1-1-1996

Internet Protocol (IP) Multicast: Final Report

Michael D. Myjak

Nicole L. Densmore

Suresh Sureshchandran

Recommended Citation

Myjak, Michael D.; Densmore, Nicole L.; and Sureshchandran, Suresh, "Internet Protocol (IP) Multicast: Final Report" (1996). *Institute for Simulation and Training*. 126.
https://stars.library.ucf.edu/istlibrary/126

INSTITUTE FOR SIMULATION AND TRAINING

Contract Number N61339-94-C-0024
CDRL A00H
STRICOM
February 21, 1996

Internet Protocol (IP) Multicast

Final Report

Institute for Simulation and Training
3280 Progress Drive
Orlando FL 32826

University of Central Florida
Division of Sponsored Research

INSTITUTE FOR SIMULATION & TRAINING

IST-TR-96-06

B309

INSTITUTE FOR SIMULATION AND TRAINING

Internet Protocol (IP) Multicast

Final Report

IST-TR-96-08
February 21, 1996

Prepared For:
STRICOM
N61339-94-C-0024
CDRL A00H

Prepared By:
Michael Myjak
Nicole L. Densmore

[illegible]

Reviewed By:
Suresh Sureshchandran

[illegible]

Institute for Simulation and Training • 3280 Progress Drive • Orlando, Florida 32826
University of Central Florida • Division of Sponsored Research

TABLE OF CONTENTS

1. INTRODUCTION	5
1.1 Networking Background	6
1.1.1 Request for Comments	7
1.1.2 RFC States for Internet Protocols	8
1.1.3 Internet Protocol Addressing	8
1.1.4 Networks and Hosts and Subnets	9
1.1.5 Internet Protocol Multicast	10
1.1.6 Ethernet	10
1.1.7 Ethernet Multicast Addressing	11
1.1.8 Multicast Addressing Summary	11
1.2 The TRIDIS Project	12
1.2.1 Multicast Conformance	12
1.2.2 Transport Layer - IP Transmissions	12
2. RESEARCH GOALS	15
2.1 Routing	16
2.2 Resource Reservation	16
2.3 Filtering	17
2.3.1 Internal Filtering	18
2.3.2 External Filtering	18
2.3.3 Improved Application Efficiency	20
3.0 APPROACH	20
3.1.1 Repair the Standard	21
3.1.2 Survey Protocol Products	21
3.1.3 Select and Purchase a Product	22
3.1.4 Integrate New Protocols into Testbed	23
3.1.5 Conduct Initial Tests	24
3.1.6 Evaluate Hardware Requirements	25
3.1.7 Port to IRIX	25
3.1.8 Equipment Setup	26
4. EXPERIMENTS	27
4.1 Theory of Operation	27
4.1.1 Segment #1	27
4.1.2 Segment #2	27
4.1.3 Segment #3	28
4.2 Experiment #1	28
4.3 Experiment #2	30
4.4 Experiment #3	32
4.5 Experiment #4	34
4.6 Experiment #5	36

5. RESULTS 38

5.1 Results of Experiment #1 38

5.1.1 Layout 39

5.1.2 The Analysis 40

5.1.3 Conclusion 43

5.2 Results of Experiment #2 43

5.2.1 Layout 43

5.2.2 The Analysis 45

5.2.3 Conclusion 46

5.3 Results of Experiment #3 46

5.3.1 Layout 47

5.3.2 The Analysis 47

5.3.3 Conclusion 49

5.4 Results of Experiment #4 49

5.4.1 Layout 49

5.4.2 The Analysis 51

5.4.3 Conclusion 53

5.5 Results of Experiment #5 54

5.5.1 Layout 54

5.5.2 The Analysis 54

5.5.3 Conclusion 58

5.6 Experiment Timing Comparisons 58

6. CONCLUSIONS 64

7. RECOMMENDATIONS 66

8. REFERENCE 67

LIST OF FIGURES

FIGURE 1 - ARCHITECTURAL LAYOUT FOR THE TRIDIS IPMC EXPERIMENTS. 26
FIGURE 2 - PHYSICAL CONFIGURATION FOR TRIDIS IPMC EXPERIMENT #1. 29
FIGURE 3 - LOGICAL CONFIGURATION FOR TRIDIS IPMC EXPERIMENT #1. 30
FIGURE 4 - CONFIGURATION FOR EXPERIMENT #2 32
FIGURE 5 - CONFIGURATION OF EXPERIMENT #3. 33
FIGURE 6 - CONFIGURATION OF EXPERIMENT #4 35
FIGURE 7 - CONFIGURATION OF EXPERIMENT #5 36
FIGURE 8 - PDU TRANSMIT AND RECEIVE COUNTS FROM EXPERIMENT #1. 40
FIGURE 9 - PDU'S TRANSMITTED AND RECEIVED IN EXPERIMENT #1. 42
FIGURE 10 - PDU TRANSMIT AND RECEIVE COUNTS FROM EXPERIMENT #2. 44
FIGURE 11 - PDU'S TRANSMITTED AND RECEIVED IN EXPERIMENT #2. 46
FIGURE 12 - PDU TRANSMIT AND RECEIVE COUNTS FROM EXPERIMENT #3. 47
FIGURE 13 - PDU'S TRANSMITTED AND RECEIVED IN EXPERIMENT #3. 48
FIGURE 14 - PDU TRANSMIT AND RECEIVE COUNTS FROM EXPERIMENT #4. 50
FIGURE 15 - PDU'S TRANSMITTED AND RECEIVED IN EXPERIMENT #4. 51
FIGURE 16 - PDU'S TRANSMITTED AND PROCESSED IN EXPERIMENT #4. 53
FIGURE 17 - PDU TRANSMIT AND RECEIVE COUNTS FROM EXPERIMENT #5 55
FIGURE 18 - PDU'S TRANSMITTED AND RECEIVED IN EXPERIMENT #5. 56
FIGURE 19 - PDU'S TRANSMITTED AND PROCESSED IN EXPERIMENT #5. 57
FIGURE 20 - DECODE TIME COMPARISONS OF EXPERIMENTS #2 AND #4 FOR SUN-1. 60
FIGURE 21 - DECODE TIME COMPARISONS OF EXPERIMENTS #2 AND #4 FOR SUN-2. 60
FIGURE 22 - DECODE TIME COMPARISONS OF EXPERIMENTS #2 AND #4 FOR SGI-1. 61
FIGURE 23 - DECODE TIME COMPARISONS OF EXPERIMENTS #2 AND #4 FOR SGI-2. 61
FIGURE 24 - DECODE TIME COMPARISONS OF EXPERIMENTS #3 AND #5 FOR SUN-1. 62
FIGURE 25 - DECODE TIME COMPARISONS OF EXPERIMENTS #3 AND #5 FOR SUN-2. 62
FIGURE 26 - DECODE TIME COMPARISONS OF EXPERIMENTS #3 AND #5 FOR SGI-1. 63
FIGURE 27 - DECODE TIME COMPARISONS OF EXPERIMENTS #3 AND #5 FOR SGI-2. 63

1. INTRODUCTION

This report is a deliverable item, CDRL A00H, under TRIDIS subtask 3.2.3.4, "Adding Other Protocols," of the U.S. Army Simulation Training and Instrumentation Command (STRICOM) contract #N61339-94-C-0024, entitled "TRIDIS: A Testbed for Research in Distributed Interactive Simulation."

This report presents the results of the Institute for Simulation and Training's (IST) implementation and integration of new protocols into the TRIDIS Testbed. This work focused on incorporating the Internet Protocol (IP) Multicast (IPmc) and Internet Group Management Protocol (IGMP) network protocols into a simulation application. The simulation application chosen by the team was the IST Computer Generated Forces (CGF) Testbed. The IST CGF was used because it was easy to work with and provided a flexible simulation application from which to investigate Internet Protocol Multicast communications.

This report describes:

- The background and approach taken to implement IPmc in the Testbed.
- The goals of this research.
- The approach taken in the selection of an available simulation application for testing.
- The experiments designed to test the Internet Protocol Multicast implementation.
- The results, analysis and conclusions from the TRIDIS IPmc experiments.

Abbreviations and Acronyms

The following abbreviations and acronyms are used in the body of this paper:

ACK	Acknowledgment
ARP	Address Resolution Protocol
CADIS	Communications Architecture for DIS
CSMA/CD	Carrier Sense Multiple Access/Collision Detection
CGF	Computer Generated Forces
COTS	Commercial-Off-The-Shelf
CPU	Central Processing Unit
DIS	Distributed Interactive Simulation
DSI	Defense Simulation Internet
DVMRP	Distance Vector Multicast Routing Protocol
GPS	Global Positioning System
IGMP	Internet Group Management Protocol
IP	Internet Protocol
IPC	Inter-process Communication
IPmc	Internet Protocol Multicast
IST	Institute for Simulation and Training
MOSPF	Multicast Open Shortest Path First
NIS	Network Information Service(s)
NTP	Network Time Protocol
OSPF	Open Shortest Path First
PC	Personal Computer
PDU	Protocol Data Unit
RAM	Random Access Memory
RFC	Request For Comments
ST2	Internet Stream Protocol Version 2
STRICOM	Simulation Training and Instrumentation Command
TCP	Transmission Control Protocol
TDB	Terrain Database
TRIDIS	Testbed for Research in DIS
TSR	Terminate and Stay Resident
UDP	User Datagram Protocol

1.1 Networking Background

"It has been said that the principal function of an operating system is to define a number of different names for the same object, so that it can busy itself keeping track of the relationship between all of the different names. Network protocols seem to have somewhat the same characteristic."

David D. Clark, 1982
MIT Laboratory for Computer Science

1.1.1 Request for Comments

The documents called Request for Comments (or RFCs) are the working notes of the "Network Working Group," that is the Internet research and development community. A document in this series may be on essentially any topic related to computer communications, from a meeting report to the specification of an internet protocol standard. Note that not all standards are published as RFCs, and not all RFCs specify standards. [Postel, 94]

RFC submissions that are intended to document a proposed standard, draft standard, or standard protocol will only be published with the approval of the Internet Engineering Steering Group (IESG). Generally, before publication the IESG will notify the Internet Engineering Task Force, allowing for the possibility of review by the relevant IETF working group, and provide those comments to the author before publication. For more detail on the development of internet standards, please see RFC 1720 entitled "Internet Official Protocol Standards," [Postel, 94].

Once an RFC document is submitted and "... assigned an RFC number and published, that RFC is never revised or re-issued with the same number." [Postal, 94] There should never be a question of having the most recent version of a particular RFC. However, a protocol previously described in an RFC (such as the File Transfer Protocol (FTP)) may be improved upon and re-documented in a different RFC. It is therefore important to verify that you have the most recent RFC on a particular protocol.

The RFC document series comprises a wide range of documents besides specifications of standard Internet protocols. Several of these protocols are referenced in the Protocol Standard for Distributed Interactive Simulation [IEEE-1278.2] and are referred to throughout the body of this document. RFCs are only published electronically and can be accessed by anonymous FTP from `FTP://ds.internic.net/rfc`.

1.1.2 RFC States for Internet Protocols

There are two independent categorizations for all RFCs describing the Internet suite of protocols. The first categorization refers to the maturity level or *state* of standardization. A protocol state may be either "standard," "draft standard," "proposed standard," "experimental," "informational" or "historic." Only those RFCs with a state of "proposed standard," "draft standard," or "standard" are considered to be *standards track* protocols. The second categorization is the requirement level or *Status* of a protocol and may be either "required," "recommended," "elective," "limited use," or "not recommended." The status and requirement level are difficult to portray in a one word label, therefore, the status labels should be considered only as an indication, and a further description or applicability statement should be also be consulted. This information is usually contained within the RFC header. When a standards-track protocol is advanced to proposed standard or draft standard, it is labeled with a status.

1.1.3 Internet Protocol Addressing

In the Internet Protocol (IP) suite, the Transmission Control Protocol (TCP or sometimes TCP/IP) provides several ways of referring to *things*. At the human visible interface, there are character string "names" that are used to identify networks, hosts, and services. Using the appropriate service, host names are translated into network "addresses" and services are translated into port numbers. A network address consists of a 32-bit value that identifies the network to which a host is attached and the location of the host on that network. Table 1 displays the addressing mode for 32-bit internet protocol addresses. Service names are translated into a "port identifier," which in TCP is defined as a 16-bit value [Clark, 82]. Port identifiers are not germane to this discussion and so will not be discussed further.

Most Significant	*Bit Format*	*Class*
0	*+ 7 bits of net, 24 bits of host*	*A*
10	*+ 14 bits of net, 16 bits of host*	*B*
110	*+ 21 bits of net, 8 bits of host*	*C*
1110	*+ 28 bits of Multicast host groups*	*D*
1111	*Reserved for future addressing modes*	*E*

Table 1 Internet Protocol address formats by class.

As shown in Table 1, networks are partitioned into various classes based upon the number of hosts they may contain. A Class-A network for example, is a network that is defined as having the most significant bit as "0." A Class-B network is defined as having the most significant 2-bits as "10." Similarly, Class-C network addresses begin with the three most significant bits being "110."

In a Class-A network, the first 8-bits (beginning with "0" for the most significant bit) are used to identify the network. The remaining 24 bits are used to identify the hosts within that network. Similarly, in a Class-B network, the first 2-bits identify the network class while the next 14-bits identify the network. The remaining 16-bits identify the hosts on a Class-B network. Similarly, Class-C networks use the first 3-bits to identify the network class while the next 21-bits identify the network. The remaining 8-bits identify hosts within a Class-C network.

1.1.4 Networks and Hosts and Subnets

The two-level hierarchy of IP addressing is further extended to three levels with the notion of subnetting in RFC 950. Subnets of an Internet are logically visible sub-sections of a single Internet network [Mogul, 85]. For administrative or technical reasons, many organizations have chosen to divide one Internet network into several subnets, instead of acquiring a set of Internet network numbers. RFC 950 specifies the approved procedures for the use of subnets within the Internet community. Important motivation and background information for a subnetting standard are provided in RFC 940.

A network number corresponds to an organization and may encompass many interconnected LANs. Typically these LANs are divided into subnets and are expressed in terms of subnet number and hosts. The structure of a network address stays the same; only the interpretation of the number is different. Therefore a network address consists of a network class and number, subnet number and a host ID. These fields are used to uniquely identify every host in an Internet Protocol network. Note that the overall size of the subnet+host field is of a fixed length for a given network class. Although the overall size is fixed, the actual division point between subnet and host identification field is flexible and can vary from network to network or from LAN to LAN within a network.

A host can determine which host addresses are located on the local subnet by using a host subnet bitmask. The bitmask determines which bits of a network address are from the network and which bits are from the subnet. By masking away the network and subnet numbers, the remaining bits will uniquely identify a local host.

The subnet broadcast capability, defined as an address containing all ones in the host ID field, is used to deliver datagrams to all hosts directly connected to a given subnet [Mogul, 84]. Broadcasting provides a good base for many applications. Broadcast communications relies on the best-efforts of the UDP and therefore is inherently unreliable. Broadcast messages are also unsequenced and can possibly generate duplicated datagrams. Even though unreliable and limited in length, datagram broadcasts are quite useful within a LAN environment. [Boggs, 82]. Further information about the nature of the internet, its addressing modes and protocol descriptions can be found in the RFC library, published electronically by the Internet Advisory Board.

1.1.5 Internet Protocol Multicast

IPmc addressing describes the transmission of an IP datagram to a "host group" represented by a Class-D IP address. A datagram refers to the data payload that is being communicated from one host to another. A host group is defined as a set of zero or more hosts identified by a single address [Deering, 89]. Multicast addresses correspond to a destination host group to which a multicast datagram will be delivered [RFC 1112]. Membership in the host group is dynamic. Dynamic membership means that hosts may join and leave IPmc groups at any time by informing their neighboring multicast router. Multicast routers keep track of the networks to which they are connected that have members of a destination group.

IPmc addressing differs from the standard IP unicast addressing in two important ways. First, when using IPmc addresses, multiple hosts are being represented by a single Class-D internet address, as defined by the Internet Group Management Protocol (IGMP) in RFC-1112 (a.k.a. Internet Standard #5). Second, an IPmc host group may be either permanent or transient, whereas IP unicast addresses are static. A permanent host group has a well known, administratively assigned address. Those Class-D addresses that are not reserved for permanent groups are available for dynamic assignment to transient groups that exist only as long as they have members.[1]

1.1.6 Ethernet

The original specification for Ethernet was developed by the Xerox corporation and is one of the most popular network cabling schemes in use today. Ethernet is described as a hardware and data link layer specification that uses Carrier Sense Multiple Access/Collision Detection (CSMA/CD) which is a fancy way of saying that only one machine connected to the Ethernet can transmit a *frame* of data at any one time. If two or more machines attempt to use the same network at precisely the time, *collisions* of Ethernet frames will occur. When collisions occur, the two machines will terminate their transmissions and attempt to retransmit their data frames after waiting some random time interval.

A newer version of Ethernet called Ethernet II was developed by Xerox in conjunction with Digital Equipment Corporation and Intel. The Institute of Electrical and Electronics Engineers (IEEE) standardized a form of Ethernet II in IEEE standard: 802.3. Today the

[1] *The addresses from 224.252.0.0 through 224.254.255.255 are reserved for virtual environments.*

term Ethernet is synonymous with IEEE 802.3 and Ethernet II specifications; both frame structures are compatible with the same hardware and can work within the same network. Note that both the IEEE 802.3 and Ethernet II standards specify a hardware protocol.

1.1.7 Ethernet Multicast Addressing

When an IP datagram is transmitted on an Ethernet LAN, the 32-bit IP destination address must be mapped to a 48-bit Ethernet address [Hornig, 84]. Mappings between IP addresses and Ethernet addresses are accomplished through the Address Resolution Protocol (ARP) [Plummer, 82].

Using IGMP, IPmc destination addresses are identified as Class-D IP addresses, i.e., those networks with "1110" as their high order four bits. Note that IPmc addresses do not include Class-E IP addresses, i.e., those with "1111" as their high-order four bits, as these are reserved for future addressing modes. In a Class-D address, 28-bits are used to identify the multicast host group address.

Ethernet directly supports sending and receiving local multicast datagrams by allowing multicast addresses in the Ethernet destination address field. A sender maps an IPmc host group address to an Ethernet multicast address by placing the low-order 23-bits of the IP address into the low-order 23 bits of the Ethernet multicast address 01-00-5E-00-00-00 (hex). Because there are 28 significant bits in an IP host group address, 32 (2^5) Class D addresses map to the same Ethernet multicast address.

In order to support the reception of IPmc datagrams, an Ethernet module (called a software driver) must be able to resolve upon reception the multiple IPmc datagrams that overlay a common Ethernet multicast address. This implies that the Ethernet multicast address may correspond to IPmc addresses representing multiple hosts and not just the local Ethernet address. This generally requires higher-level software control to filter in-bound IPmc datagrams by host group, which this paper does not discuss.

1.1.8 Multicast Addressing Summary

According to the recommended standard for IPmc, RFC 1112 states that a "... multicast datagram is delivered to all members of its destination host group with the same "best-efforts" reliability as regular unicast IP datagrams, i.e., the datagram is not guaranteed to arrive intact at all members of the destination group or in the same order relative to other datagrams." [Deering, 89] To date, no internet standard exists for routing multicast datagrams over IP networks.

The membership of a host group is dynamic; that is, Internet hosts such as simulation applications may join and leave multicast host groups at any time. There is no restriction on the location or number of members in a host group. A host may be a member of more than one group at a time. A host need not be a member of a group to send datagrams to it.

1.2 The TRIDIS Project

The number one objective in the TRIDIS work plan was to "provide a communications network infrastructure. . . expanded to support the needs of the DIS community. . . by incorporating other network protocols such as IPmc. . . " into the Testbed. Incorporating the IPmc protocols into a simulation application for testing and evaluation supports this fundamental objective.

The IGMP (as defined in RFC 1112) was chosen for this research because it was referenced in the Communication Architecture for DIS (CADIS) document [CADIS, 9?] and would be of future benefit to the DIS community. IGMP does not specify how local network multicasting is accomplished for all types of networks. IGMP does specify the required service interface to an arbitrary local network and gives an Ethernet specification as an example. Specifically, best-effort IPmc communications over the User Datagram Protocol (UDP) transport protocol was evaluated for this task.

1.2.1 Multicast Conformance

According to RFC 1112, there are three levels of conformance to the experimental IPmc protocol. These are defined in the IPmc specification as conformance level-0, level-1, and level-2. Hosts that provide level-0 conformance do not support the transmission or reception of IPmc datagrams. Conformance level-1 hosts will allow the transmission of IPmc datagrams, however they cannot accept incoming multicast datagrams. A level-2 host can both transmit and receive IPmc datagrams. Level-2 hosts can join and leave host groups dynamically as well as send multicast datagrams to different (multiple) IPmc groups. Level-2 hosts use IGMP to join and resign from multicast groups. Level-2 conformance was selected for analysis in the Testbed.

1.2.2 Transport Layer - IP Transmissions

DIS protocol messages, as defined in [IEEE 1278.2] may be transmitted on a network using one of the three defined transmission modes:

- Best-effort Unicast
- Reliable Unicast
- Best-effort Multicast

1.2.2.1 Best-effort Unicast

Best-effort unicast messages refer to those datagrams sent over IP and which use the UDP transport protocol (UDP/IP). UDP/IP datagrams are described as using ***best effort*** delivery practices because they require neither positive nor negative acknowledgments upon receipt of the datagram. This means UDP transmissions are efficient. Unicast transmissions refer to those datagrams that are sent from a source host to a destination host. In other words, unicast UDP transmissions describe a "point-to-point" communications path from a source to a destination. Best-effort unicast datagrams are the most efficient method for transmitting information between two hosts across a network. However, reliability in transmission can become a factor when network loading becomes high.

When using unicast UDP/IP datagrams, undelivered datagrams are the responsibility of the transmitting or receiving application. In general, the underlying network architecture is reliable and most UDP datagrams (transmitted) are delivered without mishap. Some applications prefer to trade 100% delivery reliability for efficiency of transfer. Thus the UDP datagram delivery scheme does not rely on acknowledgments, reducing the number of datagrams placed on a network.

In a distributed simulation environment, best-effort communication strategies are desirable because of their efficiency. However, unicast transmissions are not as desirable because more IP datagrams have to be generated. For example if a simulation exercise contains ***H*** hosts, then for an entity (***E***) to transmit a DIS PDU to all other entities requires that a minimum of ***H-1*** UDP/IP datagrams be transmitted. Considering that each entity (***E***) must transmit at least one Entity State PDU every 5 seconds, the smallest number of DIS PDUs transmitted in a exercise using unicast UDP/IP would be ***E(H-1)/5*** datagrams per second. When using best-effort unicast transports, network traffic will increase linearly with the number of entities times the number of hosts minus one, divided by the maximum interval (heartbeat) rate (in seconds).

1.2.2.2 Reliable Unicast

Reliable unicast messages refer to those datagrams sent over IP using the TCP transport protocol (TCP/IP). Like best-effort unicast datagrams, reliable unicast transmissions are sent from one source host to one destination host. TCP /IP datagrams are described as using reliable delivery practices because they require either a positive (or negative) acknowledgment upon receipt (or lack thereof) of a datagram. In other words, reliable unicast transmissions describe a "point-to-point" communications service path from a source host to a destination host where message delivery is guaranteed. The TCP/IP transport layer is responsible for the acknowledgment and any retransmissions, as required, to ensure delivery. Reliable unicast datagrams are therefore inherently less efficient than best-effort unicast datagrams because at least one additional datagram (the acknowledgment) must be communicated for each one transmitted.

Reliable unicast transmissions are not typically used for distributed interactive simulations. As described above, a minimum DIS PDU transfer rate can be calculated. The role for distributed simulation exercises built around reliable unicast transport protocols and containing ***E*** Entities, ***H*** Hosts and a 5 second heartbeat is given by the equation $2(E(H-1)/5 + L)$ where ***L*** indicates the number of datagrams which are lost and have to be retransmitted.

Acknowledging the receipt of every TCP/IP datagram will, at a minimum, double the number of datagrams placed on the network. When dropped, lost, damaged or corrupted datagrams are included in the analysis, additional network latency is incurred in addition to retransmitting the original datagram. That is because both the original datagram and its acknowledgment must be retransmitted. Note that using a reliable transport does not necessarily cause a doubling of network bandwidth utilization. This is because acknowledgments are generally very small. Nevertheless, when comparing reliable verse best-effort transports for service in a DIS exercise, best-effort transport paradigms are more efficient. As an exercise increases in either the number of entities or participating hosts, the minimum network load will increase at a rate that is proportional to two times the product of entities and hosts.

1.2.2.3 Best-effort Multicast

Best-effort multicast messages refer to those datagrams sent over IP and which use the UDP transport protocol (UDP/IP). Best effort delivery practices are described above in section 0. Multicast transmissions refer to those datagrams that are sent from one source host to a destination defined by a host group address. In other words, multicast UDP transmissions describe a "point-to-multipoint" communications path from one source to multiple destinations. Best-effort multicast datagrams currently describe the most efficient method available for transmitting DIS exercise information between multiple networked hosts. However, reliability in transmission can become a factor when network loading becomes high.

When using multicast UDP/IP datagrams, undelivered datagrams are the responsibility of the receiving and/or sending application. Typically during a DIS exercise when a datagram is lost in transmission, the time required to recover that information is usually longer than the time in which the information was valid. In other words, the data has become stale. Therefore, a best-effort network communications architecture is considered quite acceptable to the DIS community provided a high degree of transfer is achieved[2]. This is defined in [IEEE-1278.2] as 98% throughput efficiency in under 100 milliseconds transfer time for tightly coupled entities. Loosely coupled entities can withstand a slightly lower performance level of 95% throughput efficiency in under 300 milliseconds transfer time.

For example, if a simulation exercise contains *E* entities and *H* hosts, then for a given entity to transmit a DIS PDU to all other entities requires that a minimum of *1* UDP/IP multicast datagram be transmitted. Considering that each entity must transmit at least one Entity State PDU every 5 seconds, the smallest number of DIS PDUs transmitted in an exercise using best-effort multicast exercise would be ***(E-1)/5*** datagrams per second. When using best-effort multicast transport protocols, network traffic efficiency will increase linearly with the number of entities and proportional to the heartbeat rate.

As mentioned above, multicast datagrams are single messages that are addressed to a single (group) address but are received by multiple hosts. In order for a multicast group address to have a predefined destination, it must be one of a few well-defined network addresses recognized by the Internet Advisory Board[3]. Currently IPmc addresses 224.252.0.0 through 224.255.255.255 are reserved for DIS environments.

2. RESEARCH GOALS

The goal of this research task, TRIDIS task 3.2.3.4, was to provide the TRIDIS Testbed with the capability to test all the communications protocols specified by the CADIS document. In particular, IPmc was targeted for several reasons. First, early experiments with multicast addressing indicated that IPmc might provide for more efficient use of network bandwidth and gateway resources. Second, the DIS community believed that multicast addressing might also reduce sender and receiver resources [Smith95a]. Third, it was believed that the benefits obtained by migrating simulation applications to multicast addressing would help with DIS's apparent scalability problem. Fourth, it was believed that transitioning simulation applications from broadcast-only to multicast-specific transport protocols would resolve the problem of routing simulation traffic between Wide Area Networking (WAN) sites over commercial networks such as the

[2] The simulation community is already familiar with one form of multicast addressing called broadcasting. Broadcast transmissions can be viewed as a unique form of multicast addressing where every host on the local subnet is defined as having joined that multicast group address. However, unlike IPmc addressing, all hosts on a given subnet must (by definition) respond to all broadcast messages. This means that all network nodes (not just simulation hosts, but printers, file servers, network information servers, etc.) must evaluate and possibly discard all broadcasted DIS packets sent to the local broadcast address.
[3] The Internet Assigned Numbers Authority (IANA) is the central coordinator for the assignment of unique parameter values for Internet Protocols. The IANA is chartered by the Internet Society (ISOC) and the Federal Network Council (FNC) to act as the clearinghouse to assign and coordinate the use of numerous internet protocol parameters. The authority is documented in RFC 1700.

Internet. Finally, of all of the protocol research areas specifically proposed in CADIS, only one, the IPmc protocol defined by IGMP, is a recognized standard.

2.1 Routing

For DIS, IPmc addressing has a distinct advantage over IP subnet broadcast addressing, and that advantage is routing. IPmc addressing can support a larger number of participating hosts in a simulation exercise because IPmc datagrams can be routed from one subnet to another. Broadcast datagrams are limited to the maximum number of hosts permitted by a given subnet. This means that a greater number of networks can now participate in a common exercise.

By definition, subnet broadcast messages are local only to a given subnet and are filtered out from routed communication channels. In other words the transmission of subnet broadcast datagrams is not permitted across the internet. This means that a private network solution would be necessary to connect multiple sites together in a single simulation exercise. Thus the Defense Simulation Internet (DSI), a private network, was created. The DSI routes traffic using the Open Shortest Path First (OSPF) protocol developed by [Moy, 94a].

The DSI network was built specifically to connect multiple sites together into a single, larger network. The DSI permits multiple simulation applications that reside at different physical locations to participate in a concurrent simulation exercise. This was the case during the recent joint simulation Synthetic Theater of War - Europe (STOW-E) exercise. STOW-E implemented a common, real-time simulation exercise by joining multiple subnets together (using OSPF) into a common network.

By using a routable transport protocol like IPmc, concurrent multi-site simulation exercises could occur over public networks like the internet. To this end, protocols (like the experimental Distance Vector Multicast Routing Protocol, DVMRP) are currently being developed (by the IETF) to support multicast addressed applications on the internet [Deering, 88] [Waitzman, 88]. Once standardized, this protocol would alleviate the need for private simulation networks to be used in order to interconnect multiple simulation sites.

2.2 Resource Reservation

Routing, such as that provided by DVMRP is only a part of the problem facing the simulation community in its pursuit of the use of public networks. Another part of the communication problem is bandwidth resource reservation [Forgie, 79]. Resource reservation is described as the ability to provide end-to-end real-time guarantees over an internet. Resource reservation permits an application to build multi-destination simplex data streams with a desired quality of service [Delgrossi, 95] [Topolcic, 90]. Thus the Stream Protocol was created.

The Internet Stream Protocol Version 2 (ST2) is an experimental connection-oriented internetworking protocol that operates at the same layer as connectionless IP[4]. Resource reservation provided by ST2 is seen as a requirement for supporting the efficient delivery of data streams to single or multiple destinations in applications like DIS. ST2 permits an application to specify a guaranteed quality of service (i.e. guaranteed data rates, controlled delay characteristics, etc.) or resource from the network [Braden, 89]. The Internet Protocol currently does not provide the delay and data rate characteristics necessary to support applications (like DIS) which may require flow control [Pope, 89].

The DSI private network solution seems to provide a workable approach to resolving the protocol limitation problems current in IP. However, the private network approach has its limitations. First, there is little support (or funding) for the continued growth of large-scale, independent, private networks such as the DSI. Second, there have been significant developments in the state of the art since the development of the ST and ST2 protocols and the DSI, such that wide-spread support for non-COTS solutions are dropping rapidly.

Readers should note that standards and technology addressing alternative approaches to the resource reservation problem are currently under development within the IETF. A working group is developing the IP/rsvp protocol to manage resource reservation over IP. However, the IP/rsvp protocol proposal currently addresses only receiver-oriented resource reservations after initial connections have been established.

Distributed simulation over WANs will require resource reservations to be established by the sender when the link is initially established. DIS reservations will probably be somewhat dynamic in nature (meaning that the link requirements may change over time, but not frequently). Furthermore, IP/rsvp provides no support for aggregating resource links between common WAN nodes. This means that the managing structures of a WAN-based IP/rsvp will be more complicated and cumbersome to use. Currently, no work is being done within the DIS workshops to define resource reservation requirements within the DIS protocol. Only minimum recommendations have been established [IEEE 1278.2]. Therefore, the DIS community should participate in the development of IP/rsvp within the IETF, if practical uses of public networks are to be a goal for DIS applications.

2.3 Filtering

Filtering is still an area of research for distributed simulations. Two areas, filtering

[4] ST2 is part of the IP protocol family and serves as an adjunct to, not a replacement for, IP.

information from within the simulation, and filtering the simulation traffic from external systems are currently being studied.

2.3.1 Internal Filtering

Internal filtering of simulation PDUs by simulation applications is an area currently being studied by the simulation community. A recent focus by the DIS Special Task Force on Protocol Architecture (STGPA), of which this author is a member, has been working toward reducing the number of datagrams and bytes placed on the network during a simulation exercise. The benefits of this research will be to have distributed simulations transmit a higher percentage of useful information (and less redundant information). The goal is to produce an over-all reduction in both bytes and packets transmitted in an attempt to reduce the processor overhead associated with network communications. By producing fewer, smaller PDUs, more network bandwidth can be preserved.

To support that effort, one goal of this research was to show that those communication channels existed within the PDU data stream and could be targeted for multicast host group filtering. The various experiments executed under this contract were designed to show that network layer filtering between simulation applications may also reduce processor overhead associated with network communications. The area of research that has yet to be explored is the reduction (i.e., filtering) of inbound datagrams based on segmenting the simulation environment into logical, interacting areas. Specifically, which areas are appropriate for filtering or will offer the best performance trade-offs have not been fully explored by the distributed simulation community.

Some suggestions within the community are: proposed segmentations based on grid boundaries within terrain maps; to use areas of regional interest, such as a local battle zone; and to use only the local entity's ability to survey its regional area [Smith, 95b]. What ever the final solution will be, this area is one of open research with direct consequences on the operational efficiency of a distributed simulation application.

2.3.2 External Filtering

Applications that use subnet broadcast addressing have proliferated since the development of UDP. (Distributed interactive simulations are one such application area guilty of subnet broadcast abuse.) Now nearly 15 years later, large internetworks have grown to the point that much of the congestion on common network backbones is due to subnet broadcast addressing. Network information services (ARP, NIS, etc.) appear to be the primary users of subnet broadcast datagrams[Wobus, 89]. Subnet broadcast datagrams generate an effect on a LAN that is similar to that of noise on a radio. Increasing the level of noise (generic broadcast messages) corresponds to a decrease in signal strength (valid network datagrams) on a network backbone, hence reducing the signal-to-noise ration of an Ethernet. (After all, even a 10mbit Ethernet has a limited amount of bandwidth.)

There are several effects which DIS broadcast messages have on non-simulation hosts (or any host for that matter). Foremost, by definition, every host must be sensitive to every subnet broadcast datagram it encounters. This means that each subnet broadcast datagram is read by the host's network interface card and buffered for analysis. If no process is defined to handle the broadcast message, it is later discarded. This low-level processing does not in itself consume a lot of central processing unit (CPU) time but it does nonetheless consume resources (time and memory). If there are a sufficiently high number of broadcast messages on a given network (such as during a broadcast storm[5]) network communications can become entirely ineffective.

When a DIS exercise is run on an IP-based LAN, all hosts connected to that LAN segment will receive the DIS broadcasts[6]. (The only effective means of filtering DIS broadcast datagrams from a network is by routing between network segments.) However, not all networked hosts are simulation hosts. In general, this is not a problem if distributed simulations are run on private LANs. Private networks do not (typically) adversely impact other devices on neighboring subnets because routing can effectively filter out subnet broadcast datagrams.

Not every company can afford a private network to run simulation applications. If the receiving host (of a DIS broadcast) is a server system (e.g. file, email or print server) or a network device (e.g. router or bridge port) then there will like be a negative impact to the performance of that system. The problem is that all devices (by definition) are required to receive (not filter) all broadcast messages. Indeed, under these circumstances, using subnet broadcast addressing is not a good choice.

However, a solution does exist. By applying IPmc addressing to DIS exercises, the resultant effect is a reduction in the level of broadcast datagrams; meaning a reduction in the network's "background noise" or a lowering of the "noise floor." Using the IPmc addressing solution (instead of a subnet broadcast-based solution) provides an effect that would be similar to building a broadcast filter. Instead of filtering (removing) the non-simulation hosts from receiving the DIS datagrams, IPmc-based datagrams are sent only to the appropriate host group.

When a subnet broadcast datagram is transmitted, every host on the subnet will receive and decode that datagram. If that same datagram was addressed using IPmc instead of a subnet broadcast address, the datagram would be sent directly to the IPmc host group.

[5]A broadcast storm is an overloaded term that describes a condition where devices on the network are generating traffic that by its very nature causes the generation of more traffic. The inevitable result is a huge degradation of performance leading to a complete loss of the network medium.
[6] Multiple applications utilize broadcast messages to find out things like network state information, what machines are located on the local subnet, or the nearest-router address. With an increase in both frequency and use, broadcast messages begin to appear within network traffic analyses like a low-level noise floor, reducing the available signal-to-noise ratio of a given network medium.

Then only the simulation host's network interface would be interrupted and required to store the inbound simulation datagram.

2.3.3 Improved Application Efficiency

It is theorized that multicast transmissions will reduce the process overhead associated with broadcast addressing by permitting the receiving hosts to only respond to those datagrams which it is specifically addressed. In other words, a given simulation host will only read and decode those datagrams that match a multicast host group address to which the host has joined. Therefore if a host detects a datagram that is not addressed to it, the datagram is simply ignored and discarded by the network interface card, eliminating higher level process intervention. Furthermore, if a router determines that a subnet contains no hosts belonging to a given multicast group, the router will filter that datagram from that subnet. This double level "filtering" of datagrams reduces the amount of "background noise" created on the network, thereby allowing a simulation host to expend fewer CPU cycles managing network interrupts and in the case of simulation applications, processing more entities. The experiments reported in this paper will show that this is indeed true.

3.0 APPROACH

The TRIDIS Annual Report v.1 Rev.1 dated 28 June 1994 (IST-TR-94-17A) states: "TRIDIS will survey and evaluate existing products for the Transmission Control Protocol (TCP), Internet Protocol (IP) Multicast, and the Internet Group Management Protocol (IGMP). The TESTBED will either buy one or build custom implementations after determining which would be more effective."The approach taken by the TRIDIS team included the following:

- Repair the Standard
- Survey Protocol Products
- Select and Purchase a Product
- Integrate New Protocols into Testbed
    - Keyboard Blocking
    - X-Windows Interface
    - Inter Process Communications
- Conduct Initial Tests

- Evaluate Implementation and Recommend Fixes[7]
- Port to IRIX
    - Implement Modifications
    - Measurement and Evaluation
    - Write Final Report

3.1.1 Repair the Standard

IPmc is defined as a best-effort approach to transport protocols. In other words, IPmc is only defined to run over UDP. DIS applications may still choose to use TCP for point-to-point communications, such as in simulation management, but host group addressing as defined in the IGMP standard is currently undefined for TCP transport protocol use. The TRIDIS work plan specified TCP-based multicast routing. Having noted that no standard has yet been defined for reliable multicast, this requirement as stated in the TRIDIS work plan was not implemented. In addition, no attempt was made to build an experimental TCP-based multicast routing protocol. This is an area of current research within the IETF.

However, the TRIDIS team also researched the then-current DIS standard and found a similar discrepancy regarding multicast TCP. So through the DIS standards process, the author joined the Tiger Team Ballot Review Committee, which provided an opportunity to correct the recently balloted version of the standard. The latest revision of the DIS standard Communication Architecture and Profiles, IEEE-1278.2, dated August 1995 has removed any and all references to reliable (TCP) multicast communications. Only TCP unicast, and unicast and multicast UDP transport protocols are recognized for use with DIS PDUs.

3.1.2 Survey Protocol Products

The TRIDIS team discussed several options on how to proceed. The first method involved implementing IPmc in the DOS version of the IST Computer Generated Forces (CGF) Testbed. However, when the CGF Testbed was designed and implemented, only a minimal network interface was constructed. In order to implement IGMP, a full implementation of the IP stack was required. The team agreed that developing a new IP

[7] Due to time constraints, the recursive steps of evaluate, fix and implement modifications were not performed. Problems were noted with the results in both experiments #1, #3 and #5 which could have been remedied had the time been allotted to complete these steps.

network interface driver[8] for the PC was probably beyond the scope (and budget) of the project. For this reason, a complete IP implementation would have to be purchased or developed to support IPmc in the Testbed.

The TRIDIS team determined that it would be to costly in terms of development time and man-hours to implement a custom "in house" version of a full IP stack for DOS machines. A full protocol stack is necessary to properly implement the DIS protocol standard, as defined in [IEEE-1278.2]. Therefore, it was decided that a commercial version of an IP stack should be researched and procured.

3.1.3 Select and Purchase a Product

The TRIDIS team conducted a survey to find available commercial-off-the-shelf (COTS) implementations of IPmc which would be compatible with the DOS version of the CGF Testbed. At the time of the survey, the results showed that only one company produced a product that claimed to implement IPmc. This product was made by FTP Software, Incorporated. A purchase order was made and FTP software for DOS was ordered.

The product which arrived from FTP software however, was not a complete IP implementation. It lacked several of the key features necessary to implement a IPmc communication interface stack. For example, the FTP implementation was missing the Internet Group Management Protocol (IGMP) functionality. Additional support for the Distance Vector Multicast Routing Protocol[9] (DVMRP), as described in RFC 1054 [Deering88] was also missing. IGMP is used to join or resign a host from a multicast group address.

DVMRP is the multicast routing protocol that the team wanted to evaluate because of its wide distribution and use within the Multicast Backbone (MBone) community. The Mbone is an experimental multicast architecture, superimposed over the existing internet, and is used to provide multicast connectivity between multiple sites. MBone connections (called tunnels) are made using typical TCP/IP unicast transmissions that carry the multicast datagram payload. At one end of a communication link, a multicast router is responsible for encapsulating the IPmc datagram into a TCP/IP datagram. Transmission and delivery are the same as for normal IP, with the receiving unicast address being the receiving multicast router. Upon reception, the multicast datagram is de-encapsulated

---

[8]In addition to the raw man-hour requirements needed to build a custom PC socket interface, there was also the issue of available Random Access Memory (RAM) within the PC environment when the DIS CGF was loaded. The team later learned that the minimal network implementation approach was originally used because there wasn't enough available RAM in the first place. Had this choice been made, other more involved changes to the CGF application would have been required.

[9] Note that as of this date, the Distance Vector Multicast Routing Protocol (DVMRP), as described in RFC 1054 has been granted experimental protocol state with a status of not recommended by the IAB.

and then transmitted on the receiving LAN. Without IGMP or DVMRP support, implementing level-2 multicast addressing conformance for the DOS version of the CGF Testbed would not be possible.

3.1.4 Integrate New Protocols into Testbed

Since a COTS solution for IPmc was unavailable for the DOS CGF Testbed, another solution had to be found in order to complete this task. As mentioned above, IPmc support existed in the UNIX domain because of the development work by Steve Deering, Van Jacobson and the MBone community. The complete IPmc stack was available in most commercially available UNIX systems because the software developed by the MBone community to support IPmc was made available to UNIX vendors at no charge. Vendors like Sun Microsystems and Silicon Graphics incorporated this interface stack into their UNIX sockets.

After much consultation, the TRIDIS team made the decision to port the DOS version of the CGF Testbed to UNIX and integrate the IPmc software into this application. Two Sun Sparcstation-1's capable of running Solaris v.2.4 were identified and were available for use. The Sun platforms required additional disk space and additional Random Access Memory (RAM) in order to run the current version of the Sun Solaris operating system. These items were procured and installed.

The Sun Solaris v2.4 operating system was chosen for three reasons: First, the Sun machines were available and not currently in use. Second, IGMP support already existed in the Sun UNIX socket driver delivered with the Solaris system. Third, IPmc addressing was initially developed on the Sun platform. Therefore, a complete IP network interface stack and existing socket driver could be combined to make the network interface development task a straightforward implementation.

In order to port the CGF Testbed from DOS to UNIX, several simulation application subsystem components were modified to work in the new environment.

3.1.4.1 Keyboard Blocking

The newly ported CGF Testbed also needed to accept keyboard interrupts. With UNIX, programs would sit idle (foreground) or continue to run (background) until they receive input from the keyboard. In DOS, a non-multitasking environment, the CGF Testbed required a Terminate and Stay Resident function (called a TSR) to enable the program to continue running while awaiting keyboard input. One problem introduced by the porting operation was the need to incorporate similar support for non-blocking keyboard input.

3.1.4.2 X-Windows Interface

The ported system would require an X-window interface to operate in the UNIX environment. In the DOS version of the CGF Testbed, a DOS window was used for the Plan View and Stealth View displays to draw both two and three dimensional objects. In order to port the CGF Testbed, we needed at least a minimal window which was capable of echoing typed characters and displaying console messages. An X-window interface was developed to handle the keyboard X-events, but graphical development (2D and 3D line art) was postponed for later work. [10].

3.1.4.3 Interprocess Communications

Other less significant modifications were needed in order to get the CGF Testbed to function correctly in the new environment. For example, the new keyboard window process needed to be connected to the parent simulation process using Inter-Process Communication (IPC) queues. The UNIX CGF Testbed implementation needed to simulate the non-blocking keyboard input of the DOS version of the CGF Testbed.

Also, byte and bit swapping had to be performed because the byte/bit order of the Intel 80X86 platforms was different from the byte/bit order of all of the RISC-based UNIX machines in the laboratory. Once these deficiencies were identified and corrected, the Testbed appeared to be operational, preliminary functionality testing began.

3.1.5 Conduct Initial Tests

After porting the CGF Testbed from DOS to the multitasking environment of UNIX, several tests were conducted to ensure correct operation of the simulation system. Operational testing of the Testbed was needed to determine if the IPmc socket interface was functioning properly. Additional concerns about the simulation itself also had to be addressed. Before we could use the simulation Testbed, the team needed to be sure that the simulation application performed accurately. An initial system test was performed by loading existing simulation script files into the ported CGF Testbed and running it against its PC-based version.

To implement the multicast experiments, TRIDIS staff needed a minimum of four PC's; two machines located on each side of an IP gateway (i.e. router) that could all run the CGF Testbed simultaneously. These machines also needed to be able to run the same versions of IGMP and DVMRP. Of the machines on the TRIDIS project, only two Sun workstations were available for this task. Two more workstations needed to be located

[10] After the IPmc experimentation was completed, work continued with the development of the X-Window interface. A display window manager was added, as was a Plan View Display, and type-in and type-out windows.

before the experiments could begin.

Several PC workstations capable of running the current version of LINUX were initially identified. LINUX is a POSIX compliant, UNIX-derivative operating system which runs on IBM compatible machines and is currently undergoing development across the Internet. However, as the team learned through experimentation, the underlying system within LINUX is not a derivative of either AT&T or BSD UNIX. Consequently, not all of the interface modules common (POSIX-compliant) UNIX are as yet supported within LINUX.

An attempt was made to port the CGF Testbed to this platform combination and was initially successful. Once the CGF Testbed was compiled on the LINUX platform, broadcast functionality tests were conducted. During this test, it was determined that the LINUX implementation did not yet support IGMP. Without IGMP support, LINUX IPmc testing could not be performed. As mentioned earlier, IGMP support is necessary for Ipmc-based applications to join and resign from multicast host groups. In order to make this system operational, the TRIDIS team would have had to build parts of the LINUX kernel to accept IGMP socket requests. Again, the team reviewed this approach and decided that it was too costly to implement and other solutions were reviewed.

3.1.6 Evaluate Hardware Requirements

Since time was growing short and the performance period for this project was drawing to a close, the TRIDIS team made the decision to port the CGF Testbed to another UNIX platform. Two Silicon Graphics Indy workstations were identified and borrowed from another task. The two Silicon Graphics workstations were configured to run SGI's IRIX v5.3 operating system. IRIX is another variant of AT&T UNIX and IPmc support for multicast addressing already existed for this platform.

3.1.7 Port to IRIX

After porting the CGF Testbed to SGI's IRIX, the Testbed simulation application was tested for, and passed, preliminary functional testing. Initial test results showed that the SGIs had the necessary support for IPmc. In fact, the SGI machines ran almost 10 times faster then the Sun Sparcstation-1 counterparts[11]. The significance of this result would not manifest itself until the final analysis of the multicast experiments outlined in section 5.

[11] No thought had been given at this time to any disparity in machine performance. This would later turn out to be one indicator which heralded significant results.

3.1.8 Equipment Setup

The network was configured for IPmc only. The Alantec Powerhub-7000 router was configured to allow Ethernet ports E9 and E6 to operate as two isolated, routed subnets. All external traffic was filtered from ports E6 and E9, and only IGMP addresses were permitted to flow between these two ports. This prevented external network traffic from competing for network resources on these two subnets during the experiments. Figure 1 shows the general setup for the multicast experiments used by the TRIDIS team.

Identical simulation applications were installed on each of the four machines in Figure 1. The simulation application internally logged both inbound and outbound DIS PDUs and placed a time stamp on each datagram received. Outbound datagrams were counted so that the precise number of ethernet frames generated for each experiment could be counted.

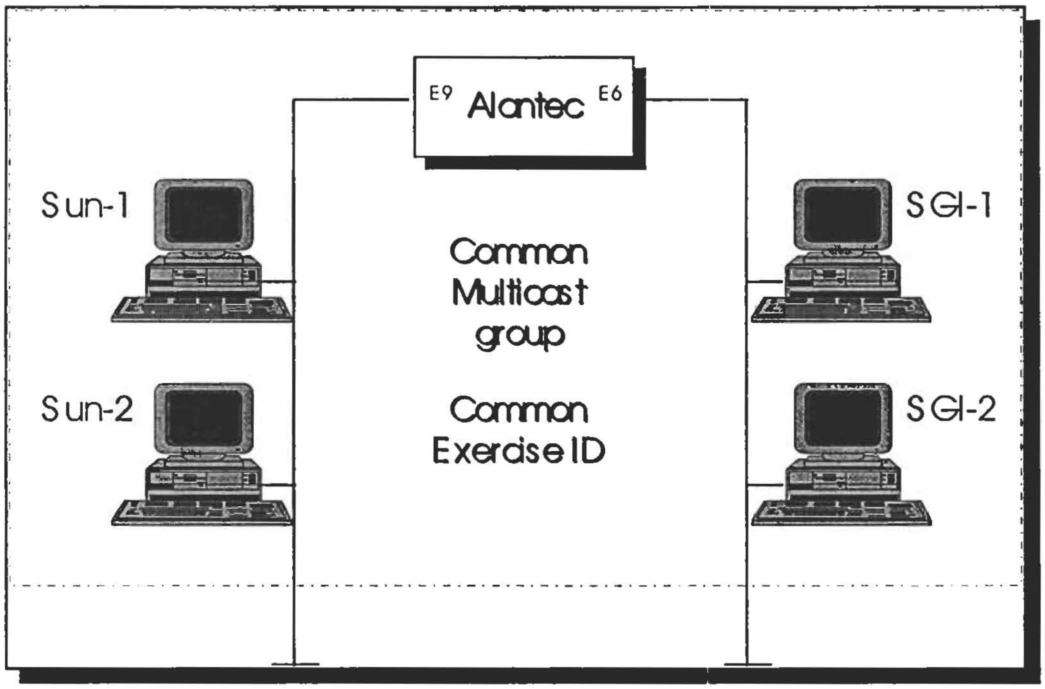

Figure 1 - Architectural layout for the TRIDIS IPmc experiments.

Although 2 PC data loggers were originally set up to collect the DIS v2.0.3 PDUs, none of the binary log files used. The multicast experiments were not designed to focus on the internal simulation application, but instead were designed to analyze the network flow of ethernet frames in a multicast environment. The log files generated by each simulation application more accurately displayed the level of interaction each simulation host played within each exercise.

4. EXPERIMENTS

Several experiments were designed for the purpose of testing the functionality of IPmc in a simulation environment. Specifically, we wanted to explore the effect that multicast addressing has on CPU utilization when either network layer filtering or application layer filtering techniques are used. Five experiments were designed. Testing used the CGF Testbed as both the transmitting and receiving benchmark application. A minimum of four workstations were required to perform these experiments and were configured as described in the previous section.

4.1 Theory of Operation

The TRIDIS team's hypothesis regarding the benefits of IPmc addressing states that any performance improvement gained by a simulation application using IPmc filtering would be due to the decreased process cycle time required to decode and interpret inbound network datagrams (as described above). If the application never had to decode and interpret those PDUs (because they were already filtered out of the input queue by the network interface layer) then some amount of performance improvement gained by IPmc filtering would become immediately apparent. In order to show how this hypothesis might be true, the simulation exercise was deliberately segmented.

4.1.1 Segment #1

The five experiments can best be viewed when divided in three sets. The first set consists of one experiment. Experimental exercise #1 was designed to *simulate* a classical DIS broadcast-based simulation by using a single multicast address. Of course the ideal scenario would have been to have conducted two experiments, one broadcast simulation and one single-group multicast simulation. The comparative results from this additional broadcast experiment could have been used to validate the initial multicast experiment. However, due to time constraints, the optional broadcast-only simulation exercise was never conducted.

4.1.2 Segment #2

The second set of experiments were a pair of multicast exercises which paired the machines into two groups consisting of two machines each. In the first exercise, the simulation hosts were grouped by subnet. This scenario created two concurrent simulation exercises. However, by using the network router as a filter, only half of the simulation traffic was seen on each subnet. In the second exercise, the two simulation hosts were grouped such that each host group forced a network router device to be in the communication path.

4.1.3 Segment #3

The third set of experiments were designed to replicate simulation processes when simulation applications grow large, as in STOW-E. What happens in large scale simulations is that a simulation application must dead reckon a larger number of concurrent simulation entities, whether or not the host's resident entities were interested in the others or not. Also, initial results from ModSAF testing by Loral [Smith, 95] seemed to indicate that simulation exercises can grow to manage more concurrent entities in a multicast environment than in a subnet broadcast environment because their network-layer filtering affords them more CPU process cycle time to manage additional entities. This set of experiments was intended to verify this idea.

4.2 Experiment #1

Experiment #1 involved creating a simulation exercise that mimicked a typical DIS broadcast-based[12] simulation exercise. To accomplish this, the team created a simulation exercise that utilized a single host's multicast group address on two LAN segments. Experiment #1 used a common Exercise ID so that any traffic analysis or performance difference with later experiments (#4 and #5) could be made. The common multicast group address used in experiment #1 provided each participating simulation application with a coherent view of the simulated world. The physical configuration for experiment #1 is shown in Figure 2.

[12] Note that a subnet broadcast communication, denoted as a *one-to-every* type of communication mode, can be viewed as a specialized form of multicast communication, denoted as a *one-to-many* mode of communication. So except for their physical address representation (all 1's for a given subnet) IP subnet broadcast communications are a form of multicast communications within a local area network.

The simulation communication architecture for Experiment #1 involved connecting two pairs of dissimilar workstations through a multi-port Alantec Powerhub-7000 router. Two router ports (marked E6 and E9 in Figure 2) provided connectivity between two Sun Sparcstation-1's and two Silicon Graphics Indy workstations. The Alantec PowerHub 7000 router provides the capability to either bridge or route Internet Protocol TCP and UDP traffic, and can route the experimental IPmc traffic using DVMRP and IGMP. Figure 3 shows the logical architectural layout for the first IPmc experiment.

One goal of the TRIDIS project was to provide support for the DIS standards community. Experiment #1 was in part designed to address this goal by showing that a typical subnet broadcast-based simulation exercise(shown in Figure 3) could be emulated using a single multicast address. It is widely accepted within the DIS community that DIS applications have not migrated to multicast addressing because of hard problems and concerns with the dynamics of host address group allocations.

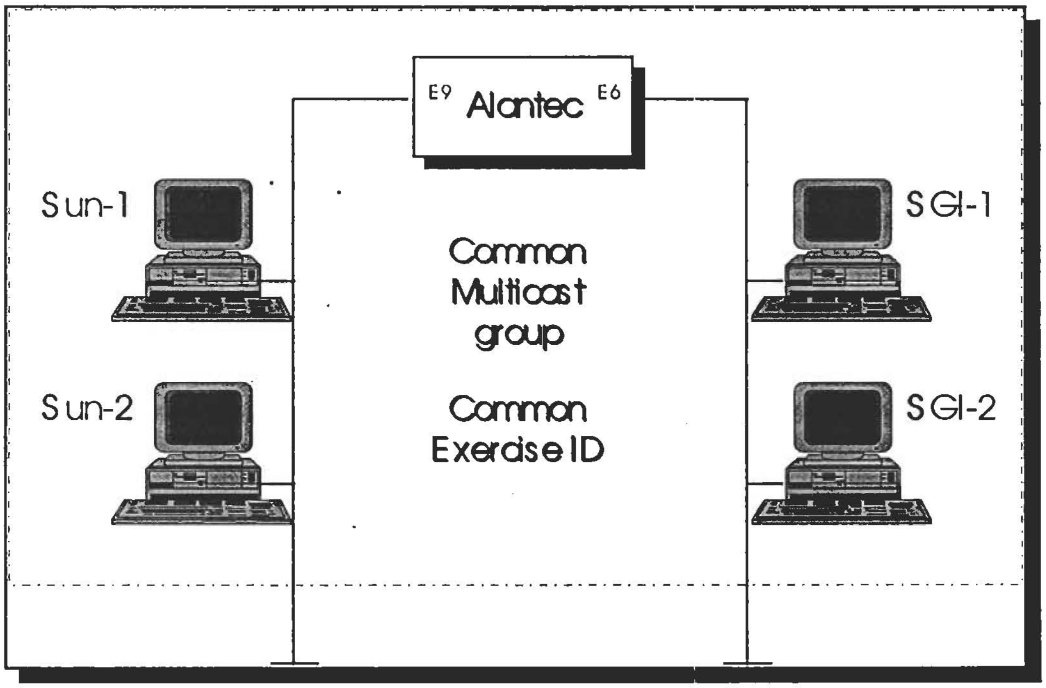

Figure 2 - Physical Configuration For TRIDIS IPMC Experiment #1.

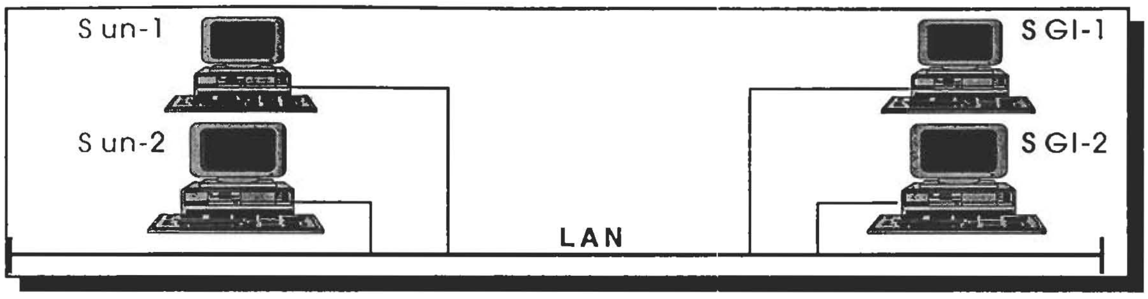

Figure 3 - Logical configuration for TRIDIS IPmc experiment #1.

In order to support this indirect goal, the TRIDIS team proposed that the first multicast-based distributed interactive simulations (similar to this experiment) be considered as the logical starting point from which to begin the broadcast to multicast migration. To this end, a formal proposal was made to the DIS Communication Architecture Working Group during their interim meeting in July, 1995. In order to help wean the simulation community from broadcast-based simulations, the TRIDIS team proposed that initial simulation exercises first migrate to a single, recognized multicast address for DIS.

Experimental evidence for this transition scenario was lacking at the time of the proposal, so experiment #1 was run in a broadcast emulation mode in order to show that the transition was relatively simple. The team learned that if the simulation application uses a standard network interface (such as UNIX Sockets) then only a minimal effort is necessary to convert a broadcast-based simulation application (one which uses broadcast addresses exclusively) into a multicast-based simulation application (one which uses a single multicast address exclusively). Additionally, to further support the simulation community's transition to multicast addressing, TRIDIS team members have assisted in the development of a multicast addressing guidance document for the DIS community. This document is currently under development by the Communication Architecture Working Group and will provide to the public suggestions, recommendations and code segments learned during the course of this work.

4.3 Experiment #2

The second multicast experiment involved creating a segmented simulation exercise in which a large portion (approximately half) of the simulation PDUs would be filtered from each receiving simulation application. If there was to be any benefit from using multicast addressing as a network-level PDU filter, then its effectiveness could be easily demonstrated if the entities were segmented into two multicast group addresses.

For example, if the simulation exercise was divided into two parts using different exercise Ids (as in experiments #4 and #5), then upon receipt of a network datagram each simulation application would be required to take a sequence break, decode the inbound network datagram, recognize and then interpret the PDU for the appropriate exercise ID number. The application would then have to either act on or reject the newly arrived PDU according to the embedded exercise ID number.

Simulation experiment #2 required that each of the simulation platforms, located on each subnet be joined to the same multicast group address. Experiment #2 showed how a router, an independent network device, can act as a low-cost simulation PDU filter while still providing for wide area network communications[13]. Network routers can filter multicast traffic by preventing certain multicast datagrams from passing from one subnet to another. This is unlike subnet broadcast transmissions which are, by definition, not forwarded[14] through a router.

The setup for this test involved configuring the two Sun workstations to use a common multicast group address. Similarly, the two SGI workstations were configured to use a different, but common multicast group address. The configuration for experiment #2 is shown in Figure 4.

[13] Wide area network communications are possible when using IPmc using a technique called *Tunneling*. Because IP is a routed protocol, a static route or tunnel is established between two sites which agree to use the same multicast group addresses. The IPmc datagrams are then embedded within an IP datagram by a special purpose multicast routing process called *mrouted*. The resultant datagram is then routed to the predefined destination address. Upon arrival, the Datagram is unwrapped and retransmitted on that LAN..
[14] Subnet broadcast communications are, by definition, designed strictly for LAN communications. Currently, WAN communications are possible for DIS exercises because a separate WAN called the Distributed Simulation Internet (DSI) has been deployed. Any site which attempted to broadcast to the entire Internet would be ostracized by the Internet Society and most likely administratively disconnected by either the local internet access provider or any available Internet Engineering Task Force member.

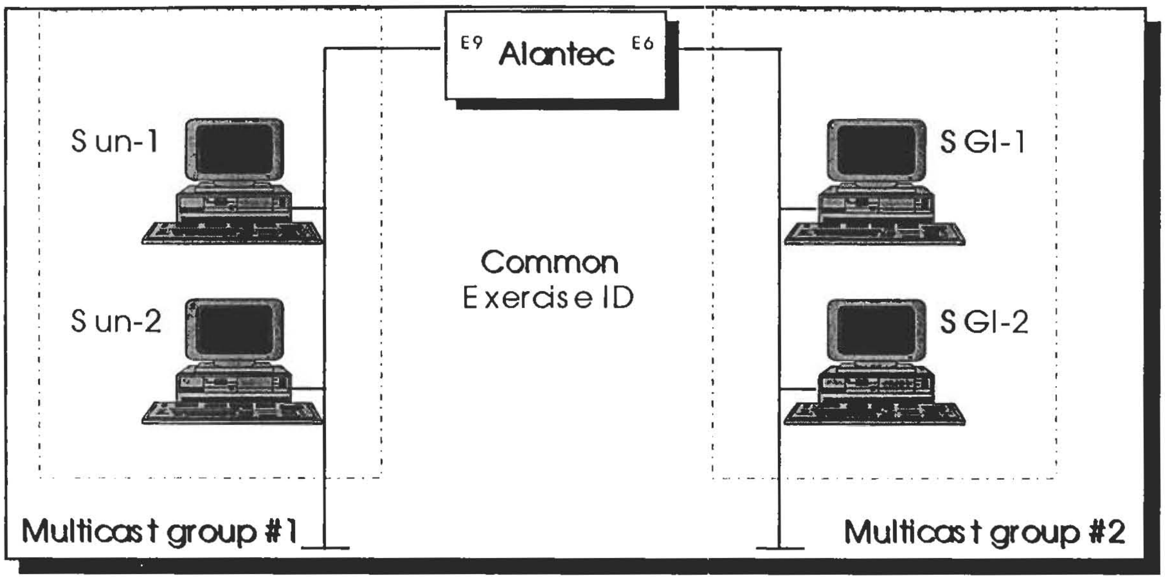

Figure 4 - Configuration for Experiment #2

The Sun workstations were connected to port E9 of the Alantec PowerHub. Off of the same router, the SGI's were connected to port E6. The router was cleared of all statistical address information and pre-assigned routes. The ports were then cleared of any external traffic and protocol filtering was enabled. The results verified that multicast filtering was occurring in the router.

4.4 Experiment #3

Experiment #3 involved creating a multicast-based simulation exercise in which a large portion of the simulation datagrams would be filtered from the receiving simulation applications. However, unlike experiment #2, the Alantec PowerHub will be routing multicast traffic across ports E6 and E9 (as shown in Figure 5). Multicast routing occurred in this experiment because hosts attached to each port have joined common multicast group addresses. The router has added these groups to its IPmc routing table, thus permitting traffic to flow. Note that in experiment #3, the effectiveness of network-layer filtering was moved from the router (as in experiment #2) to the network adapter on each simulation host.

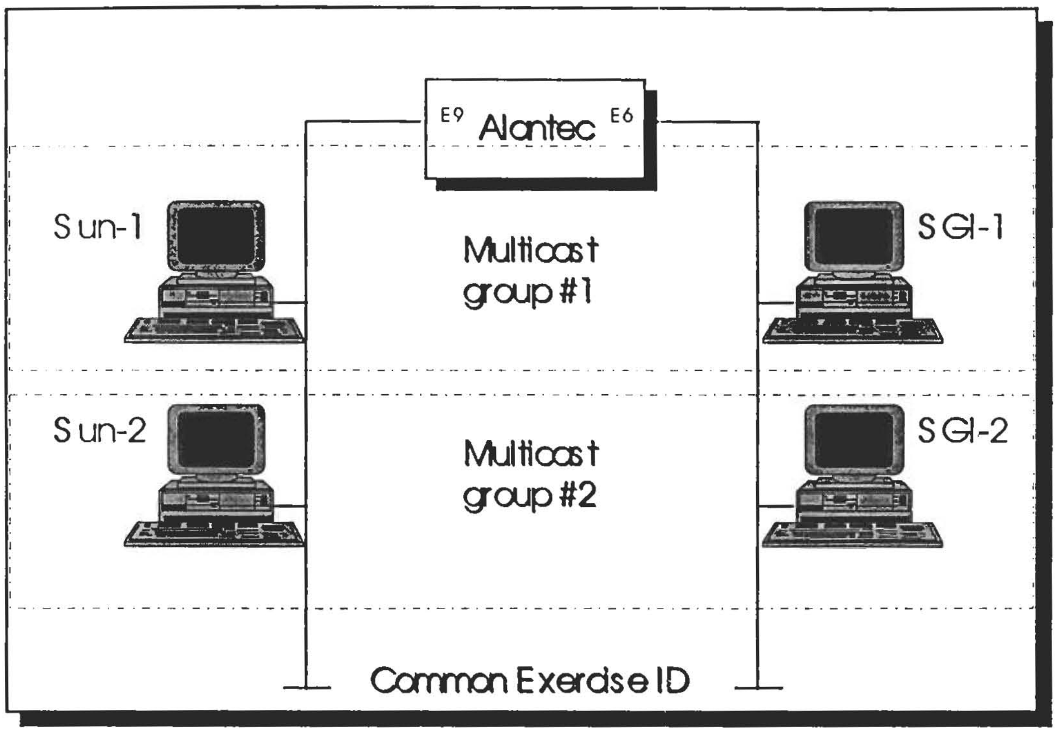

Figure 5 - Configuration of Experiment #3.

Experiment #3 was conducted to show that multiple multicast group addresses can coexist on the same physical network. In experiment #2, the router acted like a multicast datagram filter between the two subnets because no common multicast group address existed on both routed ports (E6 and E9). If a single host on any given subnet joined an existing multicast group, then the router in question must (by definition) pass any datagrams that are destined for that group, to that subnet. In the case of experiment #2, the router was presented with different, mutually exclusive multicast group addresses for each port. In experiment #3, the same multicast group addresses existed on both routed ports so the router routed (by definition) the multicast datagrams from one subnet to the other.

This experiment was designed to be similar to multiple, concurrent simulation exercises which use different exercise identification numbers (e.g. as in experiment #5). However, instead of the simulation application providing for PDU filtering, filtering took place at the network interface layer, before the receiving application process was involved. The primary difference between this simulation experiment and experiment #2 was that all simulation traffic is seen on both subnets.

The setup for this experiment involved configuring one Sun workstation and one SGI workstation for a common multicast group address . Similarly, the other Sun and SGI workstations were configured to use another multicast group address. The configuration for experiment #2 is shown in Figure 5 above. Experiment #3 was designed to highlight network level filtering at the host's port. Two host groups were selected for this experiment, just as in experiment #2. However, in experiment #3, there was at least one machine attached to both multicast group addresses and connected to each port off of the Alantec router.

It was theorized that this bi-directional connection would show similar levels of network traffic on both LAN segments while still providing a higher degree of network-layer filtering from the application process. In Figure 5, the host group configuration for experiment #3 shows that machines SGI-1 and Sun-1 were designated as members of the same multicast group address. Machines SGI-2 and Sun-2 were designated as members of the second multicast group address. With the router properly configured, all simulation datagrams generated from subnets E6 and E9 for either multicast group address successfully passed through the router without being filtered.

4.5 Experiment #4

The configuration of experiment #4 was similar to the configuration of experiment #2. However, in experiment #4, all four workstations were joined to a common multicast group address with differentiation occurring based on exercise ID. As shown in Figure 6, experiment #4 placed the two Sun workstations in the same exercise. Similarly, the two SGI workstations were placed in another concurrent exercise. The differentiation between experiment #2 and experiment #4 enabled the team to analyze and compare the results of the two experiments because the same simulation script was run for both experiments.

Multiple concurrent simulations may be a common occurrence in a simulation development laboratory. The two concurrent simulation exercises produced in experiment #4 were designed to show that there was interaction between the two simulations, even though the PDU's were segmented into two groups by exercise ID. Experiment #4 highlighted the fact that mutually exclusive simulation events occur on a common LAN could and did impact one another. In experiment #2, the datagrams received by a simulation host were those which were destined for that host's simulation application. In experiment #4 however, a large number of datagrams received by the simulation host were not appropriate for the host's simulation application. The result was that the application had to filter out the unwanted datagrams received from the network layer.

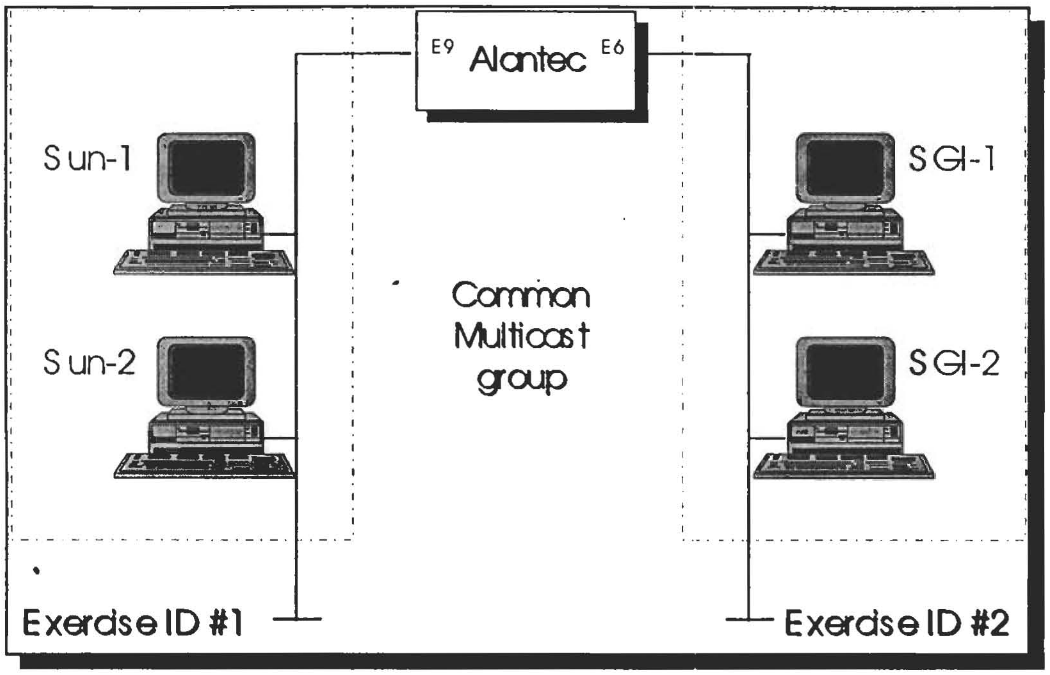

Figure 6 - Configuration of Experiment #4

As stated above, the network datagrams which would have normally been filtered by the network were filtered by the simulation application. This application layer filtering created an additional work load on the simulation host. This additional work load was presumed to be measurable and related to the number of invalid[15] inbound DIS PDUs. Removing this work load from the simulation process (as in experiment #2) provided the type of filtering which can potentially improve overall simulation performance, however, proper selection of multicast group addresses, efficient entity segmentation, and appropriate management of entity to IPmc group address relationships are other areas of research which should be investigated.

Consequently in a traditional broadcast-based simulation, additional processing time may be required to manage state information and dead reckon entities which the local host may not necessarily need to track. Initial tests into multicast addressing conducted by [Smith, 95] indicated that eliminating the time required to process these additional inbound datagrams can improve the performance of the overall simulation, making room for more entities within the simulated environment. However, there is probably an upper

[15] As defined by an inappropriate exercise ID for the local simulation application.

limit to this strategy because as more entities are created, more process cycle time will be required to manage these additional entities and their interactions.

The results of experiment #4, when compared to experiment #2 should show a marked increase in the total number of Ethernet frames received by a simulation host. Results from experiment #4 should also show a marked decrease in the total number of DIS PDUs processed as a percentage of DIS PDU's received, indicating that a performance hit has taken place. Identical simulation scripts were run for both experiments.

4.6 Experiment #5

The configuration of experiment #5 was similar to the configuration of experiment #3. However, in experiment #5 all four workstations were joined to a common multicast group address with differentiation occurring based on exercise ID. As shown in Figure 7, experiment #5 placed one Sun machine and one SGI machine into the same simulation exercise, while the other Sun and SGI machines were placed into the other exercise. Each exercise used different exercise identification numbers. Since the same simulation script was run for both experiments #3 and #5, this difference between experiment #3 and experiment #5 enabled the team to analyze and compare the results of these two experiments.

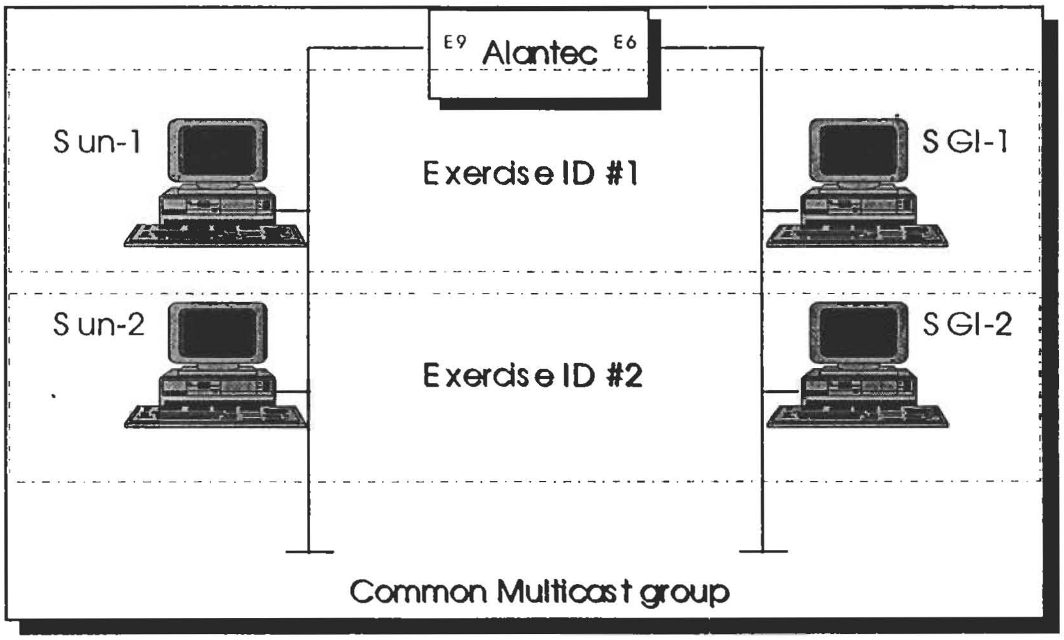

Figure 7 - Configuration for Experiment Five

As mentioned in the previous section, multiple concurrent simulations may be a common occurrence in a simulation development laboratory. Producing the two concurrent simulation exercises in experiment #5 should show that there is indeed interaction between the two simulations, even though the PDU's are segmented into two groups by exercise ID. Experiment #5 also highlighted the fact that mutually exclusive simulation events occurring on a common LAN could and did impact one another at the network interface layer. In experiment #3, the number of datagrams received by a given simulation host were those which were destined strictly for that simulation application. In experiment #5 however, a large number of datagrams received by each simulation host were not appropriate for the host's resident simulation application. The result is that the simulation application filtered out the unwanted datagrams received by lower level network interface.

As with experiment #4 and mentioned above, two concurrent simulation exercises occur within experiment #5. Again, this scenario created a situation where application layer filtering became a common occurrence throughout the exercise. This created additional work for the application and should be measurable. This increase in workload should be proportional to the number of invalid[16] inbound DIS PDUs. Removing this work load from the simulation process (as in experiment #3) provided the type of filtering which can potentially improve overall simulation performance. Again, proper selection of multicast group addresses, efficient entity segmentation, and appropriate management of entity to IPmc group address relationships are other areas of research which should be investigated.

In experiment #5, each simulation host which was required to first receive, then decode that DIS PDU, whether or not the PDU addressed the exercise to which the host was joined. If the PDUs exercise ID indicated that this PDU was for the "other" simulation, then the simulation application lost some finite amount of process cycle time. This is precisely the type of filtering which takes place when multiple multicast group addresses are used to segment simulation entities using network layer information.

As in experiment #4, as each DIS PDU was received in experiment #5, a timer was started which counted the microseconds that the system time and application time consumed decoding the inbound datagram. The system time accounted for the performance loss due to the system taking a sequence break and interrupt processing time, while the application time recorded the time used to decode the PDU and determined whether or not it should be discarded. These times were then recorded and tallied for each datagram received.

The results of experiment #5, when compared to experiment #3 were designed to show

[16] As defined by an inappropriate exercise ID for the local simulation application.

that the total number Ethernet frames received by a simulation host can exhibit a marked increase based strictly on external events. Experiment #5 was designed to also show a marked decrease in the total number of DIS PDUs processed as a percentage of DIS PDU's received, indicating that a performance hit was taking place. Note that identical simulation scripts were run for both of these experiments.

5. RESULTS

The IPmc experiments were conducted using IPmc routing and address group management over IEEE-802.3 (10base2) Ethernet. Both Alantec router ports (E6 and E9) were configured to filter out any external traffic and prevent that traffic from interfering with the experiments. Router port statistics were manually cleared prior to each experiment and manually recorded immediately after each experiment. All simulation applications were compiled and manually started. Each simulation application loaded and executed a CGF Testbed script file which created four entities and set them into motion. An additional delay of 100 seconds (for a total average runtime of 135 seconds) was added at the end of each script file and each simulation application was allowed to terminate normally.

After the experiments were completed and analysis began, an error was detected in the CGF Testbed script file located on the machine designated as SGI-2. This error caused the machine designated SGI-2 to produce only three entities instead of the four. The impact was equal to all five simulation exercises. The effect was such that SGI-2 produced slightly fewer DIS PDUs then did machine SGI-1 for all five experiments

The following sections describe the results and analysis conducted for each of the five IPmc experiments described above.

5.1 Results of Experiment #1

Experiment #1 was designed to show that each of the four simulation hosts could communicate with each other in a typical broadcast-based simulation exercise[17]. Experiment #1 was designed and executed without external Simulation Management (SIMAN) control for simulation start or stop. Therefore it is quite possible that some simulation applications could have started before others began receiving datagrams, causing some datagrams not to be counted. This would mean that some simulation applications did not receive some DIS PDU datagrams. In the final analysis, these missed

[17] The TRIDIS CGF testbed was originally implemented using subnet broadcast addressing. However, in order to test the new multicast socket interface, experiment #1 was conducted using a single multicast address to emulate a subnet broadcast-based exercise.

network datagrams would have been tallied as dropped datagrams. However, since the start-times for all four simulation applications were within a second of each other, and the maximum number of dropped datagrams was relatively low, this was not considered to be a significant problem.

5.1.1 Layout

In experiment #1, four simulation applications were compiled and installed on four simulation hosts (two on Sun Solaris v2.4 and two on SGI IRIX v5.3). The simulation application was generated using a common IPmc address and a common simulation exercise ID. Four entities were instantiated on each host (3 on SGI-2) and were placed in rapid motion. 180-byte Entity State PDUs were produced by each simulation application. Each simulation application generated DIS PDUs as fast as the host processor could cycle through the simulation application.

The single multicast group address used in Experiment #1 proved just as effective as a subnet broadcast address. However, during the analysis of this exercise, a few differences between this exercise and a true broadcast exercise were noted. Typically a subnet broadcast message generated on a local area network will impact every networked node connected to that LAN segment. For example, during a simulation event, DIS PDUs broadcasted from simulation hosts would adversely impact file servers, printers and other non-simulation hosts. Since printers and file server nodes are generally not interested in broadcast-based simulation traffic, they will be performing useless work. Nevertheless, the work that they perform (e.g. execute a sequence break, perform a processor interrupt, decode the inbound datagram, discard the data after decode, and then resume processing where they left off) is required. The "impact" on the network of subnet broadcast communications is to slow networked applications and services such as file sharing and printing, as well as other non-DIS network traffic to almost unusable levels.

The performance level of the machines designated Sun-1 and Sun-2 appeared to be quite inferior when compared to the two SGI Indy machines. The resultant data from the two Sun Sparcstation-1's shows that they could only keep up with about half of the SGI's output data stream. The data in Figure 8 appears to indicate that this experiment produced a simulation exercise which would have been unacceptable (invalid) for most training purposes. Both of the slow Sun machines failed to generate a sufficient number of DIS PDUs and in addition dropped nearly have of all inbound datagrams.

Initially, the plan for exercise #1 was simply to test and verify the IPmc routing mechanism and verify that IPmc routing does indeed work. (By definition, broadcast datagrams are not passed through routed interface ports.) Routing simulation traffic using the IPmc protocol is seen as necessary for simulation exercises to extend beyond the

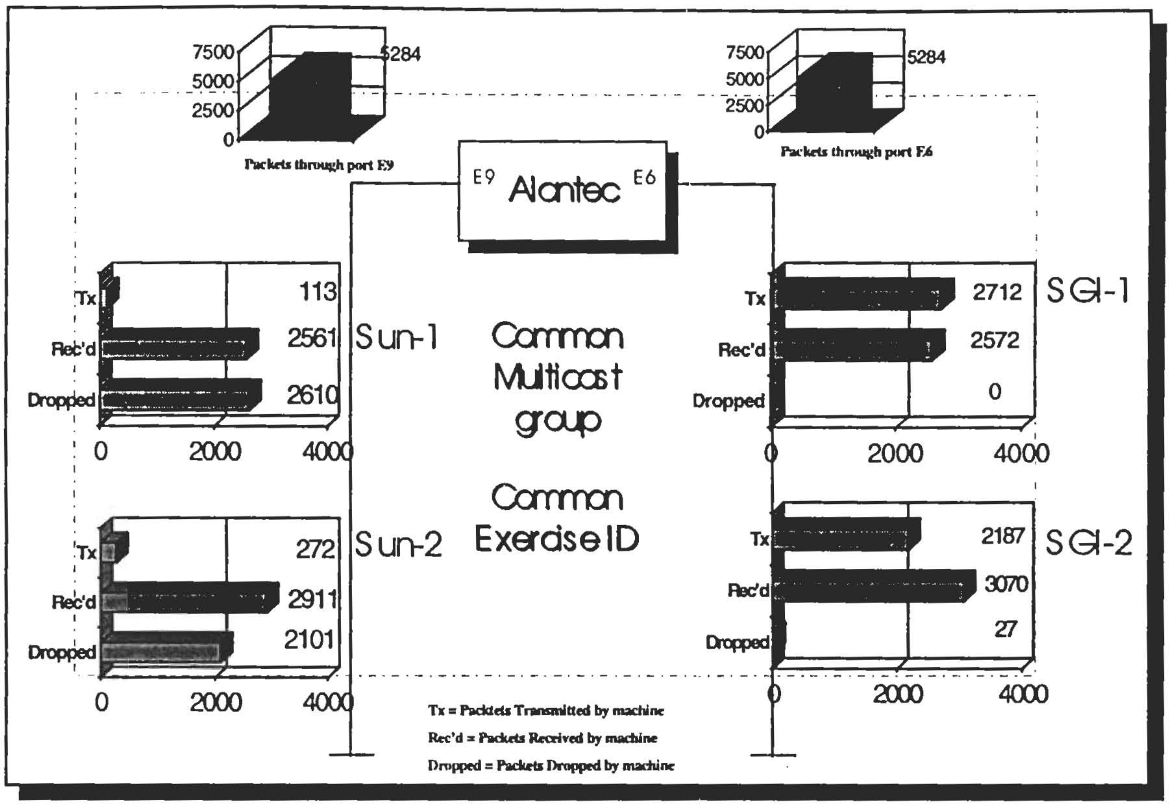

Figure 8 - PDU transmit and receive counts from experiment #1.

boundary of a LAN and into commercial wide area networks.

In addition, the results from this experiment indicate that the use of multicast routing protocols can route DIS PDUs without requiring special, dedicated networks. Multicast routing protocols could be used to route DIS traffic across multiple LANs without interrupting or adversely impacting other non-simulation hosts, or other simulation hosts which are not joined to the same multicast group address. Experiment #1 showed that a distributed simulation should be able to produce large numbers of DIS PDUs and not impact local, non-simulation network nodes by using multicast addressing. These are perhaps the first benefits that would be gained by converting broadcast-based simulations to use multicast addressing. The results from the first experiment are shown in Figure 8 and 9.

5.1.2 The Analysis

As noted above, each simulator was started manually. Therefore it is assumed that some relatively small number of DIS PDUs were initially lost from some of the host's log files. This is due to the fact that some simulation applications may not have finished their initialization procedures prior to the transmission of the first DIS PDUs from the other simulation applications which had completed their initialization. This was an unacceptable poor simulation. It could have been avoided by installing longer delays in the beginning of the CGF Testbed startup scripts, or by using a scripted trigger to initiate the start of a simulation event. Unfortunately, time ran out before another round of experimentation could proceed.

The results of the dropped datagram totals shown in Figure 8 indicate that the two SGI machines fared much better in the dropped-datagram category than did the two Sun workstations. The two graphs at the top of Figure 8 indicate the number of DIS PDUs (5284) sent through subnets E6 and E9. Each graph displays the number of network frames logged for each node on each Ethernet segment. From the graphs labeled E6 and E9, it is clear that an equal number of Ethernet frames were seen on both network segments. During the approximately 2 minute long simulation run, this amount of network traffic (load) accounts for approximately .7% of network bandwidth utilization.

By looking at the graphs for Sun-1 and Sun-2, a marked decrease in PDU transmission can be observed. By studying the transmission and reception logs for these two machines, it appears that the problem occurred within the simulation application itself. Specifically, the simulation event loop could not process all of the inbound IP datagrams and provide proper update rates for the locally managed entities. Also, the executive loop processed incoming traffic when it was available, even to the exclusion of managing the resources of the resident entities.

Since the simulation application was busy processing inbound PDUs, very little process cycle time was left to manage the rest of the simulation and therefore a marked decrease in the number of transmitted PDUs resulted. In other words, the Sun Sparcstations were severely overloaded at less then 1% of network bandwidth utilization. Note that this is in stark contrast to the general community belief that simulation applications are network bandwidth limited.

The SGI machines were capable of transmitting and receiving over 4,899 DIS PDUs during the course of this 2 minute exercise, whereas the two Sun Sparcstation-1's could barely produce 385 PDUs. Sun-1 shows almost a 20x decrease in PDU transmission performance when compared to the results of experiment #2. Sun-2 exhibited similarly poor performance characteristics with nearly a 10x decrease in PDU transmission.

At first glance, this discrepancy was attributed to a lack of available network bandwidth. However, as mentioned above, 5284 DIS PDU datagrams consumed over an approximately 2-minute long exercise duration indicates that approximately 44 network datagrams were being processed each second. That was a very low network bandwidth utilization rate and equates to approximately .633% of the theoretical maximum available.

Testing on the TRIDIS flooder (TRIDIS CDRL A00P) indicated that a maximum network utilization rate of 92.15% Entity State PDUs was obtainable. This corresponded to a maximum transmission rate of 5,808 Entity State PDUs per second over uncontested Ethernet. This places the network bandwidth utilization for experiment #1 at approximately .69% (100% efficiency) or .75% of the maximum recorded value (5808/sec.). The conclusion here is that this particular simulation application is not even close to being limited by network-bandwidth. Furthermore, looking at the number of datagrams transmitted by the two SGI's, each of the two Sun machines showed that entity interaction was at a minimum. In short, the Sun Sparcstations were thrashing[18].

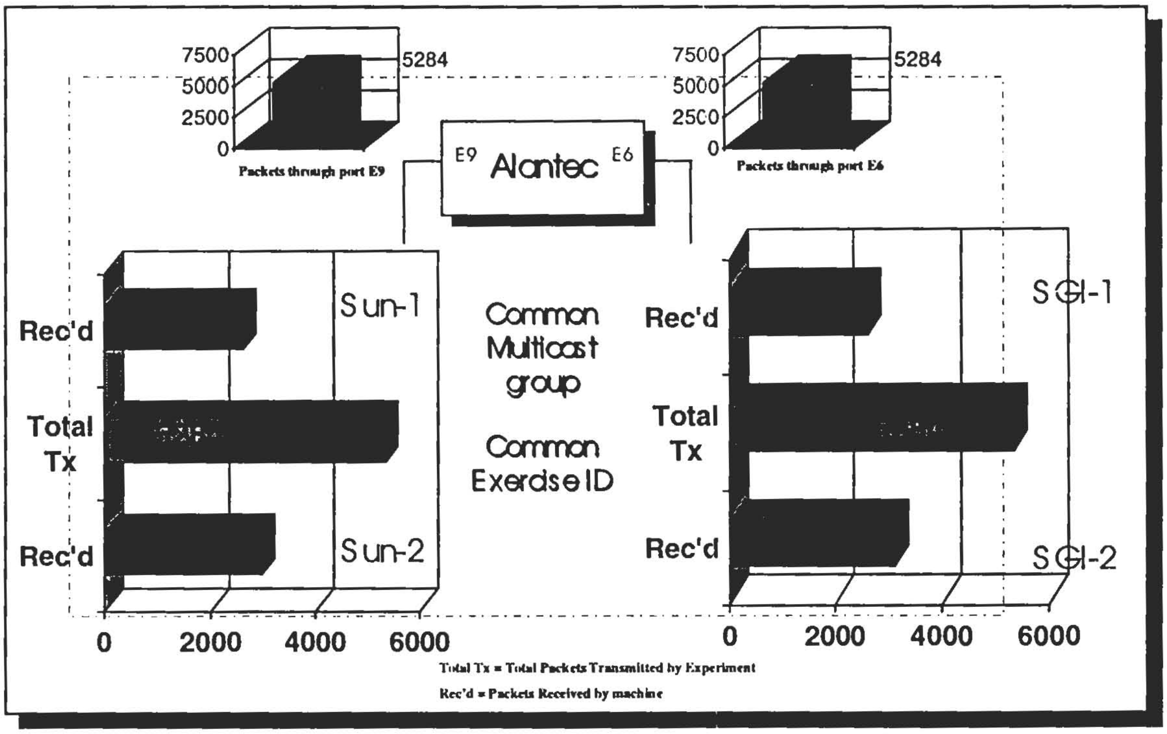

Figure 9 - PDU's transmitted and received in Experiment #1.

[18] Thrashing is characterized by a collapse in performance that occurs when memory (in a multiprogramming environment) or parts of memory become overcommitted. [Denning,86]

5.1.3 Conclusion

The results that were logged during experiment #1 generally supported what was expected. Each simulation host received DIS PDUs transmitted from the other three simulation hosts. As shown in Figure 9, the reception of DIS PDUs appeared to be balanced. However, when evaluated on a machine by machine basis, this was indeed not the case. As many as half of the DIS PDUs generated by the SGI machines were not recorded by the Sparcstation-1-based simulation applications. This discrepancy could not be because the simulation applications were started or terminated at slightly different times. The number of PDUs that fell into this category were under 2% for the total simulation. The results showed what the team expected to see: all of the four machines in the experiment joined a common multicast group and were able to communicate with each other as if a common broadcast addressing scheme had been used.

Theoretically, the only difference between the results of simulation experiment #1 and a typical broadcast exercise should have been that traffic originating from Sun-1 or Sun-2 would not have been received by SGI-1 and SGI-2. That would be because the router would have filtered out all broadcast communications between the two routed ports. However, as noted above, the differences in the underlying communication architecture were not the reason why the Sun Sparcstations dropped over 40% of the inbound datagrams.

5.2 Results of Experiment #2

Experiment #2 was designed to show that a simulation exercise could be partitioned into two mutually exclusive simulation exercises using two different multicast group addresses. By selecting one of two different multicast group addresses for each simulation host, PDU filtering was effectively accomplished at the network layer. The results of this experiment are shown in Figure 10 and Figure 11 below.

5.2.1 Layout

As in experiment #1, two of the simulation applications were compiled for Sun Solaris v2.4 and two for SGI IRIX v5.3. In experiment #2, the machines were configured to use two multicast group addresses. The first multicast group comprised machines Sun-1 and Sun-2. Machines SGI-1 and SGI-2 comprised the second multicast group. Note that the two SGI machines were configured to join a different multicast group from the Sun machines. Consequently the resultant simulation was partitioned into two halves with the Sun's and SGI's being members in separate multicast groups and located across two routed subnets. By partitioning the address space into two multicast group addresses located on separate routed subnets (router ports E6 and E9), the router was able to filter multicast traffic generated by one multicast group from the other.

By partitioning the multicast address space into two groups, the resultant simulation showed a reduction in the number of DIS PDU datagrams being presented to each simulation host. Simultaneously, the total number DIS PDUs generated during the course of experiment #2 had increased to 8,218 (See Figure 11).

As mentioned above, the reduction in the number of DIS PDUs presented to each host was accomplished by segmenting the exercise into two halves and then having the router filter datagrams based on the different multicast group addresses. This effect is similar to changing the exercise ID while holding the multicast group address constant, as in experiment #4 and #5 below. However, as shown in Figure 4, one half of this exercise was run on the two Sun machines and was located on Ethernet segment E6. The other half of the exercise was located on the Ethernet segment E9. Since each multicast group address was seen only on one of the router ports (not both), no multicast traffic was routed between ports E6 and E9 as seen in experiment #1.

Consequently, the Sun machines did not receive any traffic produced by the two SGIs. Similarly, the two SGI machines did not receive any traffic generated by the two Sun machines. An effective 100% filter was created. By reducing the number of datagrams crossing the router, a corresponding reduction in network load was presented to each of the hosts on both network interfaces.

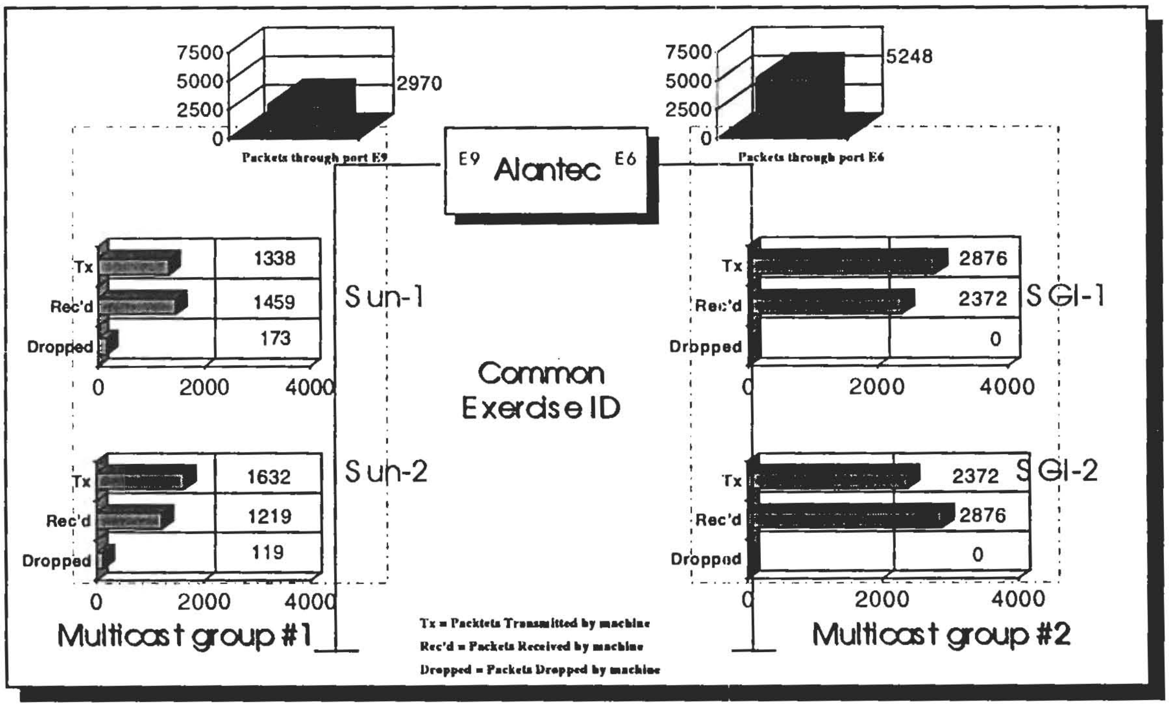

Figure 10 - PDU transmit and receive counts from experiment #2.

5.2.2 The Analysis

Using the configuration shown in Figure 4, experiment #2 produced the results shown in Figure 10 and 11. During a typical broadcast-based simulation exercise, all simulation nodes are supposed to be exposed to all transmitted PDUs. However, when that simulation exercise is segmented in such a way that the underlying network architecture can filter data from one or more segments, the change in network topology can change the results of the simulation application entirely. In general, changes to the underlying network architecture can and do affect the application layers.

The results in the graphs shown in Figure 10 illustrate how network level filtering reduced traffic flowing across the router. The IP datagrams transmitted by the Sun-1 and SGI-1 workstations were received by the Sun-2 and SGI-2 workstations, respectively. In other words, the two Sun machines and the two SGI machines exchanged datagrams only with each other. Note that the Sun machines exhibited some minor datagram loss, while the SGI machines did not drop any datagrams.

When comparing the total transmitted datagrams for each SGI machine with the total received by each SGI machine, the totals are equivalent. 2,876 datagrams were transmitted by SGI-1 and precisely 2,876 datagrams were received by SGI-2. This indicates that the router indeed filtered out all other network traffic. The 1,219 datagrams logged by Sun-2 equals the number of datagrams sent by Sun-1 (1,338) less the 119 datagrams dropped by Sun-2. Similarly, the 1,632 datagrams sent by Sun-2 can be accounted for by the 1,459 datagrams received plus the 173 datagrams dropped by Sun-1. These results indicate that the SGI workstations received no datagrams from the Sun workstations, and verify that the router did correctiy filter the datagrams based on the multicast group address.

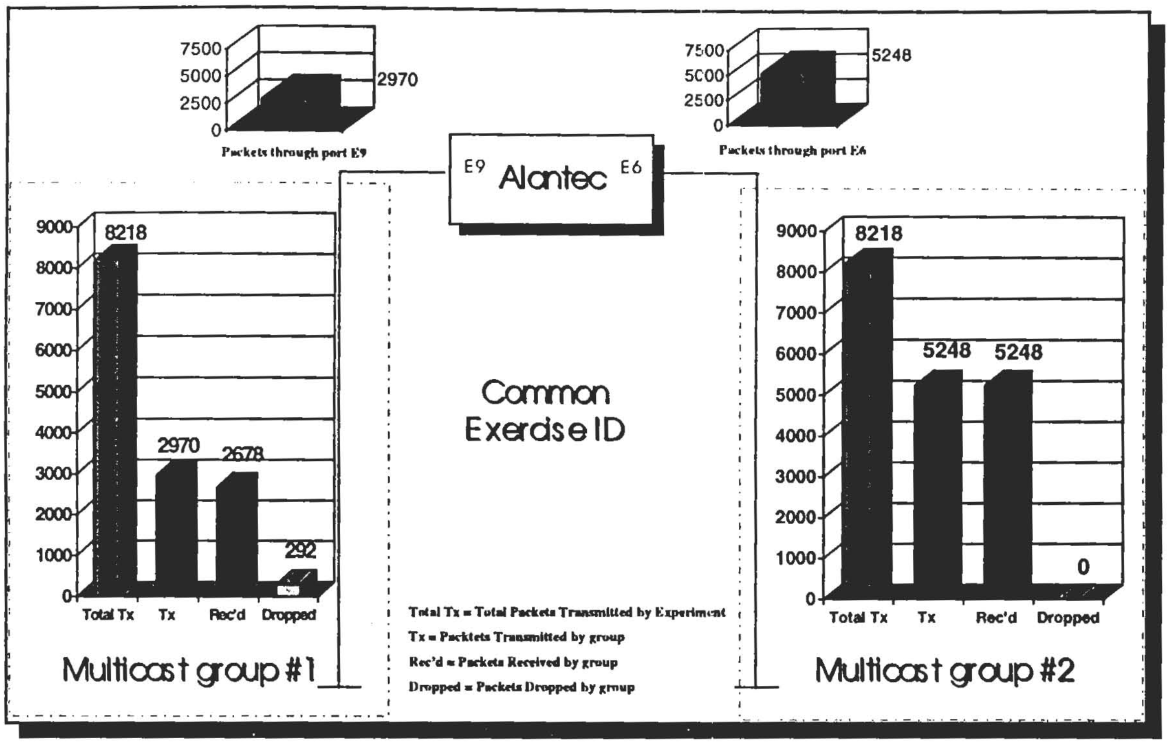

Figure 11 - PDU's transmitted and received in Experiment #2.

5.2.3 Conclusion

It is apparent when comparing the results in Figure 11 (above) with the results of experiment #1 in Figure 9 that the total number of datagrams transmitted by the two Sun workstations in experiment #2 (2,970 datagrams) was roughly an order of magnitude greater than the total number of datagrams transmitted by the Sun workstations in exercise #1 (385 datagrams). In addition, the number of datagrams dropped by either Sun machine dropped by more than an order of magnitude (4,711 and 292 for experiments #1 and #2 respectively). This disparity in the number of datagrams transmitted by either Sun-1 or Sun-2 in experiment #2 (from experiment #1) can only be attributed to a reduction in processor load induced by network traffic. In other words, the simulation application results can change based on external influences such as the underlying network interface.

5.3 Results of Experiment #3

Experiment #3 was designed to show that network level filtering can also take place at the workstation's network interface, not just at the network router interface. The same set of simulation applications were run for experiment #3 as for experiment #2. The address space was divided into two multicast group addresses with two machines being joined to each address group.

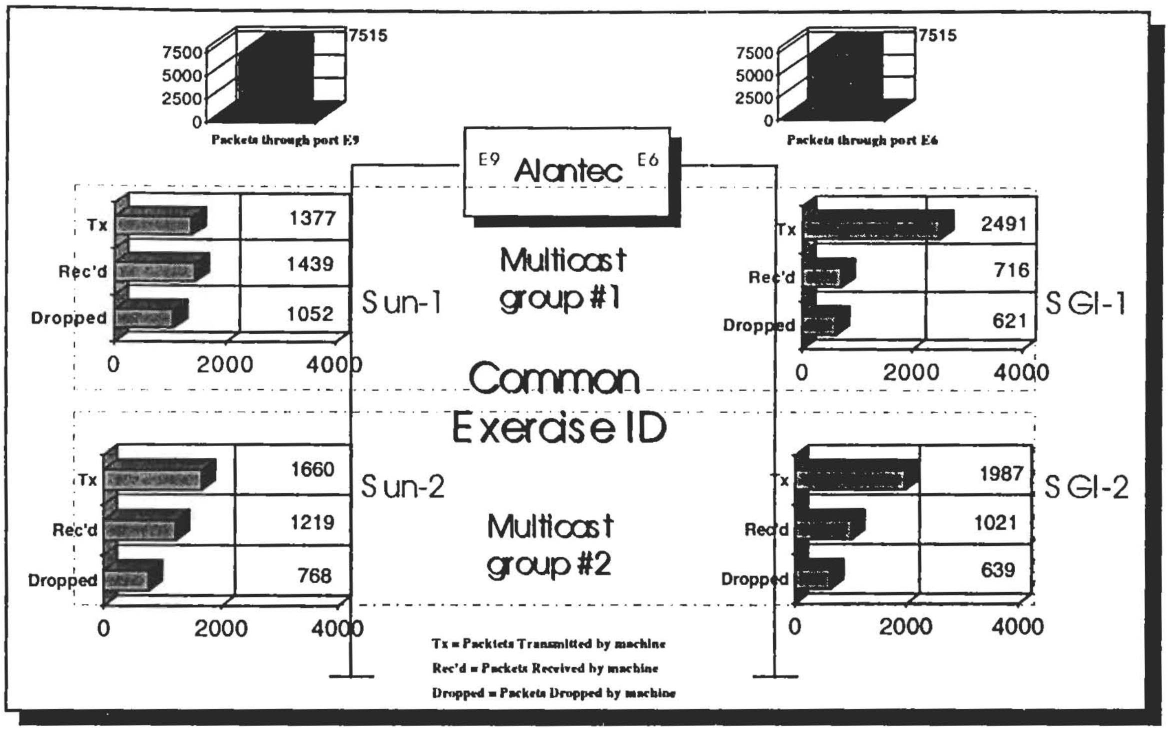

Figure 12 - PDU transmit and receive counts from Experiment #3.

5.3.1 Layout

Experiment #3 differs from experiment #2 in that the machines selected for the first multicast group were Sun-1 and SGI-1. Sun-2 and SGI-2 comprised the second multicast group. This address re-assignment allowed the router to pass multicast group addressed traffic to both routed subnets because each multicast group address was registered on both ports (E6 and E9) of the router. Network traffic results collected for experiment #3 can be seen in Figure 12 below.

5.3.2 The Analysis

The data collected from the four simulation hosts during experiment #3 indicate that all four of the simulation applications lost a significant number of packets. Each application's log file shows that no fewer then 621 datagrams were dropped by each simulation application. Yet unlike experiment #1, the two Sun workstations did not exhibit any abnormal simulation behavior. In addition, the statistics collected from the router during this exercise indicated that a total of 37 IPmc datagrams were dropped.

When analyzing the data files produced by each simulator in experiment #3, a large disparity was noted in the number of datagrams received when compared to the number of datagrams sent. Since experiment #3 was to be a simple reconfiguration of the underlying network architecture, no changes were expected to occur at the application layer. However, the number of missing datagrams was significant: 44.28% for multicast group #1, and 40.49% for multicast group #2. In addition to the large drop rate, network loading was at only 1.1% for 62 datagrams per second.

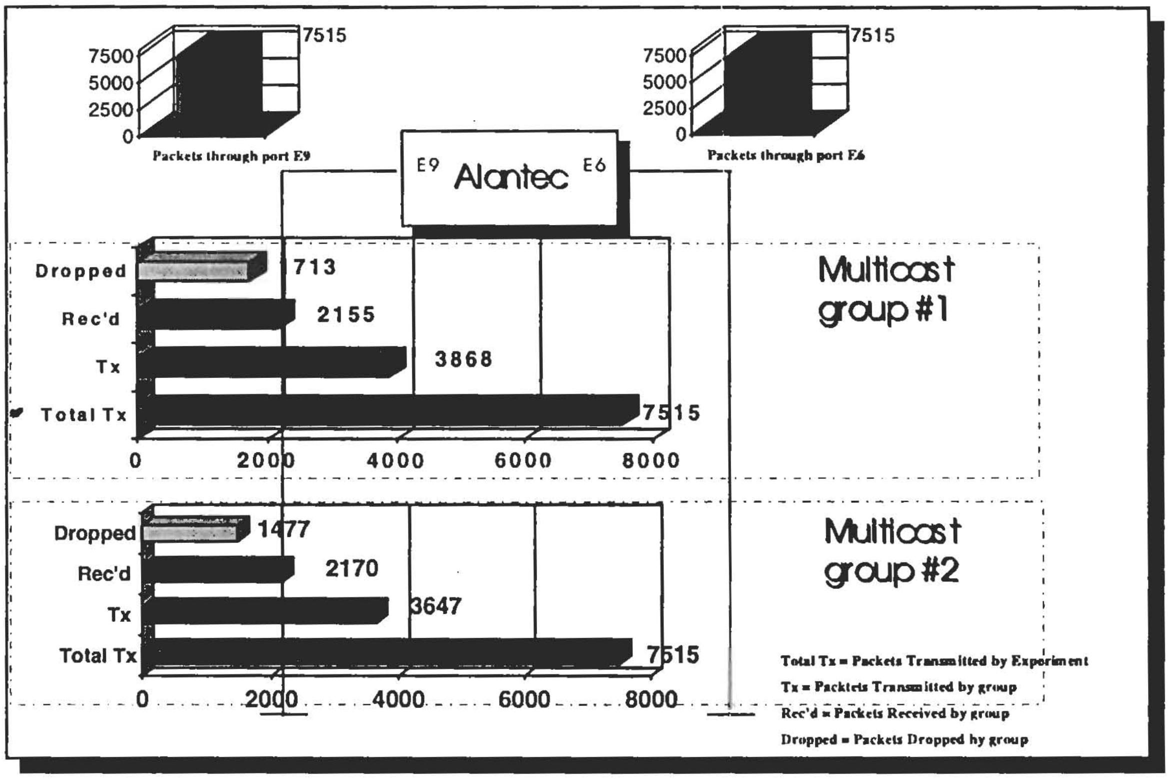

Figure 13 - PDU's transmitted and received in experiment #3.

During the course of the experiment, no other network related problems were apparent, however, it is possible that other traffic passing through the router's backplane at the time of the experiment might have had an adverse affect on the results of this particular simulation exercise. When the router's own diagnostic statistics were examined, only 37 datagrams were documented as having been dropped and no fragmented, jammed or runt datagrams were detected. In addition, the router's memory buffer did not appear to have overflowed. So at this time it appears unlikely that either the router or external multicast traffic could have affected this experiment.

Examination of the loading characteristics of the Sun workstations appears to reveal that neither of these machines became overloaded in experiment #3. As noted in experiment #1, both Sun workstations became severely overloaded when the entire 15 entity simulation was executed. Later, experiments #4 and #5 display symptoms (to varying degrees) of the performance overloading even though the simulation hosts were required to process a similar amount of IPmc datagrams. Experiment #3 provided performance results similar (relative to the other experiments) to those witnessed in experiment #2. These results are consistent between experiment #2 and experiment #3 because the experiments were of similar sized simulations with each containing (roughly) 8 entities per simulation host. Although some degradation was shown in the results of experiment #3, this is consistent with the new architectural configuration of one Sun and one SGI machine in each multicast group. In experiment #2, multicast groupings were comprised of similar machines.

5.3.3 Conclusion

The team hypothesizes that network traffic external to the IPmc data path (such as IP or FDDI traffic) could have caused the router's CPU to expend less time performing IPmc routing. Routing statistics were not collected from the data paths external to the multicast experiments so further analysis of this experiment could not be conducted to explain the high datagram drop rate. However, if the results of this are in error due to external influences, then the only recourse would be to re-run this experiment to determine if that was indeed the case. If external influences were found to have occurred during this experiment, then potentially any or all of the other IPmc experiments may have been compromised as well.

5.4 Results of Experiment #4

Experiment #4 was designed to show that there is a difference between network level filtering (as in experiments #2 and #3) and application level filtering (as in experiments #4 and #5). In experiment #4, a common multicast group address was chosen for all four simulators participating in the exercise. However, in experiment #4, two different exercise ID's were selected. By having two different exercise ID's embedded within the PDU header of each IPmc datagram, the simulation application would have to first collect the network datagram, decode the IP header, decode the UDP header contained therein, and then pass the data payload containing the DIS PDU up through the socket interface to the application. The simulation application then would be responsible for decoding the DIS PDU header and in particular, the exercise ID and determining whether or not the rest of the data buffer warranted further processing by the resident simulation application.

5.4.1 Layout

Experiment #4 can be readily compared with experiment #2 in both size and layout. The only difference between experiment #2 and experiment #4 was that the multicast group address was held constant while two different exercise IDs were used in experiment #4. The two Sun machines were chosen to be in exercise ID group #1 while the two SGI machines were placed in exercise ID group #2. Experiment #4 can also be readily compared to experiment #5 as both experiments generated roughly the same total number of datagrams per multicast group.

In experiment #4, each machine was forced to read and evaluate the PDUs for both exercise IDs. This is unlike the results for experiment #2 where different multicast addresses were used. In experiment #2, the network layer was responsible for filtering the other "exercise" from each simulation application. In experiment #4, each simulation application was required to read every inbound IPmc datagram and decode the embedded PDU. The simulation application would then examine and evaluate the PDU header for the correct exercise ID. Inbound IPmc datagrams which contained incorrect exercise IDs for the given simulation application were summarily discarded.

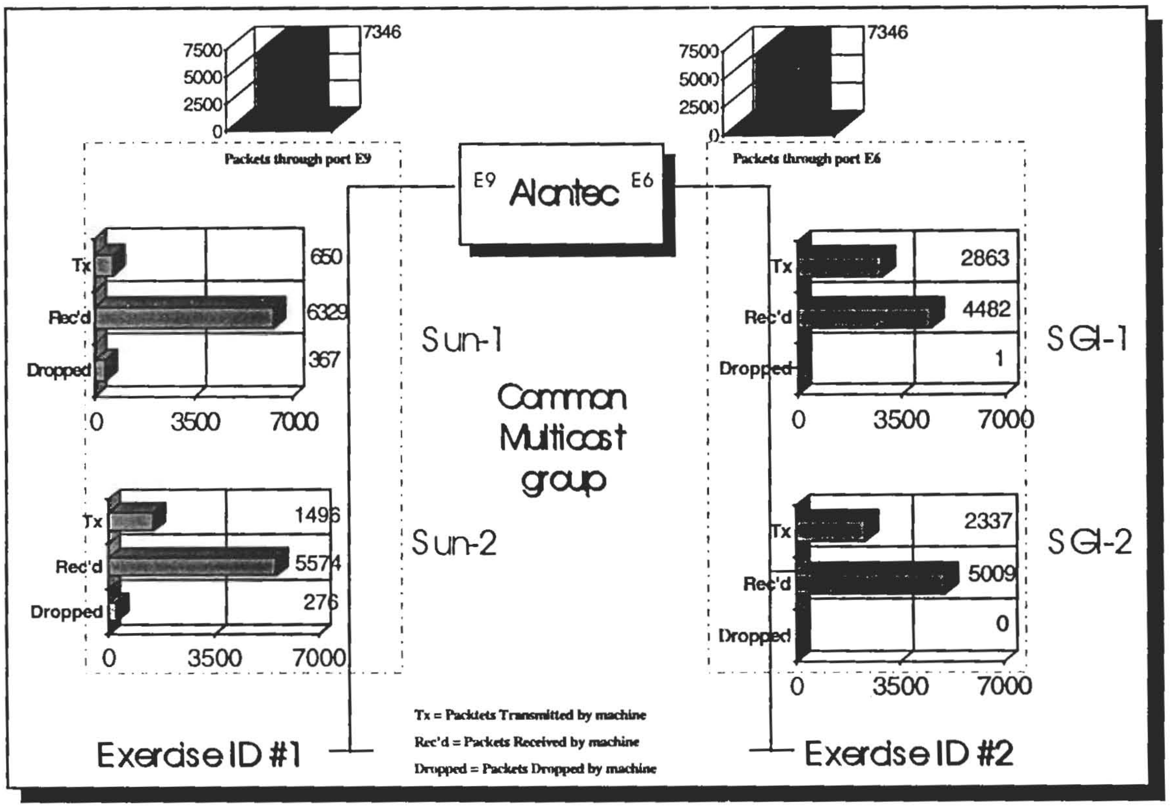

Figure 14 - PDU transmit and receive counts from experiment #4.

5.4.2 The Analysis

In the data produced by experiment #4, workstation Sun-1 showed a marked decrease in performance (approximately 50%), as exhibited by the drop in DIS PDUs transmitted. Workstation Sun-2 exhibited normal DIS PDU production counts with nearly 1,500 PDUs being produced. Both Sun workstations showed that some datagrams were lost during the course of the simulation. This seems to be consistent with the rest of the simulation exercises where most datagram drops occurred on the Sun workstations[19].

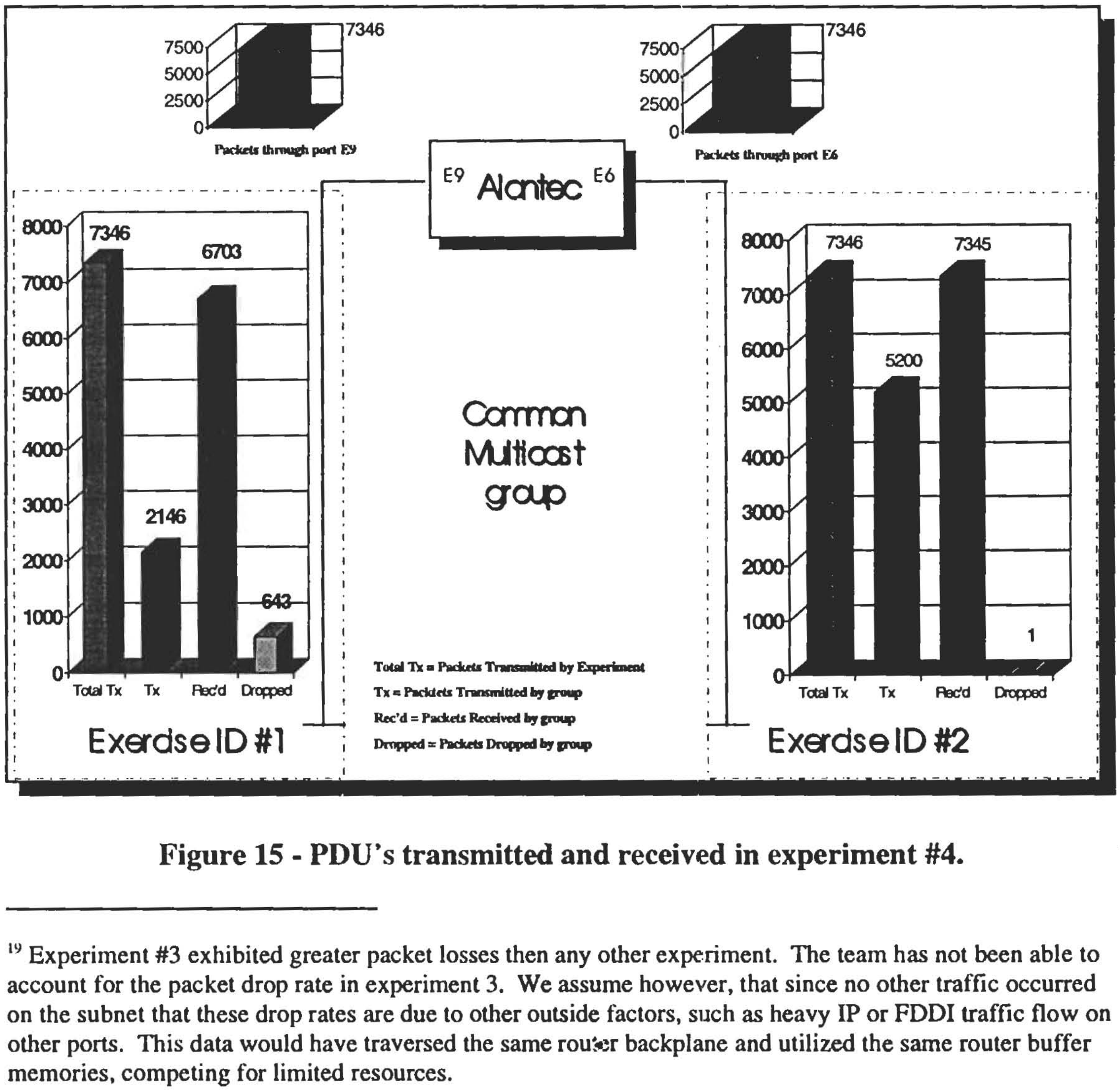

Figure 15 - PDU's transmitted and received in experiment #4.

[19] Experiment #3 exhibited greater packet losses then any other experiment. The team has not been able to account for the packet drop rate in experiment 3. We assume however, that since no other traffic occurred on the subnet that these drop rates are due to other outside factors, such as heavy IP or FDDI traffic flow on other ports. This data would have traversed the same router backplane and utilized the same router buffer memories, competing for limited resources.

The TRIDIS team also noted that both SGI workstations' maintained a satisfactory output rate of DIS PDUs, as shown in Figure 14. The transmission columns in Figure 15 clearly shows the performance differential between the Sun and SGI workstations. In this exercise, all four workstations were exposed to relatively equivalent quantities of DIS PDU datagrams. In Figure 15, the graph on the left indicates the traffic flow as seen on the Sun side of the router. The second column shows the number of datagrams transmitted by the Sun machines participating in Exercise #1. Conversely, on the graph on the right, the second column shows the number of DIS PDU's transmitted by the SGI machines in Exercise #2.

The third column in both graphs in Figure 15 shows the number of DIS PDU's received by each exercise ID. The received column in the chart is significant in that it shows how the number of DIS PDUs being transmitted by either exercise impacted the network interfaces on the local hosts participating in the other exercise.

In a typical interactive simulation exercise which employs a single multicast address, the number of DIS PDUs transmitted should equal the number of DIS PDUs received. This is the case with experiment #2. Even though the segmentation is separated across the subnet boundary, machines participating in multicast group #1 do not receive inbound network traffic from the machines in multicast group #2 (see Figure 11, above). However, according to the data shown in Figure 16 below, the Sun machines (participating in exercise #1) processed almost[20] as many inbound IPmc datagrams as the SGI machines (participating in exercise #2) during experiment #4.

[20] The numbers would have been equivalent had the Sun machines in exercise #1 not dropped 643 packets.

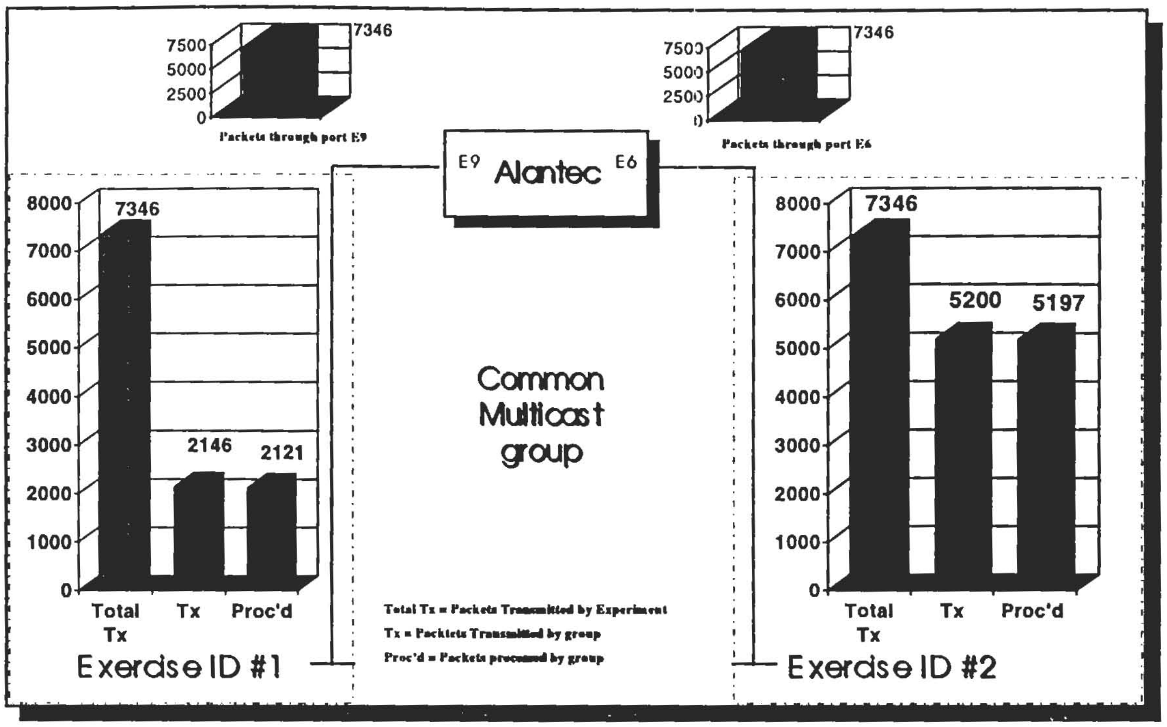

Figure 16 - PDU's transmitted and processed in experiment #4.

Examining the quantity of network datagrams received by each machine or by each exercise ID does not necessarily equate to the amount of work performed by each simulation application. Figure 16 above shows this relationship by comparing the number of datagrams transmitted, received and processed by the simulation applications. Figure 15 showed the relationship between outbound and inbound network datagrams. The difference between these two charts is the ratio of the amount of work performed filtering PDUs by the simulation applications in their respective exercises.

5.4.3 Conclusion

Figure 16 shows a clear difference in performance between the SGI workstations and the older Sun Sparcstations. The SGI workstations in experiment #4, exercise #2 produced better than 70% of the datagrams generated in this simulation exercise. The Sun machines in experiment #4, exercise #1 generated just under 29% of the DIS PDUs in this simulation exercise and even at this reduced rate, still exhibited performance related problems on PDU output.

5.5 Results of Experiment #5

Experiment #5 was designed to highlight the difference between network level filtering (as in experiments #2 and #3) and application level filtering (as in experiments #4 and #5). Experiment #5 used a common multicast group address and two different exercise ID's. The configuration for experiment #5 was such that at least one machine in each exercise ID group was located on each router port (E6 and E9). This configuration provides the basis from which to compare experiment #3, which used multicast group addresses to segment the simulation exercise. And just like in experiment #4, having two different exercise ID's embedded within the PDU header (of each IPmc datagram) required the simulation application to collect the datagram, decode the IP and UDP headers, then pass the DIS data payload through the socket interface to the application for final evaluation.

5.5.1 Layout

Unlike experiment #4, experiment #5 joined machines in a common exercise on opposite ports of the router. In experiment #5, each simulation host's outbound datagram was received by all other simulation hosts. Because different exercise IDs were used, their evaluation was made by the simulation application. Evaluation of the Ethernet frame and DIS PDU header was required before filtering on exercise ID took place.

Experiment #5 can be readily compared with experiment #3 in both size and layout. The only difference between experiment #3 and experiment #5 was that the multicast group address was held constant while two different exercise IDs were used. In experiment #5, the one Sun and one SGI machine were chosen to be in exercise ID group #1 while the remaining Sun and SGI machines were placed in exercise ID group #2. Experiment #5 can also be readily compared to experiment #4 as both experiments #4 and #5 transmitted and routed roughly the same total number of datagrams per multicast group and per exercise ID.

5.5.2 The Analysis

Like experiment #4, each simulation host in experiment #5 was required to read and evaluate the DIS PDUs containing both exercise IDs. The results from experiment #5 differ from the results collected by experiment #3 in that there were no large number of dropped or lost datagrams. In experiment #3, the network layer was responsible for filtering the "other exercise" from each simulation application. In experiment #5, each simulation application's network transport interface was required to read every inbound IPmc datagram and decode the embedded PDU. The resultant PDU was then passed up to the simulation application where the PDU header would be examined. The exercise ID field in the PDU header was evaluated to determine if the PDU datagram (still in buffered memory) contained information for the resident simulation application. If the PDU contained the correct exercise ID for the simulation application, then the simulation application continued to process the input buffer. If the exercise ID was incorrect for the resident simulation application, then the PDU datagram was discarded and the buffered memory was reclaimed.

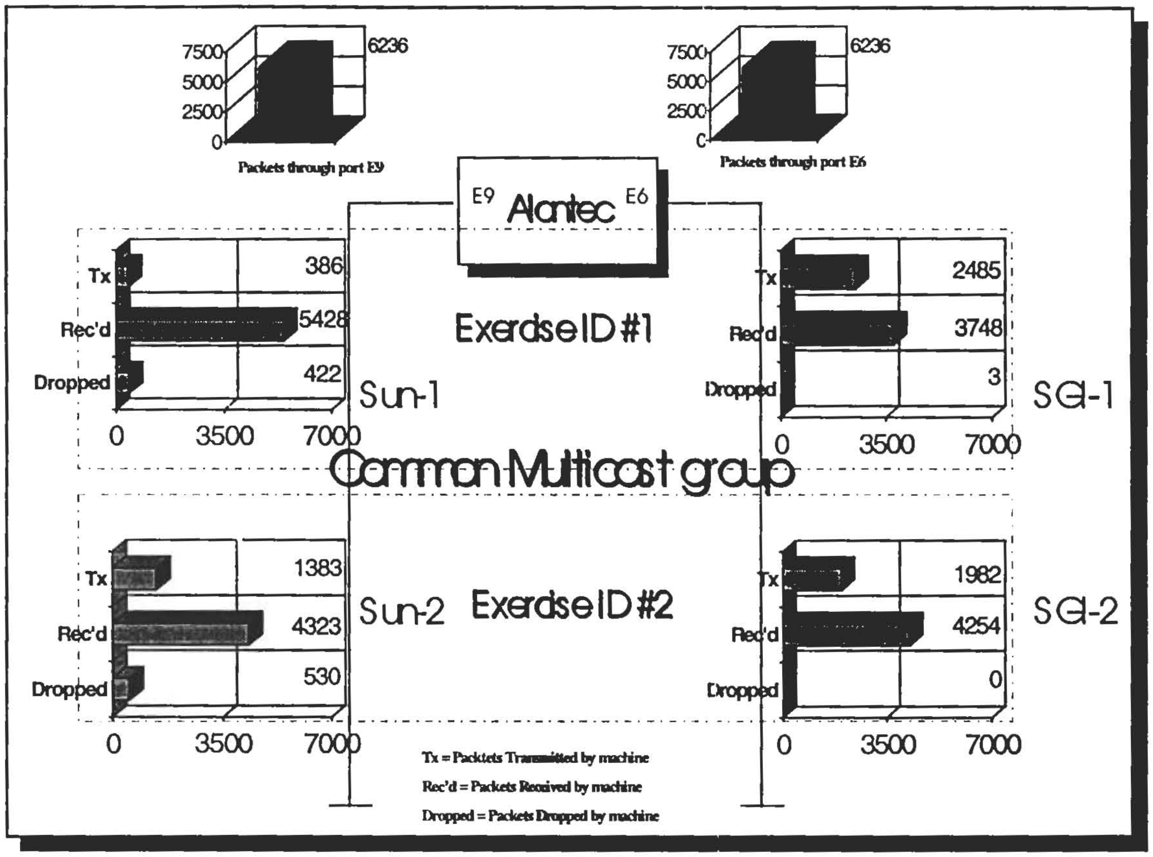

Figure 17 - PDU transmit and receive counts from experiment #5

In general, any inbound IPmc payload which contained incorrect exercise IDs for a given simulation application was discarded. This process of read, evaluate, and discard is not insignificant when hundred or even thousands of DIS PDUs are received every second. Embedding different exercise IDs within the PDU header of each IPmc datagram moves the responsibility of exercise filtering from the lower level network interface, to the higher level application program where additional work is required to move the inbound datagram up through the transport layer and into the application. The amount of additional work required varies from platform to platform. The amount of additional work therefore, is proportional to the quantity and arrival time of inbound network datagrams. During a simulation exercise, the results of this impact are first visible on the machines exhibiting lower overall machine performance, as the performance results of machine Sun-1 in Figure 17 shows.

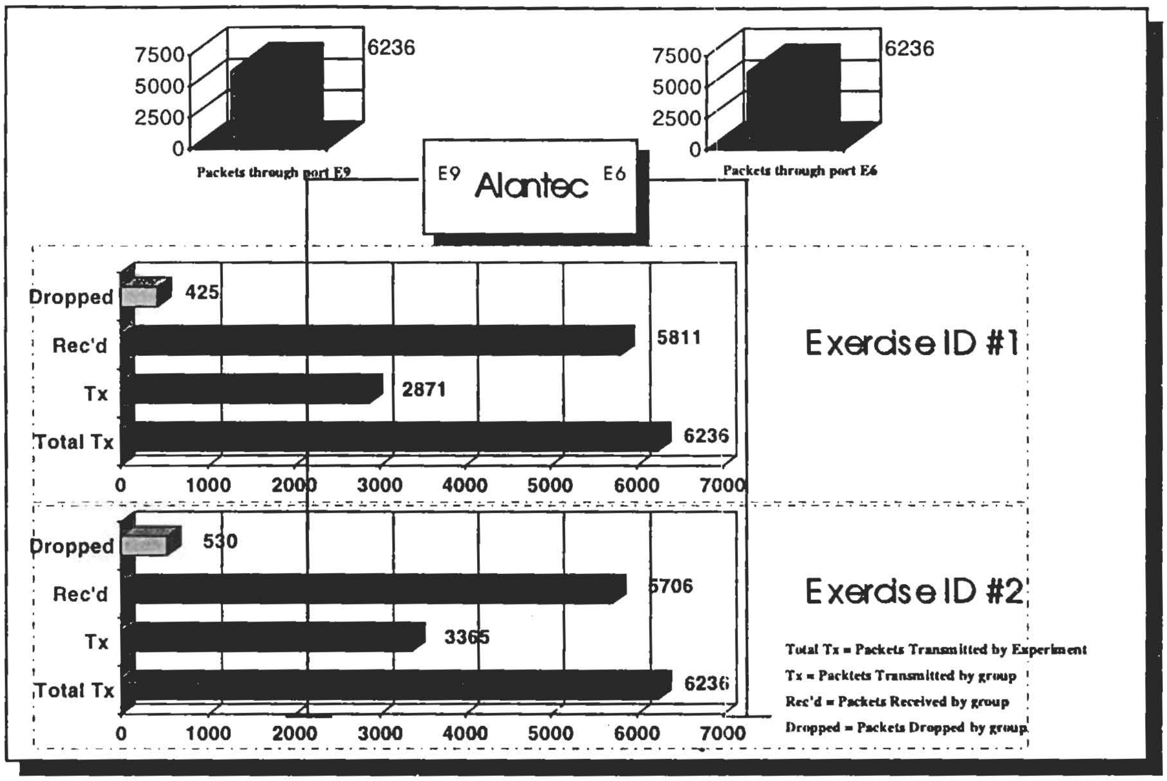

Figure 18 - PDU's transmitted and received in experiment #5.

Figure 18 shows the relationship of transmit and receive PDU's on a per exercise ID basis. This graph was generated by combining the data collected from the simulation applications engaged in the same exercise. This graph is significant in that it presents results identical to those produced by a simulation network data logger. The results presented in Figure 18 would appear to indicate that two otherwise healthy simulation exercises occurred concurrently over the same communication infrastructure without interfering with one another. However, these results are misleading.

Without comparing the transmission and reception results for each simulation host, the data presented in Figure 18 shows that both exercises transmitted and received roughly the same number of DIS PDUs. The data also indicates that both exercises dropped the same number of PDUs and were exposed to the same number of Ethernet frames (whether or not those frames were targeted for the resident simulation application). In other words, without examining each individual simulator's response, it is impossible to determine whether a given simulation application produced valid results from the inbound data stream. In other words, network level monitoring can not accurately provide performance metrics, either good or bad, for simulation exercises. Instead, a closer look at each simulation application is required.

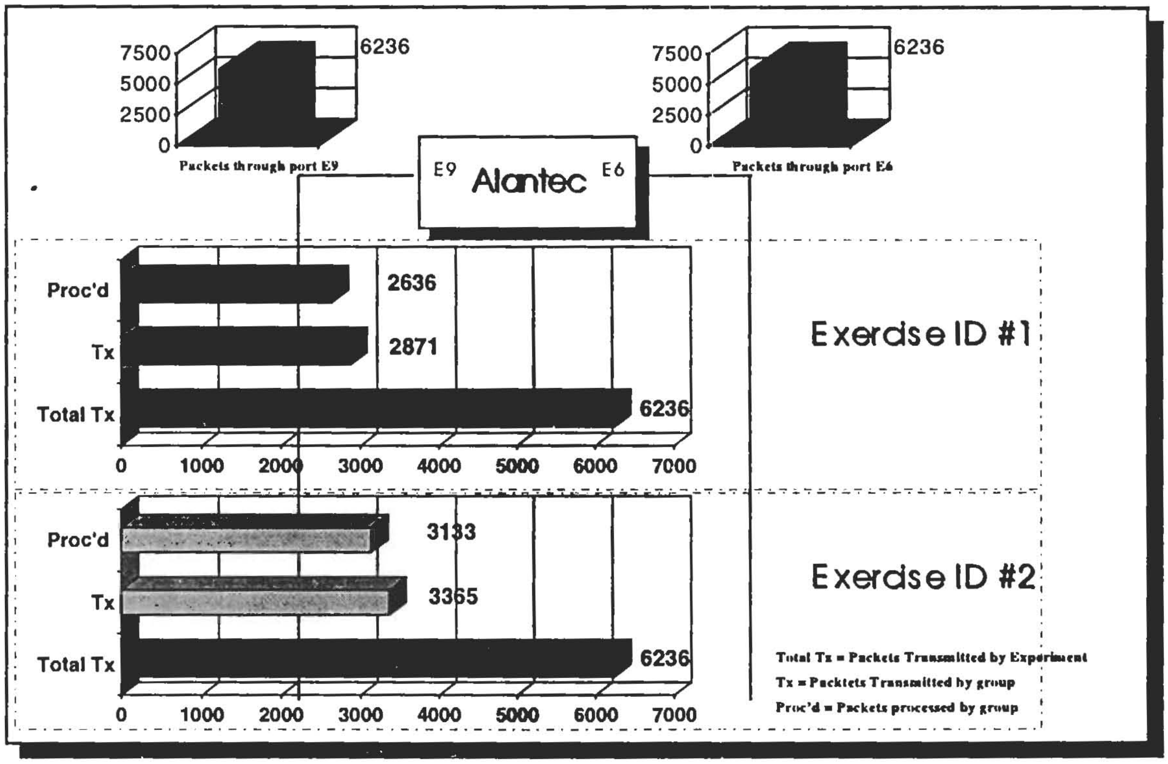

Figure 19 - PDU's transmitted and processed in experiment #5.

5.5.3 Conclusion

As in previous exercises, several hundred datagrams were dropped by the two Sun workstations. Apparently the performance of these two Sun Sparcstation-1 machines was such that they were unable to receive and process inbound network traffic while simultaneously running the CGF simulation. In experiment #5, these machines exhibited low performance output while network bandwidth utilization was about .74%. This equates to an inbound datagram rate of 52 Entity State PDUs per second. Again, the formal per-exercise presentation shown in Figure 19 does not show this degradation in the distributed simulation.

According to Figure 19, performance degradation is slight to non-existent, showing only a slight decrease in overall simulation performance as measured in the total number of PDU's transmitted per exercise. Figure 19 does not indicate the number of datagrams received or transmitted on a machine-by-machine basis, so the performance characteristics of the Sun-1 and Sun-2 machines is masked by the performance of the SGI machines. Since Figure 17 showed a marked decrease in the number of outbound IPmc datagrams generated by "half" of the machines participating in the exercise, it is fair to assume that this exercise produced what might be considered an invalid simulation.

5.6 Experiment Timing Comparisons

The modeling and simulation community has known for several years that the validity of a simulation decreases with a rise in network traffic. The more entities that are being modeled causes an increase in network traffic. As more entities are placed into the simulation environment, the assumption was that more entity-to-entity interaction would generate more DIS PDUs. This growth would then continue until such time when the network layer could no longer supply the necessary bandwidth to implement a dynamic, real-time simulation. The belief was that distributed simulations were network bandwidth limited and not processor limited.

The second theme in the Modeling & Simulation community is that if a given simulation exercise can be segmented or divided in such a way that each simulation application has a reduced set of DIS PDUs to respond to, then the entire simulation exercise can be down-scaled, reducing the network traffic and thereby improving performance. The premise for this statement was that the network was at fault and in particular has been network bandwidth was the sole limiting factor. Yet no experimental evidence has been presented to confirm or deny this.

As our experimentation has shown, the simulation application and the performance of the host platform may have a lot more to do with how well a distributed, real-time simulation operates than does a limitation on network bandwidth. To prove this statement, the TRIDIS team designed and performed these four IPmc experiments to segment the total number of entities modeled within the simulation exercise. Three types of segmentation were selected: No segmentation, network layer segmentation, and application layer segmentation.

Experiment #1 demonstrated an exercise containing a fixed number of entities (no-segmentation) and demonstrated CPU performance related problems on slower Sun Sparcstation-1 architectures. Experiments #2 and #3 then segmented the simulation at the network layer. Each group was networked together either locally or through routed ports. Experiments #4 and #5 also segmented the simulation based on exercise ID.

The multicast simulation experiments run were designed to highlight the benefits of network level filtering over application level filtering. The following eight (8) graphs show how filtering DIS PDUs at the appropriate level can effect a change in the overall simulation performance.

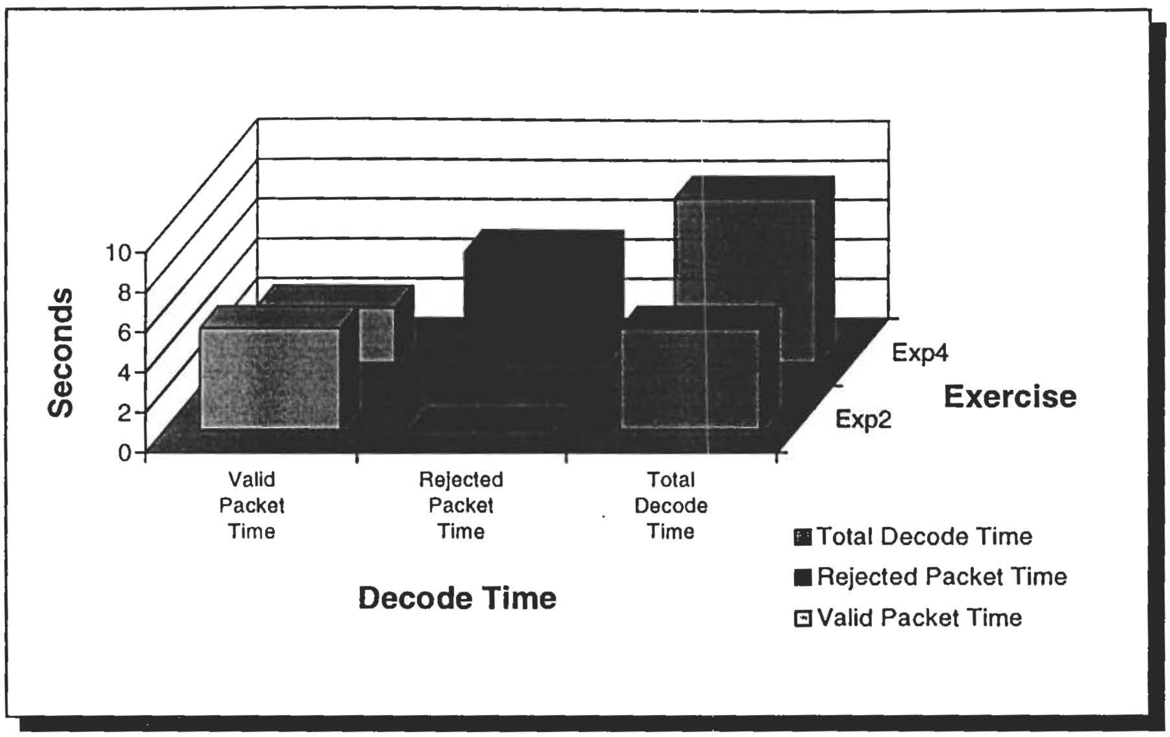

Figure 20 - Decode time comparisons of experiments #2 and #4 for Sun-1.

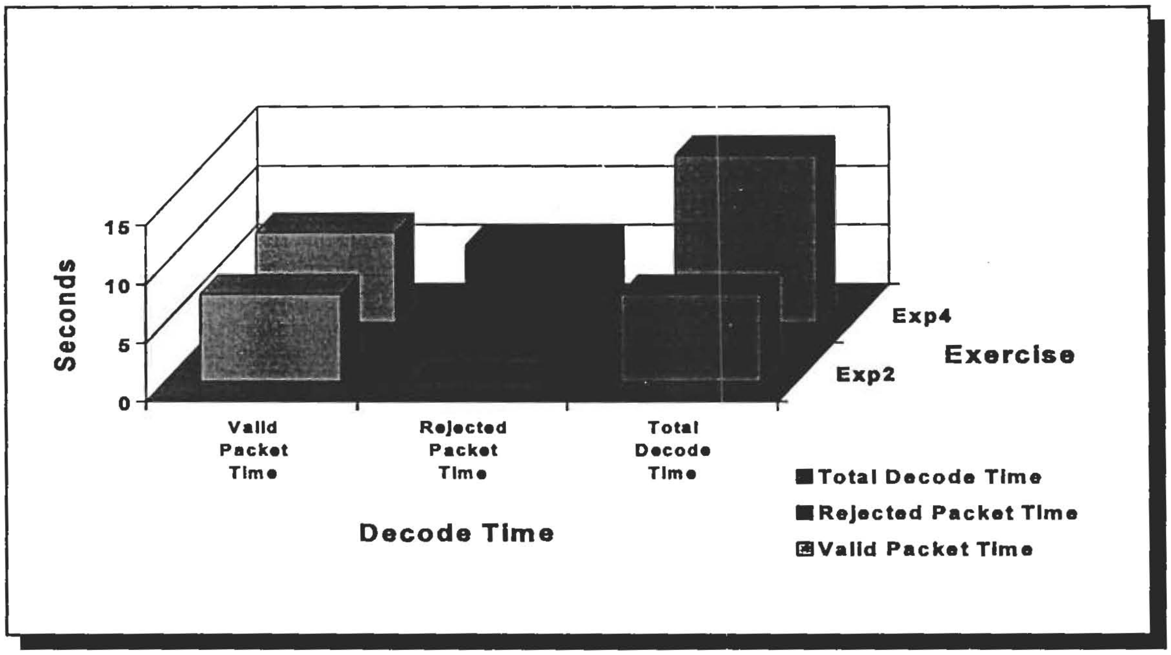

Figure 21 - Decode time comparisons of experiments #2 and #4 for Sun-2.

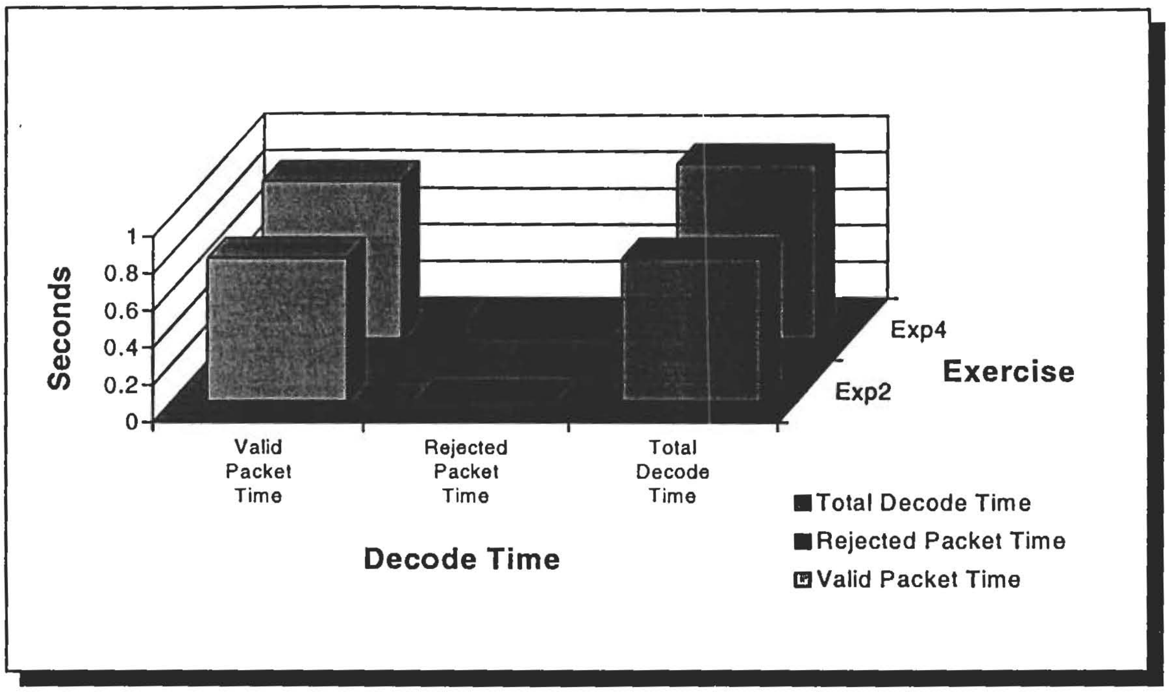

Figure 22 - Decode time comparisons of experiments #2 and #4 for SGI-1.

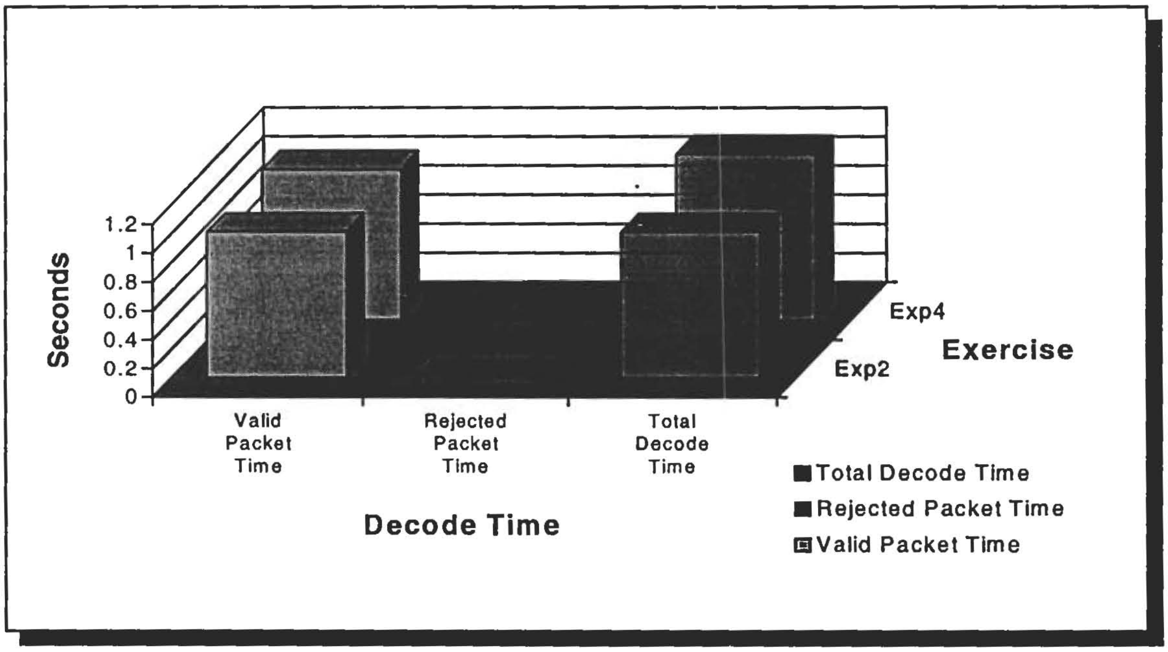

Figure 23 - Decode time comparisons of experiment #2 and #4 for SGI-2.

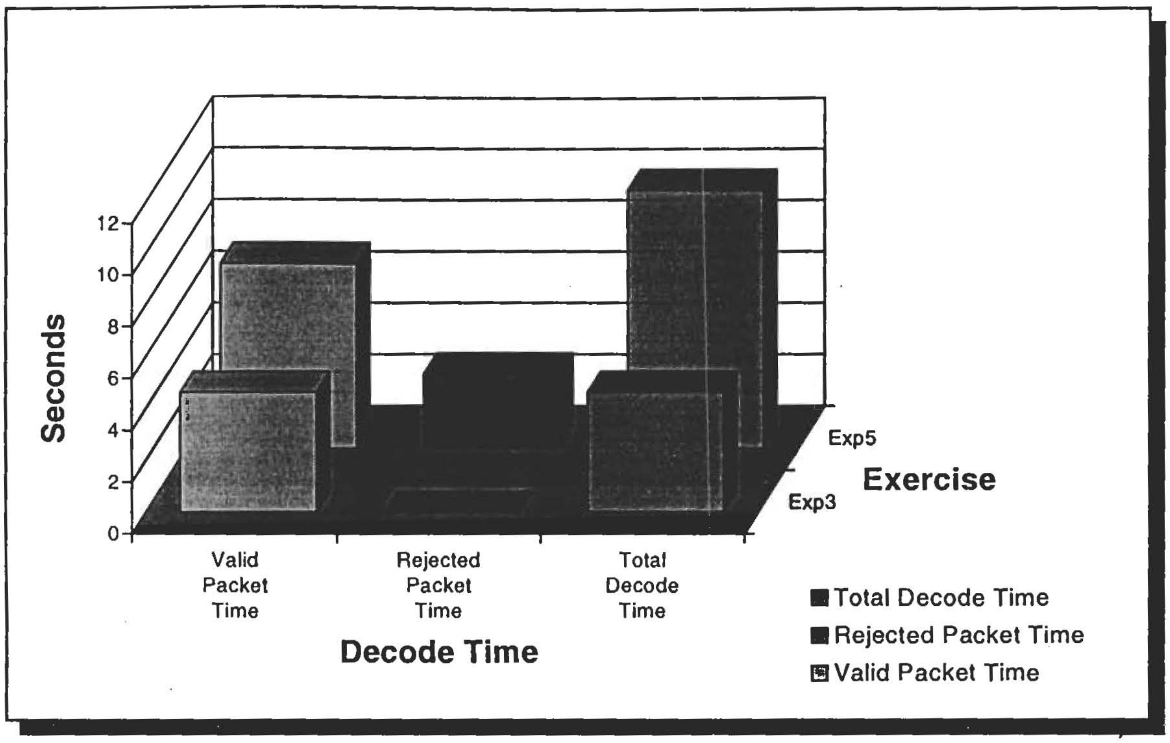

Figure 24 - Decode time comparisons of experiments #3 and #5 for Sun-1.

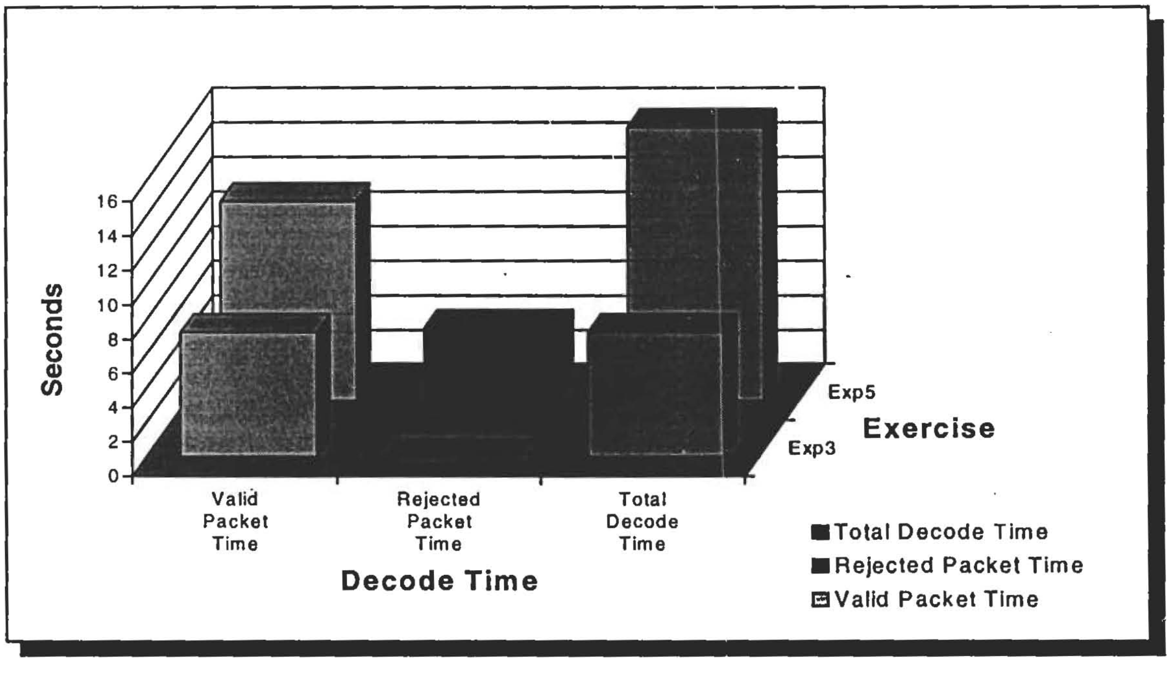

Figure 25 - Decode time comparisons of experiments #3 and #5 for Sun-2.

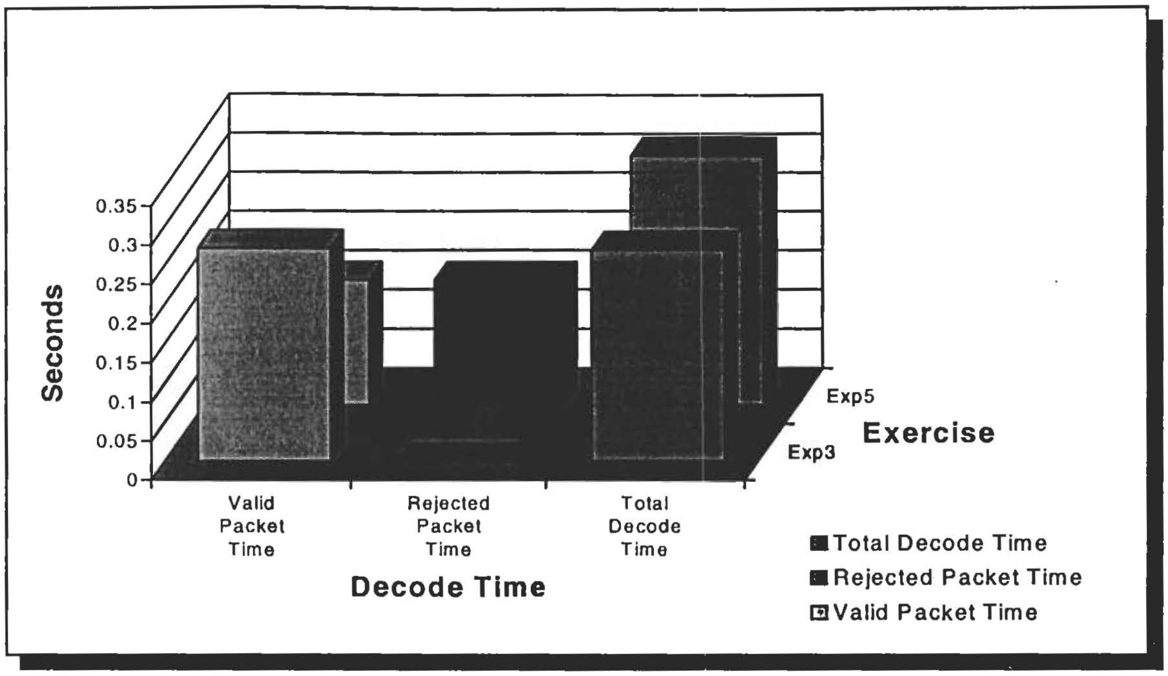

Figure 26 - Decode time comparisons of experiments #3 and #5 for SGI-1.

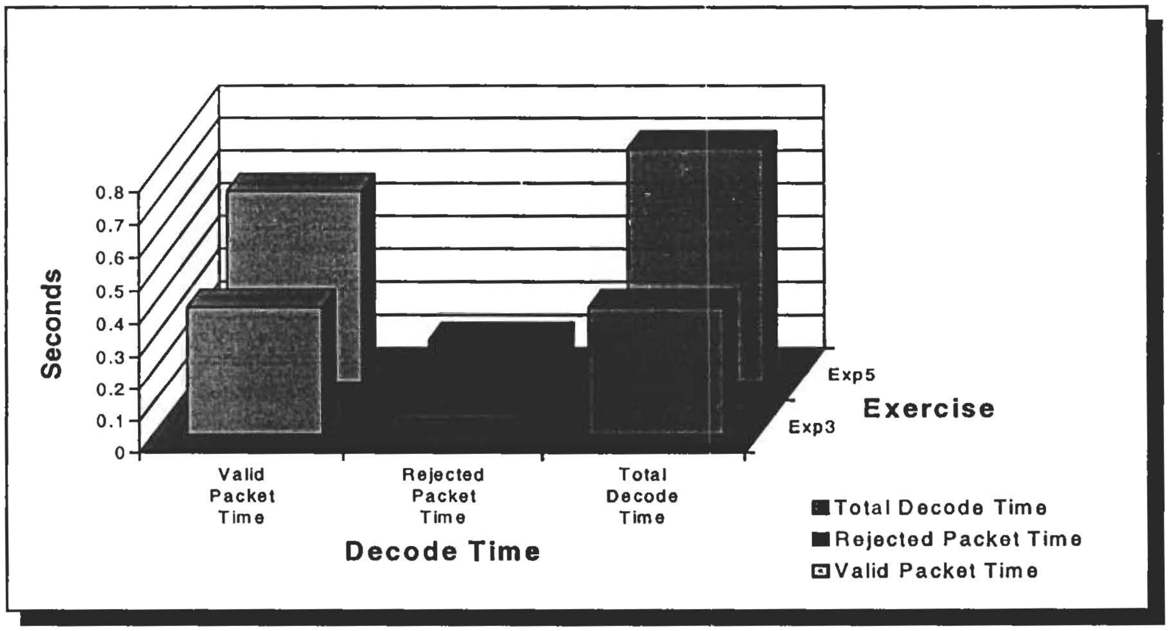

Figure 27 - Decode time comparisons of experiments #3 and #5 for SGI-2.

These figures show the sum total of raw CPU time taken by each simulation application to read and evaluate all inbound (received) DIS PDUs. The results from each simulation application for experiments #2 through #5 are represented in the previous charts. Experiments #2 and #4 are compared first followed by a comparison between experiments #3 and #5. In each example, the column on the left shows the time taken to decode valid (for the exercise, and where appropriate) datagrams. The center graph shows the amount of time taken by the application to evaluate and discard invalid datagrams. The column on the right shows the total time taken for the simulation application to read and evaluate all inbound network datagrams.

6. CONCLUSIONS

When taken collectively, the five multicast experiments conducted by the TRIDIS team indicated that there is a scalability problem inherent in DIS exercises which employ simulation hosts of varying performance levels. The effect is first noticed on the platform (i.e. simulation host) exhibiting the slowest performance response. For example, if five Pentium-based PCS are joined in a common simulation exercise along with an 80286-based PC, then the latter machine will exhibit performance degradation before the other five machines. In addition, the entire DIS exercise will suffer (i.e. become invalid for training purposes) if the performance demands placed on the an 80286-based PC reach beyond its capabilities. In other words, the performance requirements of a given distributed simulation exercise must be maintained at or below the level of the slowest performing machine in the distributed network.

Results from network traffic analysis indicated that the platforms used in these experiments displayed significantly different levels of performance. In some experiments, some machines simply could not handle the flow of incoming network traffic, even though network bandwidth utilization was around 1%. Only during experiment #2 (where inbound traffic flow was at its lowest due to network filtering) did the slowest machine produce an adequate output of DIS PDUs.

It is apparent when comparing the results of experiment #2 (Figure 11) with the results of experiment #1 (Figure 9) that the total number of datagrams transmitted (not received) by the two Sun workstations in experiment #2 (2,970 datagrams) is roughly an order of magnitude greater than the total number of datagrams transmitted by the Sun workstations in exercise #1 (385 datagrams). In addition, the number of datagrams dropped by the Suns, fell by more than an order of magnitude (4,711 and 292 for experiments #1 and #2 respectively). This disparity in the number of datagrams transmitted by either Sun-1 or Sun-2 in experiment #2 (from experiment #1) can only be attributed to a reduction in processor load induced by network traffic. In other words, the simulation application results can change based on external influences such as the underlying network interface.

Experiment #3 exhibited greater datagram losses then any other experiment. The team has not been able to account for the high datagram drop rate in experiment #3. No other traffic occurred on the subnet where these drop rates were recorded and are most likely due to other outside factors such as heavy IP or FDDI traffic flow on other ports. This data would have traversed the same router backplane and utilized the same router buffer memories, competing for limited resources. However, routing statistics were not collected from the data paths external to the multicast experiments so further analysis of this experiment cannot explain the high drop out rate. Therefore, it is the team's conclusion that experiment #3 should be re-run to eliminate any such discrepancies.

In experiment #4, Figure 16 shows that the difference in performance between the SGI workstations and the older Sun Sparcstations when application level filtering of DIS PDUs is required. The SGI workstations in experiment #4, exercise #2 produced more then 70% of the datagrams generated in this simulation exercise. The Sun machines in experiment #4, exercise #1 generated just under 29% of the DIS PDUs in this simulation exercise and even though PDU production rates were reduced, the slower Sun platforms still exhibited performance related problems.

The scenario selected for experiments #4 and #5 demonstrates again that not only can multicast group addressing be effectively used in a routed communication framework where broadcast addresses (as in classical DIS) would be ineffective, but that an appropriate selection of entities and multicast group address filters within a simulation can yield significantly different results.

In particular, the results from experiment #1 demonstrated how some machines such as Sparcstation 1's could not accurately simulate four DIS entities in a 15-entity exercise[21]. Experiment #2 shows a dramatic improvement in machine performance of the Sun workstations when half of the entities were removed from the simulation by network layer filtering. Performance on the Sun machines was again reduced, as shown in the results of experiments number four and five, when application layer filtering was employed instead of network layer filtering.

Throughout the experiments, network bandwidth utilization never rose above 1.2%, and was as low as .6% in experiment #2. In experiments #3, #4 and #5, network loading was equivalent to or greater then the load seen in experiment #1. No additional traffic was placed on the testbed network because Ethernet ports E6 and E9 were setup to run in an isolated mode only. Special care was taken to ensure that other network traffic (such as NIS, DNS, or Telnet traffic) was isolated from the testbed network.

[21] The experiment was designed to place four entities on each of four simulation hosts. However, due to a typographical error in one script file, the machine labeled SGI-2 only generated 3 entities.

7. RECOMMENDATIONS

The issue of whether a simulation application should perform inbound datagram filtering is currently a topic under much debate in the DIS community. Some parties in the community believe that application level filtering is the only way to manage and control simulation resources. These techniques require that certain control structures and operations be defined as DIS PDUs. Other parties have proposed that lower level network and transport layers are the correct place for certain process to process interaction and control (SIMAN, for example). This experiment was chosen to highlight the performance degradation that occurs when applications are required to manage and filter what has historically been considered network layer information.

The TRIDIS team suggests that the following steps be taken for implementing IPmc within in the DIS standard:

- Continue testing IPmc.
- Determine appropriate IPmc entity-group management.
- Contribute to the Internet Protocols standards development for IPmc.
- Integrate and test bandwidth reservation schemes and other internet services which contribute to network interface quality of service issues.

As shown in experiments #1, #3, #4 and #5 some machines simply could not produce enough CPU cycle time to keep up even though the network bandwidth utilization was at less then 1%. Research should be conducted to determine what the limitations are in a distributed, interactive program and whether simulation management can take control and help load-balance the entire simulation exercise.

Another area that wasn't clearly understood before these experiments was how the deterministic performance of a simulation exercise can be degraded strictly on the basis of widely varying performance levels of the simulation hosts themselves. Our results show that even small simulation exercises (those containing fewer then two dozen entities) can overload simulation hosts which exhibit low performance. If a simulation can become invalid simply because one machine cannot keep up with its data requirements, then complete simulation verification, validation and authentication may not be possible with in the current DIS paradigm.

8. REFERENCE

[Ammar92] Ammar, M. H., Wu, L. R. *"Improving the Performance of Point to Multi-Point ARQ Protocols through Destination Set Splitting"* Proceedings of IEEE INFOCOM `92, Florence, Italy, May 1992, pp 262-271.

[Andrews95] Andrews, M., Davison, E. *"Multicast Address Allocation"* Internet Draft (October 1995)

[Armitage95] Armitage, G. *"Support for Multicast over UNI 3.1 based ATM Networks"* Internet Draft (Sept. 1995)

[Armstrong92]Armstrong, S., Freier, A., Marzullo, K. *"Multicast Transport Protocol"* DARPA RFC1301 (February 1992)

[Boggs, 82] Boggs, David R., *"Internet Broadcasting,"* Ph.D. Thesis, Stanford University, January 1982.

[Bormann] Bormann, C., Ou, J., Gehrcke, H.C., Kerschat, T., Seifert, N. *"MTP-2: Towards Achieving the S.E.R.O. Properties for Multicast Transport"*

[Braden, 89] Braden, R. (ed.), *"Requirements for Internet Hosts -- Communication Layers,"* RFC 1122, USC/Information Sciences Institute, October 1989.

[Braudes92] Braudes, R. *"Requirements for Multicast Protocols"* RFC 1458 (May 1993)

[Braudes93] Braudes, R., Zabele, S. *"Requirements for Multicast Protocols"* DARPA RFC 1458, May 1993

[Calvin95] Calvin, J. O., Cebula, D. P., Chiang, C. J., Rak, S. J., Van Hook, D. J. *"Data Subscription in Support of Multicast Group Allocation"* DIS Position Paper 95-13-093 (Sept. 1995)

[Cheriton88] Cheriton, D. *"VMTP: Versatile Message Transaction Protocol Specification"*, RFC 1045, Stanford University, (February 1988.)

[Cheung94] Cheung,, S. Y., Ammar, M. H. *"Using Destination Set Grouping to Improve the Performance of Window-controlled Multipoint Connections"* GIT, College of Computing, Tech Report GIT-CC-94-32, August 1994.

[Chimiak93] Chimiak, W. *"A Comment on Packet Video Remote Conferencing and the Transport/Network Layers"* RFC1453(April 1993)

[Clark95] Clark, R., Ammar, M. *"Providing Scaleable Web Service Using Multicast Deliver "*, Georgia Tech, College of Computing, Technical Report GIT-CC-95-03, January 1995. (to appear in Proceedings of the IEEE Workshop on Services in Distributed and Networked Environments, Whistler, BC, June 1995.)

[Conta95] Conta, A., Deering, S. *"Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (Ipv6) Specification"* Internet Draft (June 1995)

[Crowcroft88]Crowcroft, J., Paliwoda, K. *"A Multicast Transport Protocol"* ACM SIGCOMM 1988: Commun. Arch. Protocols, 18(4), August 1988, p. 247-256.

[DEC, 80] *"The Ethernet - A Local Area Network,"* Version 1.0, Digital Equipment Corporation, Intel Corporation, Xerox Corporation, September 1980.

[Deering, 89] Deering, S. *"Host Extensions for IP Multicasting"* RFC 1112 (August 1989)

[Deering95] Deering, S., Estrin, D., Farinacci, D., Jacobson, V., Liu, C., Wei, L., Sharma, P., Helmy, A. *"Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification"* Internet Draft (Sept. 1995)

[Deering88] Deering, S., Partridge C, and Wairzman, D. *"Distance Vector Multicast Routing Protocol"* DARPA RFC 1075, November 1988

[Delgrossi, 95] Delgrossi, L., and Berger, L., Editors, *"Internet Stream Protocol Version 2 (ST2),"* RFC 1819.

[Dempsey94] Dempsey, B. J., Lucas, M. and Weaver, A C, *"High-Quality Video Distribution using XTP Reliable Multicast"* IWACA 94, Heidelberg, Germany, September 1994

[Dempsey89] Dempsey, B. J., Strayer, T and Weaver, A C, *"Issues in Providing a Reliable Multicast Facility"* University of Virginia Department of Computer Science Technical Report CS-89-15, November 1989

[Dempsey90] Dempsey, B. J., Weaver, A C, *"Issues in Designing Transport Layer Multicast Facilities"* University of Virginia Department of Computer Science Technical Report CS-90-18, July 1990

[Denning68] Denning, P. J. *"Resource Allocation in Multiprocess Computer Systems"* PhD. dissertation, Dept. of Elec. Engr., MIT, Cambridge, Mass. [and available as MIT Project MAC technical report TR-50].

[Donnelley95]Donnelley, J. E. WWW Media Distribution via Hopwise Reliable Multicast, Paper to be presented at the WWW 95 Conference. A European copy is held at Stuttgart University in Germany.

[Donnelley] Donnelley, J. E. WWW Page Distribution Using Multicast, Paper circulated to the mbone@isi.edu mailing list. The up to date copy is held at LLNL.

[Floyd95] Floyd, S., Jacobson, V., Liu, C., McCanne, S., and Zhang, L., *"A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing"* To appear in ACM SIGCOMM 95.

[Forgie, 79] Forgie, J., *"ST - A Proposed Internet Stream Protocol,"* IEN 119, M. I. T. Lincoln Laboratory, 7 September 1979.

[Handley] Handley, M., Wakeman, I, *"CCCP: Conference Control Channel Protocol. A scaleable base for building conference control applications V1.4"* University College London

[Herbst95] Herbst, J. C., Kirkland, J. H., Berg S. *"An Architecture to Support Highly Interactive, Large N, Distributed Simulations"* DIS Position Paper 95-13-079 (Sept. 1995)

[Hofmann95] Hofmann, M., Braun, T., Carle, G., *"Multicast communication in large scale networks"* to appear in Proceedings of the IEEE Workshop on the Architecture and Implementation of High Performance Communication Subsystems (HPCS'95), Mystic, Connecticut, August 1995.

[Hornig, 84] Hornig, Charles, *"A Standard for the Transmission of IP Datagrams over Ethernet[2] Networks,"* (April 1984), RFC 894, electronically published by the Network Working Group and the Internet Architecture Board.

[IEEE 1278.1]IEEE Standard 1278.1 1995. *"Standard for Distributed Interactive Simulation--Application Protocols."*, Institute of Electrical and Electronics Engineers, Inc., 345 East 47th St., New York, NY 10017.

[IEEE 1278.2]IEEE Standard 1278.2, 1995. *"Standard for Distributed Interactive Simulation--Communication Services and Profiles,"* Institute of Electrical and Electronics Engineers, Inc. 345 East 47th St., New York, NY 10017.

[Jones91] Jones, G. W. M., Sørensen, S-A, and Wilbur, S R, *"Protocol design for large group multicasting: the message distribution protocol"* Computer

Communications, 14(5), June 1991, p. 287-297

[Kelly95] Kelly, B. L., Aggarwal, S. *"Hierarchical Structuring for Distributed Interactive Simulation"* DIS Position Paper 95-13-016 (Sept. 1995)

[Laubach95] Laubach, M. *"IP over ATM Working Group's Recommendations for the ATM Forum's Multiprotocol BOF Version 1"* RFC1754 (January 1995)

[Lear94] Lear, E., Fair, E., Crocker, D., Kessler, T. *"Network 10 Considered Harmful (Some Practices Shouldn't be Codified)"* RFC1627 (July 1994)

[Macedonia94a]Macedonia, M. R., Brutzman, D. P. *"MBone Provides Audio and Video Across the Internet"* IEEE Computer Magazine (April 1994)

[Macedonia94b]Macedonia, M. R., Zyda, M. J., Pratt, D. R., Barham, P. T. *"Exploiting Reality with Multicast Groups: A Network Architecture for Large Scale Virtual Environments"* DIS Position Paper 94-163 (Sept. 1994)

[Malkin94] Malkin, G. *"RIP Version 2 Protocol Analysis"* RFC1387 (November 1994)

[McLaughlin89]McLaughlin III, L. *"A Standard for the Transmission of IP Datagrams over NetBIOS Networks"* RFC1088 (February 1989)

[Mills88] Mills, D., *"Network Time Protocol (Version 1), Specification and Implementation"*, STD 12, RFC 1119, University of Delaware, July 1988.

[Mogel, 84] Mogul, J., *"Broadcasting Internet datagrams in the presence of subnets,"* FC 922 (October 1984)), electronically published by the Network Working Group and the Internet Architecture Board.

[Mogel, 85] Mogul, J. and Postel, J., *"Internet Standard Subnetting Procedure,"* FC 950 (August 1985)), electronically published by the Network Working Group and the Internet Architecture Board.

[Moy, 94a] Moy, J *"The OSPF Specification"*, RFC 1583, Proteon, March 1994.

[Moy, 94b] Moy, J. *"MOSPF: Analysis and Experience"* RFC 1585 (March 1994)

[Pullen95a] Pullen, J. M., Laviano, V. P. *"A Selectively Reliable Transport Protocol for Distributed Interactive Simulation"* DIS Position Paper 95-13-102 (Sept. 1995)

[Plummer, 82]Plummer, David C., *"An Ethernet Address Resolution Protocol"* RFC 826

(November 1982), electronically published by the Network Working Group and the Internet Architecture Board.

[Pope, 89] Pope, A., *"The SIMNET Network and Protocols,"* BBN Report No. 7102, BBN Systems and Technologies, July 1989.

[Postel, 94] Postel, J., Ed., *"Internet Official Protocol Standards"* RFC 1720 (November, 1994), electronically published by the Network Working Group and the Internet Architecture Board.

[Pullen95b] Pullen, J. M., White, E. L. *"Analysis of Dual-Mode Multicast for Large Scale DIS Exercises"* DIS Position Paper 95-13-096 (Sept. 1995)

[Pusateri93] Pusateri, T. *"IP Multicast over Token-Ring Local Area Networks"* RFC1469 (June 1993)

[Schug95] Schug, K. H. *"DIS NG - A Flexible Protocol For All Simulation Applications"* DIS Position Paper 95-13-061 (Sept. 1995)

[Simpson95] Simpson, W. *"ICMP Domain Name Messages"* RFC1788 (April 1995)

[Siyan92] Siyan, K. *"An IP Address Extension Proposal"* RFC1365 (September 1992)

[Smith95a] Smith, J. E. *"Effectiveness of Various New Bandwidth Reduction Techniques in ModSAF"* DIS Position Paper 95-13-092 (Sept. 1995)

[Smith95b] Smith, J. E., Russo, K. L., Schuette, L. C. *"Prototype Multicast IP Implementation In MODSAF"* DIS Position Paper 12-95-032 (March 1995)

[Symington95]Symington, S. F., Armitage, G. *"IP multicast over ATM in support of DIS"* DIS Position Paper 95-13-048 (Sept. 1995)

[Symington94]Symington, S., Wood, D., Pullen, M. *"Modeling and Simulation Requirements for Ipng"* RFC1667 (August 1994)

[Topolcic, 90] Topolcic, C. Editor, *"Experimental Internet Stream Protocol, Version 2 (ST-II)"*, RFC 1190, CIP Working Group, October 1990.

[Van Hook95]Van Hook, D. J., Brittain, E. A. *"An Implementation of a Data Consistency Mechanism"* DIS Position Paper 95-13-097 (Sept. 1995)

[Van Hook94]Van Hook, D. J., Calvin, J. O. *"An Approach to DIS Scaleability"* DIS

Position Paper 94-141 (Sept. 1994)

[Waitzman, 88] Waitzman, D., Partridge, C., and Deering, S. *"Distance Vector Multicast Routing Protocol"*, RFC-1075, BBN STC, Stanford University, (November 1988.)

[Wobus, 89] Wobus, J., *"Introduction to Internet Networking,"* Computing & Network Services, Syracuse University., February 1989, Document Number:

0000181