tightly synchronous tasks.

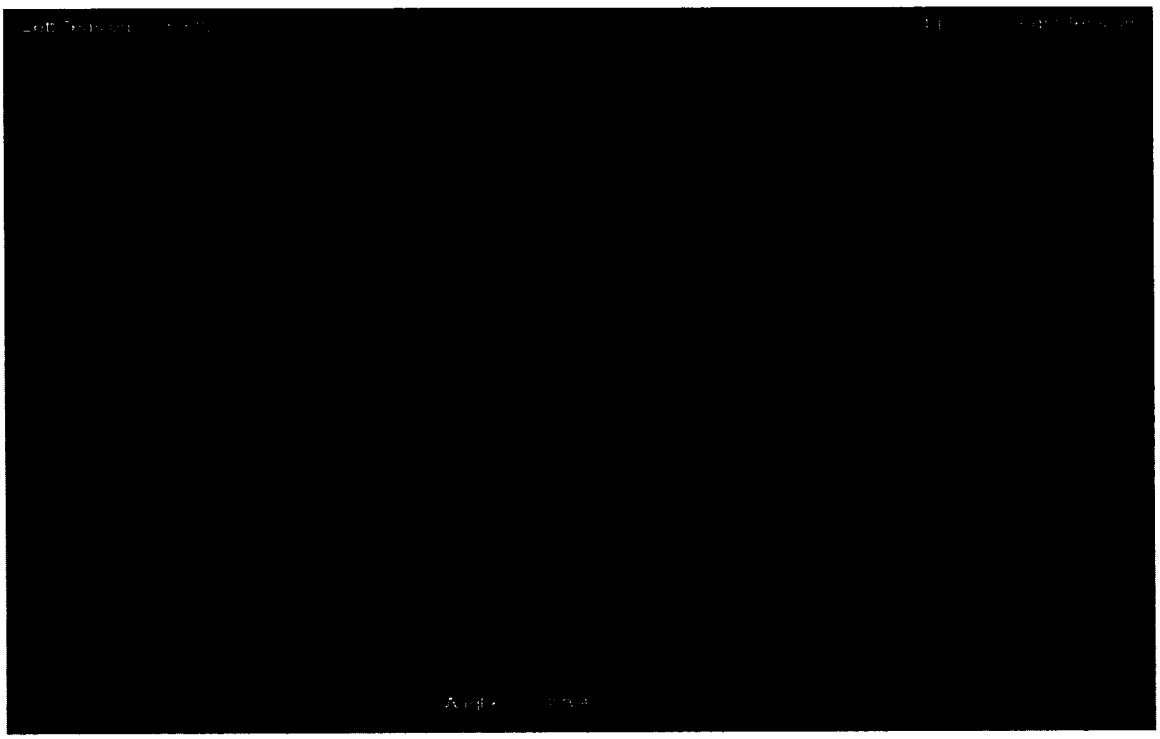

Figure 40: The Civilian application: players are moving a sensitive object

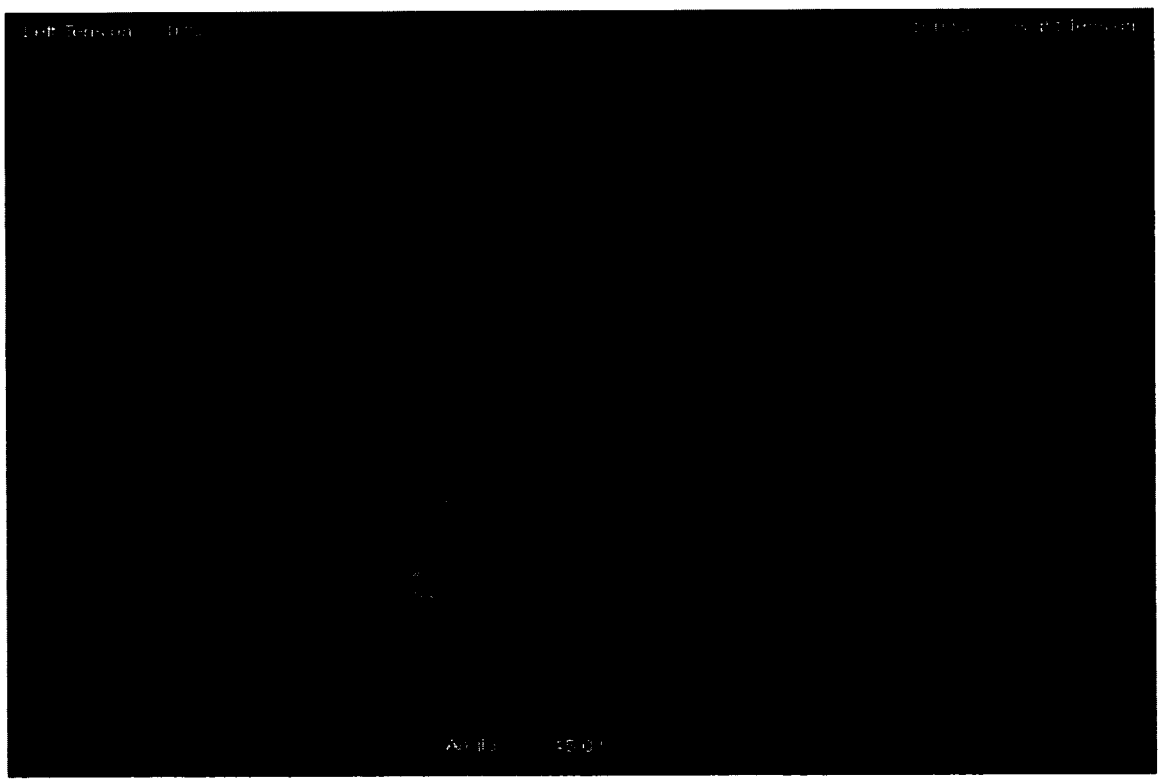

Figure 41: Collaboration failure leads to radioactive material spilling.

5.2 Performance Evaluation

It is a known fact that ALM protocols introduce more delay in data transport than IP multicasting. As it was mentioned earlier, HDSP supports the transmission of short frequent messages. Therefore, to measure the scalability and efficiency of HDSP, a test to measure the latency introduced by the transmission, processing and forwarding of data between peers in the ALM-tree has been conducted.

5.2.1 Delay Measurement

To measure the delay introduced by the traversal of packets across ALM-tree links two important measures are taken into consideration:

1. Round trip time which is the time taken by a data packet to move from one node to another adjacent node. When a packet is sent to an adjacent node an acknowledgement is sent back from the receiving node. The Round trip time is: RTT = (time Ack is received - time packet was sent) /2.

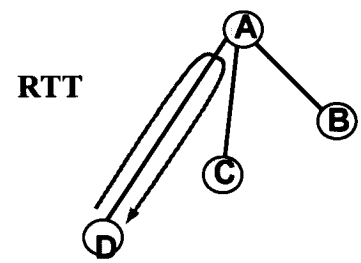

Figure 42: Round trip time

2. The average round trip time in the ALM-tree. In Figure 41, the average delay is equal to (RTT (A to B) + RTT (A to C) + (RTT (A to D))/3.

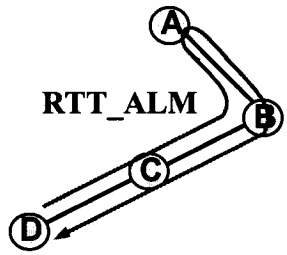

Figure 43: Average round trip time to all nodes

5.2.2 Testing Environment Setup

Performance evaluation was conducted using three PCs with Internet cable connections. Two PCs were using Rogers's connection (cable) in Ottawa and one used Videotron (cable) in Hull-Gatineau. The testing procedure is shown in Figure 42.

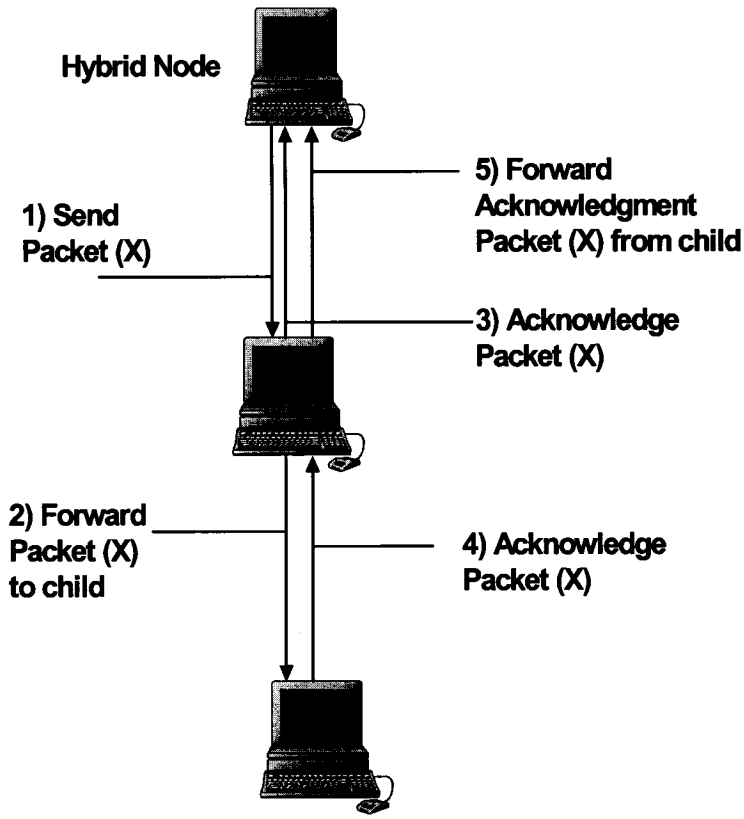

Figure 44: Test procedure for a period of 5 minutes

One of the machines was acting as the hybrid node. The out-degree was set to one, such that a two-level tree is constructed. Continuously, the Hybrid node was sending messages in a rate of one message per second. The receiving node forwards the packet to its children and acknowledges the messages by sending an acknowledgment to the parent. Each parent would forward the acknowledgment message to its parent until the Hybrid node receives acknowledgments from all users. Several trials were conducted ranging from five minutes to seventeen minutes.

The Hybrid node measured the end-to-end delay as the RTT explained above. Table 1 and table 2 represent the average time delay and the packet loss rate for the trials conducted.

	# Packets	Delay Average at Level 1	Delay Average at Level 2	Delay Average for all Nodes in the Tree
Trial #1	300	38.30	130.6	84.45
Trial #2	500	39.20	134.40	86.80
Trial #3	400	40.10	132.80	86.45
Trial #4	300	38.90	135.80	87.35
Trial #5	600	37.81	130.30	84.05
Trial #6	1000	39.20	133.50	86.35

Table 1: Average delay values in milliseconds

	# Packets	#Packets lost	Packet Loss Rate
Trial #1	300	5	0.017 Packet/sec
Trial #2	500	4	0.008 Packet/sec
Trial #3	400	4	0.010 Packet/sec
Trial #4	300	4	0.013 Packet/sec
Trial #5	600	6	0.010 Packet/sec
Trial #6	1000	7	0.007 Packet/sec

Table 2: Number of Packets Lost and loss rate in Each Testing Trial

5.2.3 Analysis

It can be seen from the obtained results that the processing time in the first receiving node before forwarding the data has an impact on the transmission delay. It is deduced that 2-levels ALM-trees are easily supported without violating the 200 ms end-to-end delay

threshold. It is also obvious that the processing time because of different ISPs has an impact on the delay from one node to another. Therefore the following can be hypothesized:

1) In the case where the nodes are under the same ISP, a third level should be easily supported and the expected delay from the Hybrid node to that third level should be about 120-140 ms. A fourth level might also be supported with slight violation of the 200 ms threshold.

2) In the case where the nodes are under different ISPs, HDSP should be able to support the third level with the maximum average delay around the threshold (200 ms) or less. However, a fourth level will likely not be supported.

5.3 Subjective Evaluation

The Subjective evaluation reflects the perceived quality of the game when the players were interacting with each other. Some may evaluate the perceived quality of the game as the quality of experience. As expected, the playing experience of "Panzer Blaster" and "Civilian" on a LAN was very good. Since the delay was minimal, all the messages got sent with constant and minimum latency. As more and more levels were added to the simulation (up to 6 levels), no difference in users' experience was observed.

5.3.1 Testing Trials

The players who participated in this game were organized in an ALM-tree fashion. The ALM-tree structure (the out degree and the number of levels parameters) was adjusted and changed in each trial to test the bandwidth and end-to-end delay capabilities of HDSP. In each trial the nodes of the ALM-tree had the same out degree. During the testing trials, the out degree varied from one to three while the ALM-tree levels varied from one to six levels resulting in eighteen trials.

5.3.2 Monitoring the Game Performance

Besides the players themselves evaluating the game performance, there was an observer in the LAN such that when the player at the root of the tree moves the tank the player at the leaf of the tree was observed to see if he received the message at an acceptable delay time.

5.3.3 Analysis

The playing experience did not change much when it was tested on the Internet, between home users. It was difficult to say whether the end-to-end delay affected the simulation. However, it was very playable and similar to the LAN experience. More warping was noticed on the Internet than on the LAN, but this is attributable to the simulation itself, and among other things, its lack of smoothing when applying the messages.

Chapter 6

Conclusions and Future Work

Collaboration data require special treatment in terms of transmission across the network. While a lot of studies have addressed many transport issues related to distributed simulations in general [1], [14], [15], [19], they fail to fulfill collaboration needs mainly because they do not consider the properties of collaboration data itself.

Distributed simulations in collaborative virtual environments require the transport of update messages between entities participating in the session. Collaboration between entities can be classified as closely coupled and loosely coupled collaboration. The former uses dead-reckoning techniques and assumes that the loss of one update message is compensated by the next [8], while the latter is unpredictable and requires reliable delivery of certain "key" updates [17][23].

This thesis has investigated the existing ALM protocols for distributed simulation. It has proposed a hybrid distributed simulation protocol (HDSP) that is suitable for all types of simulation data and that fulfils the requirements of collaboration data.

6.1 Conclusions

Based on HDSP architecture and based on the performance evaluation test, the following conclusions about HDSP can be drawn:

- HDSP provides a number of communication services:
  1) Best Effort LAN Multicast
  2) Timely-reliable LAN Multicast
  3) Best Effort Peer-to-Peer delivery on the Internet

4) Timely-reliable Peer-to-Peer delivery on the Internet
5) Any combination of the above, including LAN to Internet translation and vice versa.

- HDSP provide guaranteed reliability for key updates. For this, HDSP incorporates SCTP protocol to provide reliability for key updates with the minimum bandwidth usage, i.e. acknowledging only key update messages.
- HDSP architecture supports efficient peer-to-peer communication through organizing end-systems in an overlay multicast tree.
- HDSP is a multi-source ALM protocol where each end-system can send and receive data at the same time. Hence, HDSP supports multi-player online games, multi-party video streaming and so on.
- The hybrid node acts as a mediator and a maintainer of the ALM-tree and also restructures the tree in case of a failure happening to one of the end-systems.
- Performance results demonstrate how application layer multicasting can be used in conjunction with sender- initiated reliable communication to support large-scale networked games on the Internet. Performance evaluation results showed that 4 levels of users could be supported in our architecture, with many users being able to connect at each level.
- Subjective evaluation shows that HDSP can support up to 6 level ALM tree without noticeable deterioration in the perceived quality because dead reckoning algorithm was implemented in the graphics application

Other than the related IP-multicast based protocols mentioned above, Mauve [23] presents an architecture that uses proxies to provide fairness between players, low latency, congestion control and robustness. However, the work presented in this thesis differs in a sense that HDSP categorize the messages as regular or key update messages which is an essential part in providing reliability for important messages while reducing network congestion by acknowledging only key messages. This work also differs in a sense that the Hybrid node has the tree structure, regulates the growth of the tree by assigning new children to their prospective parents and rearranges the tree in the case of nodes leaving the tree or crashing.

6.2 Future Work

While HDSP demonstrated robustness and scalability in terms of supporting a large number of users, there is still a room for improvement, which is beyond the scope of this thesis. In the following section some of the functionalities that are under continuous research in the discover lab are briefly mentioned.

6.2.1 Area of Interest Management

As the number of users grows in the session, efficient handling of resources and scalable communication between users becomes critical, due to network traffic and computation required to track all objects. It is of great importance to implement efficient mechanisms to handle the session without compromising the delay and scalability requirements of the CVEs, and the feeling of immersion the users experience. The solution typically used is to partition the virtual simulation into a number of areas, and to manage each area separately. This is known as Area of Interest Management (AoIM).

6.2.2 Fault Tolerance

HDSP is fault tolerant, in a sense that if a node crashes, it handles the reorganization of the ALM-tree. For future research, it is suggested that HDSP should be placed under severe test to prove it can handle many simultaneous crashes without disturbing the session state. Moreover, HDSP should also allow restructuring the ALM-tree periodically. This function will be useful since the network traffic varies through time and links that were not congested might be congested after words. Moreover, if the hybrid node fails, then another node from the LAN should replace it and become a hybrid node. This could be achieved if the original hybrid node selects a potential node to be the future hybrid one and transmits the tree structure periodically to this node.

6.2.3 Security Issues

One of the key issues in ALM is security. In the current HDSP design, it is assumed that users do not have access to modify the data sent and received. However, this will work

for certain applications where authenticity of source is assumed. For critical applications where security is of high concern, HDSP must be modified to handle techniques, such as data encryption and decryption, digital signatures, and authentication.

Appendix A – ALM Protocols Classification

ALM Protocol	Application Domain	Deployment Level	Metric	Centralized or distributed	Exploit IP Multicast	Refinement
ALMI	2	End-system	Delay	Centralized	No	Yes
Amcast	1	Proxy-based	Delay Bandwidth	Centralized	No	No
Bayeux	1	Proxy-based	Delay	Distributed	Yes	Yes
Borg	3	Proxy-based	Delay Bandwidth	Distributed	No	No
CAN Multicast	3	Proxy-based	Delay	Distributed	No	No
CoopNet	1	End-system	Delay Bandwidth	Centralized	No	No
Delaunay	3	Proxy-based	Geographical position	Distributed	No	No
Gossamer	3	Proxy-based	Delay Bandwidth	Distributed	Yes	Yes
HBM	2	Combined	Delay	Centralized	No	Yes

HMTP	1	Proxy-based	Delay	Distributed	Yes	Yes
Narada	2	End-system	Delay Bandwidth	Distributed	No	Yes
NICE	1	End-system	Delay Bandwidth	Distributed	No	Yes
OMNI	1	Proxy-based	Delay Bandwidth	Distributed	Yes	Yes
OverCast	4	Proxy-based	Bandwidth	Distributed	No	No
RITA	1	Proxy-based	Delay Bandwidth	Distributed	No	Yes
RMX	4	Proxy-based	Delay Bandwidth	Distributed	Yes	Yes
Scribe	3	Proxy-based	Delay Bandwidth	Distributed	No	Yes
SpreadIt	1	End-system	Bandwidth Delay	Distributed	No	No
TAG	1	End-system	Topology (delay) Bandwidth	Distributed	No	No
TBCP	3	End-system	Delay Bandwidth	Distributed	No	No
Yoid [37]	3	End-system	Bandwidth Delay	Distributed	Yes	Yes

ZIGZAG	1	End-system	Delay Bandwidth	Distributed	No	Yes

Table 3: Brief survey of existing ALM protocols

References

[1] B. Blau, C. Hughes, M. Michael, and L. Curtis, in ACM SIGGRAPH, Symposium on 3D Interactive Graphics, March 1992, pp. 157--160. Blau et al, "Networked Virtual Environments", Proc. ACM SGGRAPH, 1992, pp. 157-164.

[2] Pullen, J., Mytak, M. and C. Bouwens, "Limitations of Internet Protocol Suite for Distributed Simulation in the Large Multicast Environment", RFC 2502, February 1999.

[3] M.R. Macedonia, and M.J. Zyda, "A Taxonomy for Networked Virtual Environments", IEEE Multimedia Magazine, January-March 1997, pp. 48-56.

[4] S. Seidensticker, "Scenarios and Appropriate Protocols for Distributed Interactive Simulation", Internet draft, draft-myjak-lsma-scenarios-02, March 1997.

[5] S. Shirmohammadi and N.D. Georganas, "Collaborating in 3D Virtual Environments: A Synchronous Architecture", Proc. IEEE International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, IEEE WETICE 2000, U.S.A., June 2000, pp. 35-42.

[6] T. Funkhouser, "Network Topologies for Scalable Multi-User Virtual Environments", Proc. IEEE Virtual Reality Annual International Symposium, 1996, pp. 222-228.

[7] S. Shirmohammadi and N.D. Georganas, "An Architecture for Collaboration in Virtual Environments", Proc. IEEE Virtual Reality (VR 2000), p.283.

[8] S. Shirmohammadi and N.D. Georganas, "An End-to-End Communication Architecture for Collaborative Virtual Environments", Computer Networks Journal, Vol.35, No.2-3, Feb. 2001, pp.351-367.

[9] R. M. Fujimoto, "Parallel and Distributed Simulation," Winter Simulation Conference, pp. 122-131 (invited tutorial).

[10] Shervin Shirmohammdi, "Synchronous Collaboration in Virtual Environments: Architecture, Design, and Implementation", Ph.D. thesis, University of Ottawa, Ottawa, Canada, 2000.

[11] http://en.wikipedia.org/wiki/Distributed_Interactive_Simulation, last visited on April 20, 2006.

[12] Olof Hagsand, "Interactive Multi-user VEs in the DIVE System", IEEE Multimedia, pp. 30-39, Spring 1996

[13] High Level Architecture (HLA), https://www.dmso.mil/public/transition/hla/, last visited on April 20, 2006.

[14] S. Shirmohammadi, N.D. Georganas, "An End-to-End Communication Architecture for Collaborative Virtual Environments", Computer Networks Journal, Vol. 35, No. 2, pp. 351-367, February 2001

[15] D. DeLucia, K. Obraczka, "A Multicast Congestion Control Mechanism for Reliable Multicast," iscc, p. 142, Third IEEE Symposium on Computers & Communications, 1998.

[16] R.M. Fujimoto, Parallel and Distributed Simulation Systems, Wiley, 2000.

[17] J. M. Pullen, "Reliable Multicast Network Transport for Distributed Virtual Simulation," presented at Proceedings of the 3rd IEEE International Workshop on Distributed Simulation and RealTime Applications, October 23-24, College Park, Maryland, 1999.

[18] IEEE Standard for Distributed Interactive Simulation, Application Protocols, IEEE 1278-1995.

[19] A. El-Sayed and V. Roca, "A Survey of Proposals for an Alternative Group Communication Service," IEEE Network, 17(1), pp. 46-51, Feb. 2003.

[20] B. Zhang, S. Jamin, and L. Zhang, "Host multicast: A framework for delivering multicast to end users," In Proc IEEE INFOCOM, June 2002.

[21] Y. Chu, S.G. Rao, S. Seshan, and H.S. Zhang, "A Case for End System Multicast", IEEE Journal on Selected Areas in Communication, special issue on networking support for multicast, 2002, Volume 20, Issue 8, pp. 1456-1471.

[22] Diot, C., Levine, B.N., Lyles, B., Kassem, H., & Balensiefen, D. (2000). Deployment issues for the IP multicast service and architecture. IEEE Network Magazine, 14(1), 78-88.

[23] M. Mauve, S. Fischer, and J Widmer, "A generic proxy system for networked computer games", Proc. of NetGames 2002, pp. 25-28, Braunschweig, April 2002.

[24] Knut-Helge Vik, "Game state and event distribution using proxy technology and application layer multicast", Proc. ACM Multimedia, Doctoral Symposium, Singapore, Nov. 6-11 2005, pp. 1041-1042.

[25] Wierzbicki A., Szczepaniak R., Buszka M., "Application Layer Multicast for Efficient Peer-to-Peer Application", Proceedings of 2003 IEEE Workshop on Internet Applications (WIAPP'03), pp. 126-130.

[26] Samadian-Barzoki, S. Bag-Mohammadi, M. Yazdani, N., "BMP: an efficient and scalable multicast protocol", Proceedings of 2004 Canadian Conference on Electrical and Computer Engineering, Volume 4, pp. 1993- 1996.

[27] Banerjee, S., Kommareddy, C., Kar, K., Bhattacharjee, B., Khuller, S., "Construction of an Efficient Overlay Multicast Infrastructure for Real-time Applications", Proceedings of 2004 IEEE Conference on Computer and Communications (INFOCOM'04), Volume 2, pp. 1521-1531.

[28] Brosh, E., Shavitt, Y., "Approximation and Heuristic Algorithms for Minimum Delay Application-Layer Multicast Trees", Proceedings of 2004 IEEE Conference on Computer and Communications (INFOCOM'04), Volume 4, pp. 2697-2707.

[29] Yong Zhong, "Proxy-Based Single Source Application Layer Multicast Media Streaming", Master's Thesis, University of Ottawa, Ottawa, Canada, year 2005.

[30] Pascal Lacombe, University of Ottawa, Ottawa, Canada, Site co-op report, year 2005.

[31] Sandeep Singhal, and Michael Zyda, Networked Virtual Environments, ACM Press, NY, NY, 1999, p. 143.

[32] Y. Chu, S.G. Rao, S. Seshan, and H.S. Zhang, "A Case for End System Multicast", IEEE J. on Selected Areas in Communication 2002, Volume 20, Issue 8, pp. 1456-1471.

[33] Mojtaba Hosseini, "An End System Multicast Protocol for Multi-sender 3D Video conferencing Applications", Ph.D. thesis, School of Information Technology and Engineering, University of Ottawa, Ottawa, Canada, October 2004.

[34] OpenSceneGraph Open source cross platform graphics toolkit, available at: http://www.openscenegraph.org/, last visited on April 20, 2006.

[35] http://www.answers.com/multicast-backbone, last visited on April 20, 2006.

[36] http://en.wikipedia.org/wiki/Peer-to-peer, last visited on April 20, 2006.

[37] P. Francis. Yoid: Extending the multicast Internet architecture, 1999. On-line documentation: http://www.aciri.org/yoid/.