A SCALABLE ARCHITECTURE FOR DISTRIBUTED INTERACTIVE SYSTEMS

William J. Rowan, Naval Air Warfare Center TSD, Orlando, Florida
Sam T. Giambarberee, Naval Air Warfare Center TSD, Orlando, Florida

BACKGROUND

The ability to conduct interactive simulations over a standardized networking architecture was introduced in the mid-1980's with the advent of SIMNET. The success of SIMNET, designed to accommodate relatively slow-moving vehicles, spawned the effort to develop the Distributed Interactive Simulation (DIS) standard to handle a wide spectrum of simulated entities including fighter aircraft and missiles. Distributed Interactive Simulation offers the means of conducting simulated military operations involving units in different parts of the world without these units having to leave their home bases. The value of this capability is now widely recognized among military leaders.

The Army has led the way with the training center at Ft. Knox and state-of-the art systems such as Close Combat Tactical Trainer (CCTT). The Air Force's Advanced Distributed Simulation was employed in a major demonstration of live, virtual and constructive simulations in December of 1995 called "Warfighter 95". In Warfighter 95, actual aircraft flying over Nevada engaged piloted simulators and constructive aircraft in Florida. Warrior Flag 97 will take advantage of lessons learned from Warfighter 95. The Navy's Battle Force Tactical Trainer is currently deployed on 20 Aegis class ships and is used to stimulate the various ship consoles such as radar, sonar, fire control, etc. Future plans call for large-scale deployment.

All services are participating in the development of JSIMS (Joint Simulation System) which in the future could be used to realistically rehearse operations such as Desert Shield and Desert Storm.

The Advanced Research Project Agency has projected that future distributed exercises will involve up to 100,000 entities. Projections made in 1995 were for 50,000 entities at 30 sites by 1997 and 100,000 entities at 50 sites by the year 2000 [1]. The networking and computational problems of such exercises remain to be solved. Today's network interface units can handle a maximum of about 1000 - 2000 entities per second.

In concurrence with the increasingly demanding requirements, advances in the areas of personal computers and networking technologies have presented an opportunity for a scalable approach to performing all the tasks associated with local and remote entities and to distributing these entities to the other sites involved in the exercise.

OBJECTIVE

NAWCTSD was funded by the Office of Naval Research to develop, demonstrate, and evaluate an architectural model to enable DIS sites to meet all projected needs for network bandwidth and computational power. The model had to accommodate a spectrum of users ranging from those involved only infrequently in distributed exercises or only in small exercises, to sites participating in frequent and/or large scale exercises. Obviously, scalability needed to be achieved in terms of the resources which need to be applied as a function of the number of entities of interest. The goal was to design the architecture to take advantage of: 1) the ever-improving price/performance ratios of commercial Personal Computers and 2) potential public networking services from the telephone companies such as ATM (Asynchronous Transfer Mode) services with pricing similar to today's Switched Data Service or Frame Relay.

APPROACH

The first step in the process was to establish the network bandwidth and computational capability required for DIS sites to participate in large scale, distributed exercises. Computations by Thomas Gehl [2] determined that 100,000 entities would equate to a packet throughput of 65,600 packets per second and a backbone bandwidth requirement of 134 Mbps using 256-byte packets. This conclusion is based on the use of the currently-employed broadcast paradigm. Assuming a simulator might interact with 10% of the entities participating in an exercise and considering the performance of

current Network Interface Units (NIU), an NIU to support these large exercises would probably require the power of 5 to 10 modern microprocessors.

It was recognized that requirements would vary from exercise to exercise and from site to site. Therefore, an architecture had to be designed which would allow sites to add resources only as needed for a particular exercise; ideally, these same resources could be used for other purposes when not participating in distributed exercises. The architecture, once designed, had to be evaluated for actual scalability and cost effectiveness.

TEST BED

The primary objective of the design was scalability for both the computational element and the networking technology. A "building block" approach was used to be able to apply computational power to the network interface unit depending upon the number of relevant entities. The architecture is depicted in Figure 1 which shows multiple simulator hosts, each of which would be connected via the ATM switch to the PCs comprising its own NIU. In the test bed at NAWCTSD, one simulator host is used with a 4-PC NIU. The PCs use single 90 MHz Pentiums and are interconnected via a 155 megabit per second OC-3 ATM network. Windows NT 3.51 is the PC operating system.

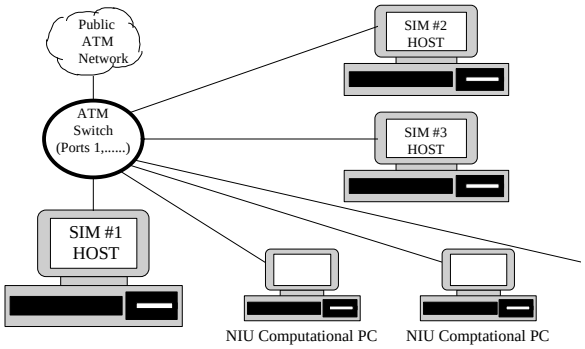

FIGURE 1
Scalable Architecture

PCs were selected for the platform because of their price/performance ratio, their fast-paced speed improvements, and their general availability at military sites. To achieve true scalability, these PCs needed to be interconnected via a scalable networking technology; ATM, with the potential to scale to OC-12 and possibly OC-48 at 2.4 gbps fulfills this requirement. The ATM network conformed to LANE (LAN Emulation) Version 1.0. As suitable ATM services become available, this LAN will integrate seamlessly with the Wide Area Network.

Each PC assumes NIU tasks such as dead reckoning and collision detection for locally generated and remote entities based on its own loading and the availability of the other PCs. The algorithm employed for associating entities with PCs must be distributed in nature to avoid potential choke points such as would be the case if a single PC were used to perform the assignments. Further, there should not be a requirement for apriori knowledge of the number of PCs involved.

The method employed requires registration of all PCs with the simulator host prior to the start of an exercise. As each PC registers, it is assigned a number ranging from 1 to the number of PCs employed (Figure 2). For example, in the left half of Figure 2, the topmost processor is the fourth to send the proc_reg to the simulator host; it is then designated as "#4" via the proc_assign message. This number is used in the distributed entity acceptance process.

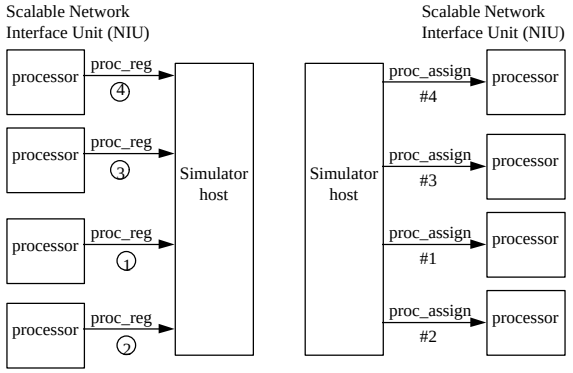

FIGURE 2
NIU Processor Registration

Each PC maintains a list of entities it is currently handling (OwnPC) and a single list of entities assigned to all the other PCs (OtherPC). When a new entity arrives, each PC finds that it is not in the OwnPC list. If it is not in the OtherPC list either, and the PC is not fully loaded, it checks to see if any lower numbered (from registration process) PC is available. If no lower numbered PC is available, it accepts the entity and

announces acceptance to the other PCs. See Figure 3.

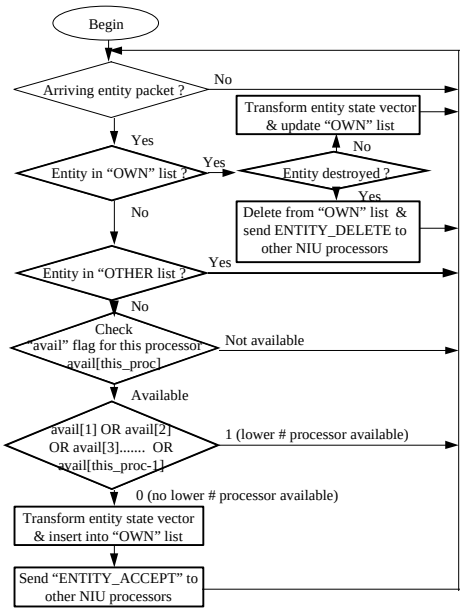

FIGURE 3
Distributed entity acceptance process

Each frame, all the NIU PCs send a PC_AVAIL message to all other NIU PCs, based upon the amount of spare time remaining in the frame and the amount of entity buffer space available. Spare time is defined as the amount of time remaining between the completion of the NIU computational chores for a frame and the beginning of the next frame.

This approach offers some degree of fault tolerance which is important in large long-running exercises. A PC failure will result in the loss of only a percentage of the entities - in effect, a soft failure. Recovery, in fact, could easily be incorporated by interpreting an absence of PC_AVAIL messages from any PC as a failure of that PC and removing all entities handled by it from the OtherPC list. At that point, entity-state PDUs from the removed entities would be processed as "new" entities and would be accepted by the appropriate PC.

MEASUREMENTS

A PC was configured to mimic a simulator host by sending sync frames and "local" Entity State (ES) PDUs to the NIU PCs, and receiving dead-reckoned updates from the NIU. This PC was also configured as a net loader to inject "remote" ES PDUs into the ATM LAN, in effect simulating a connection through the switch to the ATM Wide Area Network. The number of these artificially generated entities was increased incrementally to determine the point at which the PCs began to report "not_available". The intent was to determine the number of entities which each PC could handle and the relationship between number of PCs and the total number of entities. Simultaneously, ATM network traffic was monitored using a RADCOM RC-200-C ATM analyzer.

A two-pronged approach was used to determine the limits of the 90 MHz Pentium systems. The first goal was to find the computational limits of the system independent of the I/O load. The second goal was to find the I/O limitation of the system before packet loss was incurred due to either NIU processing or to hardware limitations.

To determine the computational limit of the NIU PCs, the system was loaded with three ownships, and allowed to accept new entities as fast as they arrived. The system performed all required NIU functions (such as dead-reckoning, collision detect) on the incoming entities while the time for handling these entities was measured. When this time reached 30ms (90% of a 33ms frame) the entity count was recorded and the system was considered computationally loaded.

To determine the I/O limit of the NIU PCs, packet sequence numbers were employed. This allowed us to determine when the NIU would begin dropping packets while performing all required NIU functions. We were careful to rule out processing-induced limitations for I/O bandwidth. For example, if the incoming packets hashed poorly and resulted in a large overflow list, significant processing time would be spent searching the entity lists to determine if an entity had already been received by OwnPC or OtherPCs.

At this writing, the problems associated with frame-rate updates of the simulator host have not been fully investigated. In order to minimize transmissions from the NIU to the host, multiple entity state updates would be sent in a single packet with the updates being limited to single byte deltas of the state vector components.

RESULTS

Initial tests were performed to validate the distributed entity acceptance algorithm. Once the algorithm was validated, tests were run to determine separately the aggregated computational and networking capacities. For testing purposes, the assumption was made that

remotely generated entities would greatly outnumber locally generated entities; for testing, then, only remote entities were considered.

Our first attempt to handle the functions of the distributed NIU was to use multiple threads under NT. The "MAIN" thread handled PC initialization including Simhost registration and PC assignment, START_FRAME messages from the Simhost, and interPC message traffic such as PC_AVAIL, ENTITY_ACCEPT, and ENTITY_DELETE. The "T1" thread interfaced to the Wide Area Network and processed remote PDUs deciding whether or not to accept arriving entities. Vector transformations from geocentric to UTM were performed on the accepted entities. The "T2" thread performed dead reckoning, collision detect, and other NIU functions on the remote entities as well as sending remote entity updates to the Simhost. The "T3" thread processed local PDUs from the Simhost, but was not implemented for this test.

With this seemingly logical selection of threads, blocking sockets were appropriate for both the MAIN and T1 threads. After processing a packet, each of the threads would wait at the recvfrom() call until another packet arrived. Our primary concern was to give highest priority to the T1 thread processing the remote PDUs since we didn't want to lose any of the incoming packets. Setting the thread priority using SetThreadPriority (THREAD_PRIORITY_HIGHEST) did not change the fact that NT assigned a minimum amount of time to handle other computable threads when they obtained the processor [5]. These time slices ranged from 10ms to 20ms in a rather unpredictable manner. It was during the non-T1 time slices that we observed several packets being dropped. The results suggested that more packets were arriving during these time slices than the buffers could accommodate. At an incoming rate of 1000 packets/second, at least 10 packets were dropped for each time slice devoted to non-T1 threads. As expected, at a rate of 2000 packets/second, at least 20 packets were dropped per time-slice. This was unacceptable and forced us to reconsider our thread scheme. NT threads were not suitable for this type of real-time process.

In the end it was necessary to utilize a single process design that did nonblocking T1 reads by periodically polling the receive sockets during idle frame time. It was also necessary to perform occasional I/O polling during compute-intensive dead-reckoning/collision detect stages. While threads might be useful on a multi-processor PC, we were unable to use them effectively in a real-time environment on a single processor system.

The computational capacity was measured by simulating the arrival of large numbers of remote entities at a fixed rate and observing the point at which the aggregated computational power was exhausted. A fixed rate of 1500 packets/second assured that Entity State PDUs would not be lost due to NIC buffer overflow. Sequence numbers were used to verify that packets were not being dropped. Each PC performed all NIU functions (vector transformations, dead reckoning, collision detect, etc) and was allowed to accept entities until the computational time reached 30 ms. At that time, "not_available" was reported to the other PCs. The algorithm employed resulted in the acceptance of all new entities by PC#1 until its threshold was exceeded, followed by similar action by PC#2, etc. These tests showed that each of the 90 MHz Pentium PCs could handle approximately 3200+ entities for a simulator running at 30 iterations per second. As PCs were added to this testbed, the number of entities increased linearly to a total of 13,000 entities on 4 PCs (see figure 4). The linearity was not unexpected since increasing numbers of PCs do not appreciably add to the overhead. At some point, however, as more and more PCs are added, the OtherPC list will get larger and each successive PC could possibly have to traverse many more entities if the incoming entities didn't hash well. This would make the process more compute intensive for additional PCs and possibly affect the linearity. The limited number of PCs available to the test bed did not permit further exploration of this issue.

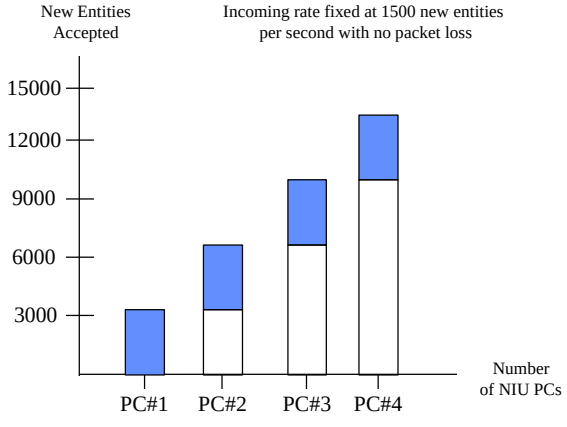

FIGURE 4
Entity acceptance rate

In a highly dynamic simulation of 100,000 entities, the processing of interPC messages such as ENTITY_ACCEPT, ENTITY_DELETE, and PC_AVAIL, would probably not affect the results as that information is sent out at frame rate buffered in an array. These arrays are capable of handling 5000 new entities each second. What would be affected however, is the ability to handle the massive incoming packet rate that 100,000 entities would generate. This clearly becomes an I/O issue at this point.

The next testing objective was to look at the I/O issue and determine the throughput limitations of the network interface controller (NIC). The NIC's used were Interphase 5515 ATM boards using the PCI bus. In contrast to this architecture's ability to add computational power by attaching another PC, the network interfacing capacity is fixed by the network speed and the speed of the Network Interface Card, with scalability resulting from that of the network technology employed. This is true, of course, because of the broadcast paradigm resulting in all of the NIU PCs receiving all of the Entity State PDUs from the wide area network. For the purposes of this testing, NIC buffer size was virtually irrelevant since the objective was to measure maximum sustained rate; buffering is ineffective for high-rate uniform traffic. ES_PDUs were applied to the network with the NIU PCs continuously reading from the receive socket.

Initial results of the number of ES_PDUs which could be processed before packet loss occurred was disappointing, the number being less than 200 per second. Investigation revealed that the use of NT threads was the cause of this limitation. After redesigning the process as mentioned above each NIU PC was able to handle 1500 ES_PDUs per second without packet loss. With the receiving PC programmed to do nothing more than receive PDUs and note losses, the limit was raised to 2000 PDUs per second. Bandwidth utilization at this rate was less than 4 megabits per second per fiber, so the ATM network itself was not contributing to this limit. Considering the capabilities of the Pentium processor, it appears that the NIC and/or PCI bus combination was the limiting factor.

ADVANTAGES

One of the major advantages of this architecture is the ability to "erect" an NIU from inexpensive PCs to address the metrics of a particular exercise. The alternative is to design a Network Interface Unit to handle the largest possible exercise. Many military sites would not be able to afford such an NIU. The use of PCs and Windows NT takes advantage of the market-forced improvements in speed and performance and, from a Navy point-of-view, is consistent with the requirements of Information Technology 21 [3] policy. Another advantage is the use of ATM which offers a natural connection to ATM wide area network services, and imbues scalability into the local area network interconnecting the PCs. OC-12 network interface cards are available now with speeds of 622 Mbps and OC-48 NICs, when they become available, will provide over 2gbps.

DISADVANTAGES

The use of Windows NT as used for this project precluded the precise timing implementation demanded by real-time simulation. Frame timing was implemented using NT's "TimeGetTime" which returns the time since NT was started. Resolution was set at 1 millisecond via the "TimeBeginPeriod". These functions were used in a polling fashion by the PC acting as the simulator host to provide frame syncs to the NIU PCs; TimeGetTime was polled after each PDU was sent to determine if the frame time had elapsed; if so, frame_sync was sent to the NIU PCs. To use Windows NT effectively in this application, a better frame synchronization method would have to be implemented.

SCALABILITY ACHIEVED

As can be seen from Figure 4, the capability of the NIU scaled rather linearly as PCs were added to the simulation. A total of 4 PCs handled 13000+ entities in a linear fashion. To draw a valid conclusion regarding scalability to 100,000 entities, further tests need to be run using additional PCs. It is expected, of course, that PC's operating at 200+ MHz will reduce the computational time required to handle 100,000 entities. Likewise, the use of faster ATM boards could overcome the I/O limitations mentioned previously.

COMMERCIAL ATM SERVICES

Network scalability at the LAN level is achieved via the spectrum of commercially available switches and NICs. For wide area networking, affordable usage-priced services

have just begun to arrive with offerings at a variety of speeds, generally in increments of DS1 (1.544 Mbps). Although Permanent Virtual Circuit (PVC) services have been available since about 1993, their use incurred substantial, fixed monthly charges. Now, the three major InterExchange Carriers (AT&T, MCI, and Sprint) will offer SVC services with a fixed monthly port charge plus a charge per unit data [4]. Port charges are a function of the line speed. As examples of pricing, AT&T's T1 port charge is $2200/month with a usage charge for a Variable Bit Rate (VBR) service with a 1 Mbps Committed Information Rate (CIR) of $0.69 per minute. Sprint's "2 x T1" port charge is $2800 with usage charges of $1.20/million cells for Constant Bit Rate (CBR) service and $0.70/million cells for Variable Bit Rate service regardless of line speed.

Asymmetric Digital Subscriber Line (ADSL) technology is now providing downlink speeds upward of 5 Mbps from the Internet Service Providers (ISP's) into homes in some U.S. areas over existing copper wires. In Korea, Very High Speed DSL (VDSL) is delivering 27 Mbps to the home. Aggregately, the various DSL technologies are referred to as "xDSL". Using xDSL modems, it would be possible for the Local Exchange Carriers (LEC's) to provide high speed ATM links to sites without having to install new cabling between the Central Office's (CO's) and the sites. This should lead to reduced charges for accessing Inter Exchange Carriers' (IXC's) ATM services. Within limited areas, it would be possible for the LEC's to interconnect sites directly using ATM over xDSL. xDSL could also lead to the advent of Wide Area Service Providers which, like the ISP's, could provide access to wide area services from the CO's.

CONCLUSIONS

Scalability was achieved in a linear fashion from a computational viewpoint using up to 4 PCs. Based upon Gehl's estimate for an exercise involving 100,000 entities [2] and a supposition that a simulator would be "interested" in 10% of these entities, this architecture would need to scale to a 10000 entity capability involving 6500 PDUs per second. Computationally, the test bed met this objective with 4 PCs. It would probably be possible to add several more PCs before this linearity would begin to fall off. In practical terms, then, this architecture could accommodate the computational load imposed by the entities of interest even in the largest of exercises.

The I/O limitations of the architecture are set by the selection of a particular ATM speed, in this case 155 Mbps OC-3. Since 1500 ES-PDUs per second consumed less than 3 percent of the bandwidth, it would be tempting to conclude that a more efficient NIC/ PCI bus/ processor combination should handle upwards of 40,000 packets per second. With the small pdu size, however, this number would undoubtedly be greatly reduced by a packet per second limitation.

In fact, Figure 5 shows that for DIS packet sizes of approximately 200 bytes, only 3.5 Mbps of the available bandwidth is utilized [6]. This is consistent with the results obtained during laboratory experimentation.

The Interphase 5515 board and driver used were essentially "first-generation", and later releases will certainly have improved performance. As performance improves, the architecture as implemented in the test bed should be able to handle the largest of exercises, even in consideration of the additional loading imposed by simulator host frame-rate updates.

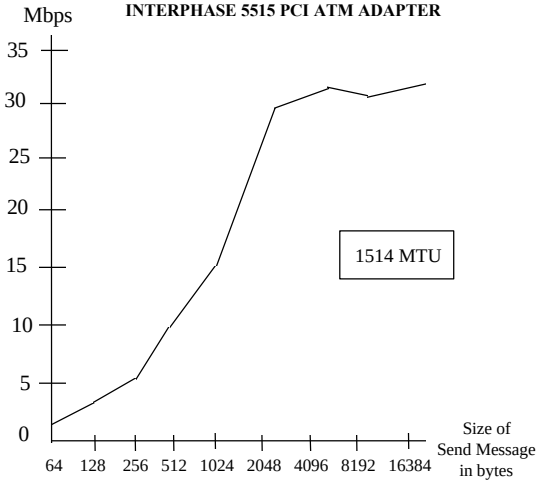

FIGURE 5

ATM Adapter Performance[6]

The Interphase adapter shows consistent growth with increased message sizes. The graphs look even throughout the MTU range of 1514-9218 bytes

Since I/O is the limiting factor for running large scale exercises, techniques such as aggregating PDUs and altering the ES_PDU retransmission interval, would reduce I/O requirements. I/O scalability may also be addressed by upgrading the OC-3 boards on the PC to OC-12 or OC-48 ATM boards.

Windows NT 3.51 also had its limitations with not providing a technique for handling interrupt driven I/O events. This affected synchronization among NIU PCs since each PC received the START_FRAME at different times. For true real-time simulation, additional deterministic features would have to be added to the Windows NT operating system.

RECOMMENDATIONS

The use of the "ganged PC" approach to NIU design is both efficient and cost effective from a computational viewpoint. More research needs to be done to improve the I/O capability of the NIU, but it is felt that this limitation will be eliminated with normal technological improvements in ATM Network Interface Cards. With the enforcement of such policies as the Navy's IT-21, most sites will have access to personal computers running Windows NT. To be able to employ this approach, military sites may only need to implement an ATM local area network. Prices of switches and ATM NIC's have been coming down in recent months. It is recommended that this architecture be considered for any new NIU designs.

Commercial ATM services now seem like a cost-efficient alternative to leased lines in some applications. Comparing a fixed price leased T1 line from Orlando to San Diego ($8000/month) to a "typical" ATM VBR SVC over a T1 circuit with a port charge of $2200/month and usage charge of $0.64/minute, the ATM service could be used for about 90 hours a month before its price would exceed that of a leased line. For infrequent users, ATM would obviously be more economical. On the other hand, for the more complex scenarios such as those envisioned by ARPA, multiple-site connectivity is required. Ideally, any number of Virtual Circuits (VC's) could be established over a single port at appropriate rates. An ABR service would probably be ideal for this purpose to minimize the total port bandwidth required to satisfy all the individual VC bandwidth requirements. A simulator hosting relatively slow moving vehicles might burst outgoing network traffic every 5 seconds and be relatively quiet at other times. In a network of such simulators, it would be impracticable to lease VBR ports at the burst speed to establish a mesh of VC's; what is needed is a port which would accept multiple ABR VC's with a Peak Cell Rate (PCR) equal to the burst speed. The speed of the port would probably not have to be any greater than the burst speed. ABR was conceived to accommodate just such bursty traffic, but since it is very complex with not all of the details fully specified. it might be some time before the service is offered commercially. For the immediate future, leased networks will be more cost-effective for multiple-site distributed interactive simulations.

REFERENCES

[1] Dr. Stu Milner, ARPA, in address to "Distributed Interactive Simulation IP/ATM Multicast Symposium" at Naval Research Laboratory, May 25, 1995

[2] T.L. Gehl, "Dynamic Multicast on Asynchronous Transfer Mode for Distributed Interactive Simulation", Proceedings from the 16th Interservice/Industry Training Systems and Education Conference. November, 1994

[3] Admiral Archie Clemins IT-21 address at AFCEA/Naval Institute EXPO Conference January 24, 1997

[4] Data Communications, April 1997, Robin Gareiss

[5] Mastering Windows NT Programming, 1993, Brian Myers, Eric Hamer

[6] Performance Benchmarking for Interphase PCI ATM Adapters. 1996 Interphase Corp. Mike Eckley

****