Performance Evaluation of the XMSF Overlay Multicast Prototype

Dennis M. Moen
J. Mark Pullen
George Mason University, C3I Center, MSN 4B5
4400 University Drive
Fairfax, VA 22030
703-993-1705, 703-993-1538
dmoen@gmu.edu, mpullen@gmu.edu

Keywords:
Overlay Multicast, Distributed Simulation

Abstract: *Growing demand for use of Internet/Web-based services in large scale real-time distributed virtual simulation (RT-DVS) and other real-time applications is fueling extensive interest in overlay multicast protocols. These applications require many-to-many multicast services that are not available as an open Internet service today and are not likely to be offered as an Internet-wide service in the near future. This paper describes an early implementation of the Extensible Modeling and Simulation Framework Overlay Multicast (XOM). XOM is an overlay multicast protocol designed to support many-to-many multicast for RT-DVS applications, using Internet Protocol (IP) multicast above the overlay and IP unicast under the overlay. We summarize the architecture and key design considerations of XOM and provide preliminary analytical, simulation, and laboratory measurement results for our prototype. We conclude with an overview of the range of applications for which the current prototype can be expected to yield satisfactory results.*

1. Introduction

The ability to perform many-to-many multicast over an open network is very important to the real-time distributed virtual simulation (RT-DVS) community and key to implementing the Extensible Modeling & Simulation Framework (XMSF). That framework, recently recognized by the Simulation Interoperability Standards Organization (SISO) [1], has the objective of using XML and commercial Web based technologies for expanding the user base of RT-DVS. Implementing end system or overlay multicast for real-time distributed simulations allows the continued use of open protocols as implemented across the Internet. As a result, RT-DVS is no longer dependent on consistency of network policy implementation across the Internet, Global Information Grid, etc. and supports the RT-DVS community's effort to move to Web based technologies.

The XOM project's objectives are first to develop a workable architecture for an overlay multicast protocol and then to validate its feasibility in a working prototype that is available to interested users for further experimentation [2]. Section 2 of this paper provides background information describing the motivation for XOM and the history of development of the concept. Section 3 provides a description of the proposed XOM architecture and the current prototype, with early laboratory testing results. Section 4 describes a five-node wide area demonstration of XOM across the Internet and includes results of network message traffic characterization using the OPNET ACE [3] modeling tool.

2. Background

Interest in application of Web services to support distributed simulation prompted examination of the adequacy of existing network protocols to support the demanding data distribution needs of the Web services approach. In particular, providing reliable multicast services to RT-DVS is an important requirement to enable use of Web based services for RT-DVS across open networks such as the Internet. The initial requirements document for the Web services approach to distributed simulation was the result of the first XMSF working group [4]. The assembled group of experts specified network requirements that can best be provided using multicast networking. The group also recognized that many open issues with Internet Protocol (IP) multicast over the Internet were likely to continue to be insurmountable and would prevent deployment in a manner that will meet the Quality of Service (QoS) and the many-to-many multicast needs of RT-DVS.

2.1 Motivation for Overlay Multicast

Distributed virtual simulations operating across a network in human time generate large amounts of message traffic among the computers hosting the simulation applications. Efficient distribution of this traffic requires many-to-many communications in a dynamic group environment because unicast transmission among N computers in the group requires O(N^2) total message transmissions, where multicast requires only O(N) [5]. In addition, this simulation environment may not necessarily be homogenous, e.g. each simulator is likely to be different although they dynamically share common simulation objects over time. The result is that simulation objects may have membership in multiple groups with each group's membership changing at different rates.

Distributed virtual simulations also require specific delay bounds to support the delivery of real-time, interactive visual and audio information at human response times. This environment can be described as a multiparty collaborative environment running multimedia applications. The underlying networking environment needs to support a large number of participants dynamically joining and leaving the applications across the myriad of public and private networks that make up the Internet. Because each of these networks is independently managed, the RT-DVS applications cannot rely on the Internet to deliver the necessary QoS even where QoS mechanisms are deployed. As a result, networking real-time simulators together has seen deployment only in specialized local area networks or on private networks dedicated to the simulation environment.

A number of multicast protocols have been developed over several years to support group communications. Historically, these protocols have focused on supporting applications, such as streaming audio and video, which normally have one-to-many type data distributions as indicated in the top of Figure 1. These protocols tend to use core or root based approaches for data distribution, where the data source must first send the data to the root of the tree from which distribution occurs. They also typically are offered as a service only in a private network or single management domain. Hence, these early network layer protocols have exhibited limitations in support of more demanding types of applications [6] and have proved too complex in operation for acceptance by most network operators. For RT-DVS using Web services, available network services are inadequate to support the many-to-many multicast requirement as well as their need for improved QoS.

The constraints of the underlying network services of the open Internet represent inhibitors to deployment of Web Services based distributed simulation. As an example, the throughput required for communicating object status updates potentially is extremely large, consisting of thousands of state updates per second from simulation objects. These object updates typically have message sizes of at least 120 bytes without tag or other protocol overheads. Web-based RT-DVS may be run on heterogeneous sets of workstations with differing processing and display capabilities, traveling over a heterogeneous network with capacities varying by many orders of magnitude between the initial down link and the slowest end user.

We therefore have proposed an overlay multicast protocol designed to support efficient, reliable multicast transmissions over existing network protocols such as UDP/IP. The bottom portion of Figure 1 depicts the flexibility of overlay multicast. Each source is able to build independent distribution trees for data. This allows for cross-network operation: distribution across multiple network administrative domains found in the Internet.

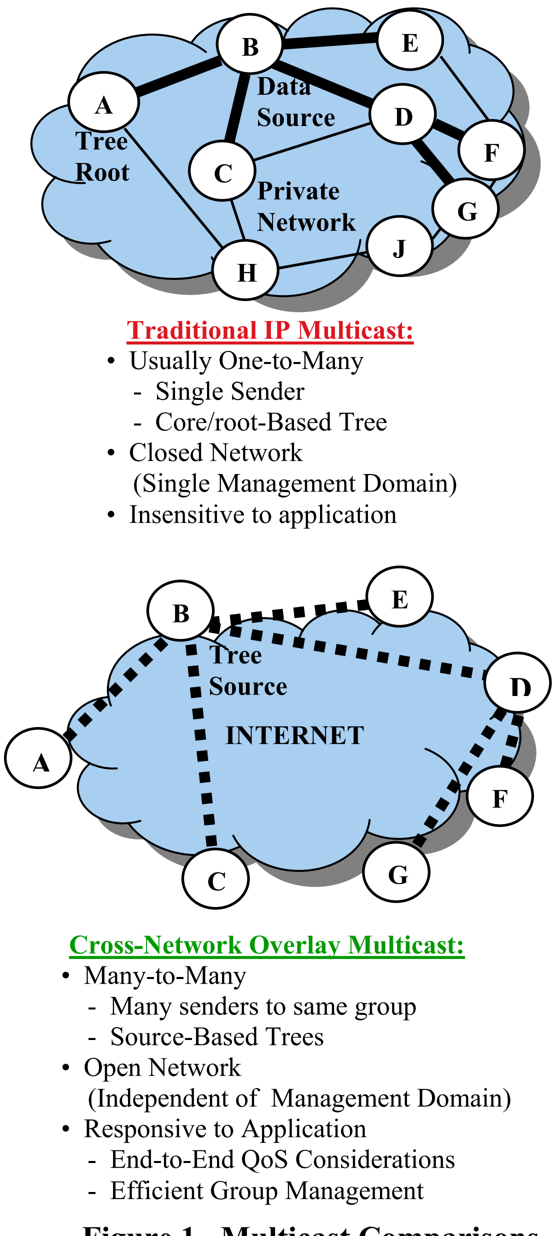

B
E
A
Data Source
D
Tree Root
F
C
Private Network
G
J
H
Traditional IP Multicast:
• Usually One-to-Many
- Single Sender
- Core/root-Based Tree
• Closed Network (Single Management Domain)
• Insensitive to application
B
E
Tree Source
D
INTERNET
A
F
C
G
Cross-Network Overlay Multicast:
• Many-to-Many
- Many senders to same group
- Source-Based Trees
• Open Network (Independent of Management Domain)
• Responsive to Application
- End-to-End QoS Considerations
- Efficient Group Management

Figure 1. Multicast Comparisons

2.2 History of Development

Our laboratory has studied multicasting technologies for several years [5, 11]. Leading-edge work in primarily one-to-many overlay multicast [12, 13] convinced us that this approach offers considerable promise to provide the many-to-many multicasting across multiple domains and heterogeneous networks that will be needed for Internet-wide Web-based RT-DVS. The fact that overlay multicast involves middleware that can potentially provide for QoS management also encouraged us. We therefore undertook a student project in partnership the Naval Postgraduate School (NPS) Modeling, Virtual Environments and Simulation (MOVES) Institute, to create a limited prototype to enable experimentation with the concepts involved. Results of this effort persuaded the Defense Modeling and Simulation Office (DMSO) to support a study to create a workable architecture for many-to-many cross-network overlay multicast, including an open-source prototype that will provide early infrastructure for wide-area Web-based RT-DVS and also serve as a basis for further research in many-to-many overlay multicast with QoS management.

3. XOM Description

Overlay multicast allows Web-based RT-DVS to make efficient use of open protocols such as the Internet User Datagram Protocol (UDP) and IP as implemented across the Internet. XOM is an overlay protocol that provides many-to-many multicast for real-time distributed visual simulations. Its objective is to provide multicast service over a unicast network environment using existing, widely-deployed unicast protocols. From the multicast sender and receiver's point of view, each XOM behaves like an IP multicast network. This is achieved by a multicast relay agent located on the same subnet (LAN) as the application host computer. The agent software is called the XOM Relay (XOMR), and performs functions comparable to those of a router in a traditional IP multicast network, as indicated in Figure 2. The XOMR uses unicast protocols to communicate with its peers on other subnets and controls all aspects of the subnet's group communications. The next sections of this paper provide more details on the XOMR and results of preliminary demonstrations of our prototype in the laboratory and across the Internet.

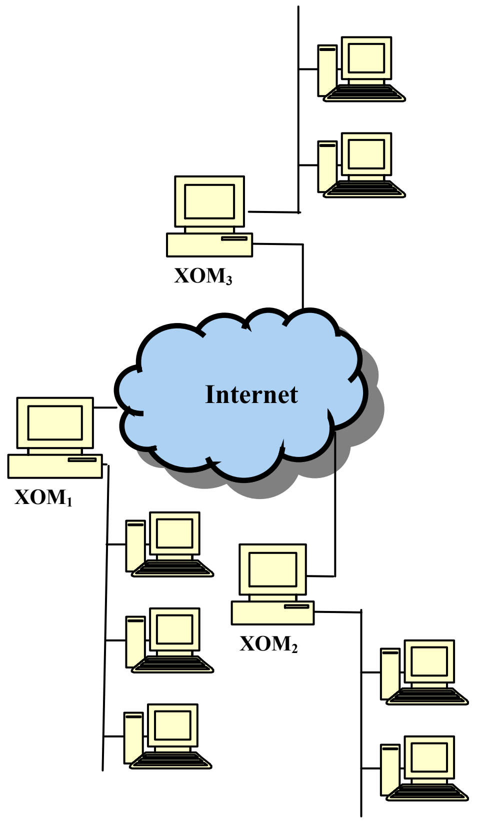


Figure 2. XOM on Subnets

3.1 Objective Architecture

The XOM architecture, as previously described in [7], is presented in Figure 3. It is designed to be highly modular so that module optimization and alternative strategies for each module can be prototyped for evaluation. The current version uses the UDP as the underlying transport protocol and offers services to the application layer for two different classes of services: Class 0 (best effort) and Class 1 (urgent traffic).

The approach employs a priority queuing strategy to give precedence to class 1 traffic and mark class 0 traffic as "discard eligible" in the event of network congestion. This approach is consistent with previous efforts in development of the Selectively Reliable Multicast Protocol (SRMP) [5, 8]. Since simple precedence does not provide error control, any form of desired error control must be added to the client application. The design assumption is that most messages are relatively small (less than 120 bytes) and the underlying network is able to deliver messages with a reliability greater than 99% and also has reasonable routing path stability on the order of minutes, diminishing the need for error control. If needed, highly reliable transport can be provided using SRMP or other reliable multicast protocol, as shown in Figure 3 as providing the application program interface.

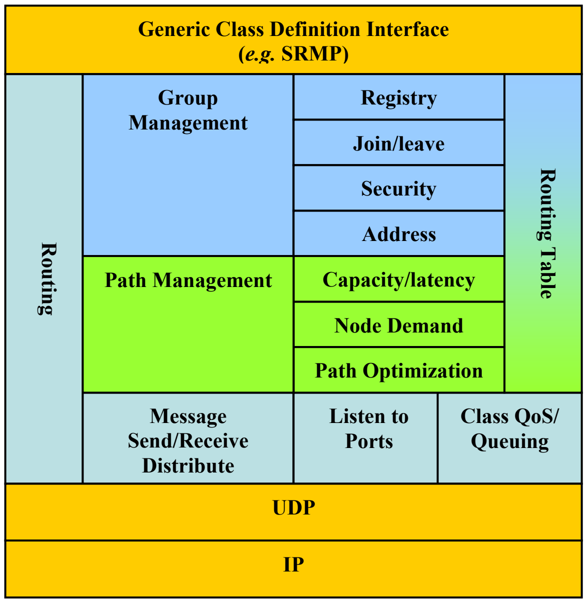


Figure 3. XOM Architecture

The XOM architecture uses an external registry authority to identify an XOMR host. The registry maintains a list of all XOMRs in the overlay and registered application users. This use of an external registry provides a level of security by establishing an authority that authorizes a source to be a sender which can be used by networked XOMR receivers for recognition of authorized senders in the network. The registry also maintains the public routable IP addresses of all XOMRs that are active in the overlay and subsequently is used to establish efficient overlay multicast paths among XOMRs. Once an XOMR host is connected, internal protocol mechanisms provide for path optimization among XOMR hosts and manage multicast group membership.

The core of XOM is provided in the Group and Path management modules. These activities provide the information for the routing table to be used in making message forwarding (routing) decisions. Receivers issue a request to join existing groups using a unique connection identifier that is pre-assigned by the registry. Using this approach, the RT-DVS application is able to control or specify which sources of information are of interest. Group membership control helps protect an individual application from receipt of unspecified or undesired information flows and also aids in minimizing overall network traffic load.

The group membership approach assumes that group definition is based on a specific application running behind an XOMR on the local area network. Multiple application instances are supported, each of which may have different group membership characteristics to include membership in multiple groups. It is also feasible for an application to have membership in more than one group. The architecture also implies no explicit set-up processing between the sender and the receivers prior to the establishment of group communications, other than that already attendant to IP multicast. The XOMR mechanism is required to pass the multicast group (IP/group tag) address to the XOMR of the associated receivers. The receivers' XOM must have established "filters" for the address prior to transmission in order to receive the data.

Our architecture uses a decentralized algorithm to construct an overlay by searching for partner XOMRs as nearest neighbors with which to form a connection in the overlay. This decentralized algorithm relies on information provided by the registry to the joining XOMR. In this process, we employ two mechanisms that contribute to global service guarantees and help manage the construction of path overlays. The first is that we put a limit on the out degree of an XOM using Bollobas' [9] definition of the degree of a vertex. That is, we do not allow the construction of more than *n* forwarding connections to other XOMs where *n* is determined by the resources of the XOM and access capacity of the underlying network. This serves to limit the processing demands and network access capacity of individual XOMs in the overlay and therefore establishes lower bounds on the performance. The second is that we create a threshold for the end-to-end overall path delay from a sending XOM to a destination XOM and only offer best effort above this threshold to joining XOMs that do not successfully find an existing XOM node that is adequate to maintain end-to-end path delay thresholds.

3.2 Prototype Implementation

Our open source prototype is available online at http://netlab.gmu.edu/XOM. The QoS features of the architecture are not implemented in the current prototype. As of this writing, the Registry function also is incomplete, so that group identification is accomplished by providing IP addresses of the participating XOMRs on the command line. We anticipate that the Registry function will become available by May, 2005. The software architecture we are using is shown in Figure 4.

Our XOMR is implemented in the Java language for maximum portability and ease of experimentation. In this design, only the central "router" module is computationally stressed. Therefore, we also have developed a C++ version of that module using the Java Native Interface (JNI), permitting higher performance on a given platform. We have tested this software on Windows, Linux, and Solaris systems. In principle, the

Java code should run on any platform for which a Java Virtual Machine is available; the C++ module would require porting to be used on any other platform. The Java implementation is extremely simple to use; it runs as a normal Java application, with all necessary parameters established by the command line parameters shown in Table 1. The C++ version is only a little more complex to use, requiring that the executable code with JNI bindings be loaded also.

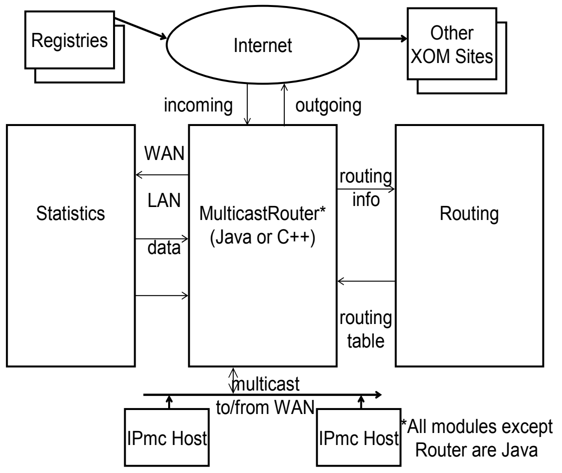


Figure 4. XOMR Prototype Software Architecture

registryAddress	InetAddress of registry, 0 if none
numberOfMulticastGroups	count of groups/ports we will support
numberOfPortsPerGroup	count of ports each group will support (non-overlapping)
lowestMCAddress	first group address to multicast from the subnet, dotted decimal notation (other addresses follow in sequence)
lowest port	first UDP port to multicast (each address will get one port in sequence)
routingUpdateInterval (optional)	time in ms between routing updates (default 10 s)
thisSubnetMaskBits (optional)	number of bits used for routing in subnet address (default 24)
useTCP (optional)	0 for UDP tunnels, 1 for TCP tunnels (default 0; 1 does not work yet)
partnerHostAddress (optional in future, when Registry becomes available)	zero to MAX_PARTNERS IP addresses, in dotted decimal format, to be used as partners without checking the registry

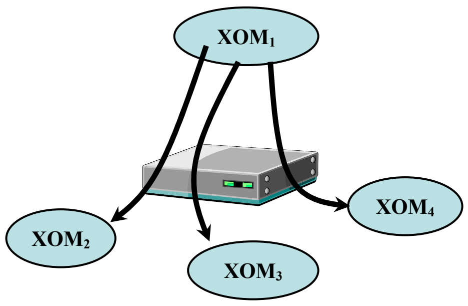
Table 1. XOMR Command Line Parameters

3.3 Laboratory Analysis of XOMR Prototype

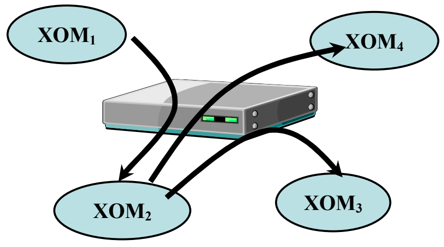
We have conducted a preliminary evaluation of our XOMR approach with the objective to demonstrate message throughput of the basic message routing functions which was initially presented in [7]. Our specific performance interests were to measure message throughput of the send/receive functions and to gain an understanding of the impact of the *n*-degree or the number *n* of reflect messages that might be achievable from an XOMR. We deem the *n*-degree factor a significant parameter as it represents the ability of the XOMR to replicate messages to multiple paths or channels, therefore a driving force behind overall path construction and path optimization.

The initial test environment was established in the George Mason University (GMU) C3I Center Networking and Simulation Laboratory. The test configuration consisted of four XOMRs, each operating on separate subnet LAN segments. These four segments were connected together via a single IP router to represent a WAN. Two test scenarios were conducted as represented in Figure 5. In Scenario 1, we tested the XOMR as a 3-degree routing function, where degree means number of paths required for the XOMR message replication and forwarding. In Scenario 2, we tested the XOMR as a 2-degree routing function. Notice that we also, in Scenario 2, required the data path to traverse the WAN router twice, which represents the typical type of access architecture that we would expect under XOMR. This scenario also gives us a measure for maximum throughput based on a single channel.

We used the source traffic generator described in [8] and varied the number of source objects in order to map performance against offered load. During test run start up, there is a high message rate as objects execute join commands. We executed each test run for 1800 seconds to ensure that we reached steady-state. At steady-state, each object represents approximately 3.1 messages per second of offered load with a message payload size of 120 bytes.



Scenario 1. XOMR *3*-degree



Scenario 2. XOMR 2-degree

Figure 5. XOMR Test Scenarios

The results of our experiments are presented in Figure 6 and Figure 7. Our area of interest for the results is for loss ratios of less than 1%. Under both scenarios, the XOMR message throughput was approximately 1300 messages per second. For the 2-degree network, this represented slightly greater than 400 simulated objects supported with delay averaging 30 msec. In the case of the 3-degree network, more than 300 source simulated objects were supported with message delay averaging 32.2 msec. In another system we have studied, reported in the paper by Morse *et al* [10], 4000 messages per second would

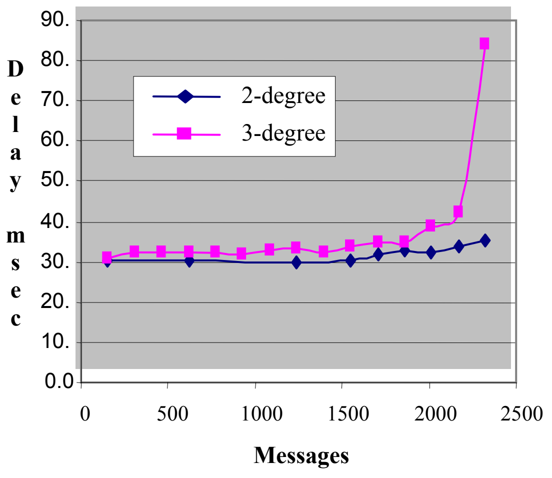


Figure 6. Mean Delay

support 10,000 objects. While we have not yet performed code optimization of the existing XOMR, our experiment has given us confidence that we are able to achieve desired throughput capabilities that will support a useful range of Web based services using overlay multicast.

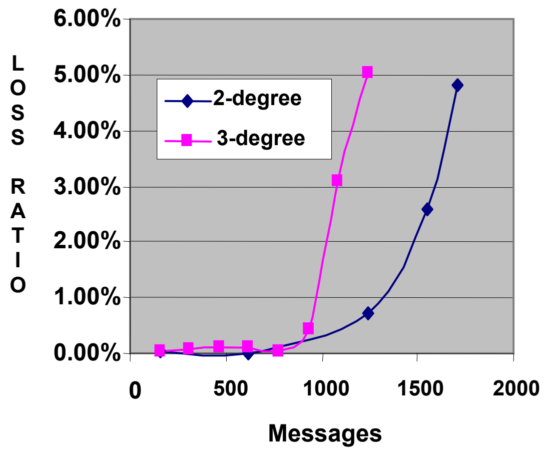


Figure 7. Loss Ratio

4. Network Demonstration of the Prototype

4.1 Wide Area Demonstration

Laboratory tests, while important, do not provide the credibility of actual application. This is especially true for wide-area applications; the LAN environment is a tame one when compared with the latency, variability of latency, traffic losses, and overall difficulty of running over a WAN. The initial public demonstration of XOM was held at the Interservice/Industry Training, Simulation and Education Conference (I/ITSEC) 2004. The sites involved were the conference floor in Orlando (both XMSF and DMSO booths) and three academic XMSF laboratories: NPS MOVES, Old Dominion University (ODU) Virginia Modeling, Analysis and Simulation Center (VMASC), and GMU C3I Center. The RT-DVS workload consisted of an HLA Federation supported by the Pitch pRTI 1516, which is multicast-capable, and the Web Service Interest Management (WSIM) prototype as described in [10, 14]. Simulated objects were naval vessels from the NPS SAVAGE library, visualized on a 3D viewer from NPSNET (http://www.npsnet.org). We had no difficulty sustaining synchronized steaming-in-circles behavior with ship models running at NPS, ODU, and GMU. The data rates involved were low: on the order of tens of messages per second per site. The project will conclude with a loading test to be performed by XMSF participant SAIC San Diego, which will characterize the useful capacity of XOMR when supporting an HLA federation.

4.2 XOMR Instrumented Characterization

Figure 8 shows the tier relationships between the nodes in the network and the two multicast groups used in the message exchange. The tiers are defined as: Simulation Federate, the WSIM Server (database), Client, Viewer, and the two multicast groups.

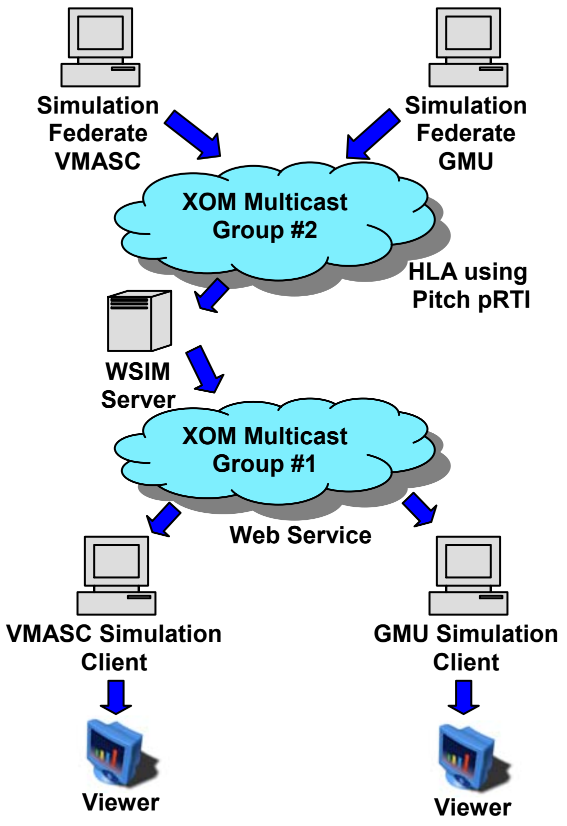


Figure 8. Tier Relationship for Multicast

The logical flow of messages is that the Federates publish to multicast group 2 using the XOMR multicast services and the HLA via Pitch's pRTI. The WSIM server (web service) listens to multicast group 2 and receives the published updates. The client, using TCP, establishes a tunnel connection to the WSIM Server announcing request for registered information and subscribes to multicast group 1 using the XOMR multicast service. The client then listens to multicast group 1 and provides the received information to the local viewer.

In our sample experiment, the vessel simulator federate represents a single simulation element or object. This allows for excellent understanding of the value of the multicast service as it relates to a single object in a visual simulation.

Figure 9 graphs the message inter-arrival time from a single Federate to the WSIM server. We used the OPNET ACE [3] application characterization tool to capture this data.

It is easily observed in Figure 9. that this particular simulation object generates uniformly distributed message at the rate of 4 messages per second. This message rate is consistent with our early model development for simulation objects [8] where we applied an average of 3.1 messages per second per object. In our experiment the simulation object is in a continuous movement state, thus generating uniformly spaced update messages. The measured standard deviation for the inter-arrival times is .04228 which supports the observation of a uniform distribution.

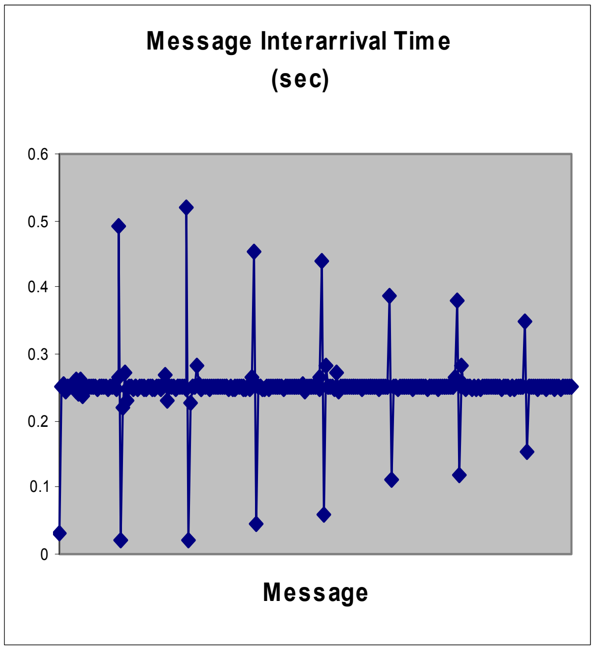


Figure 9. Federate Message Flow

In our early experiments using the NPS vessel simulation with the WSIM services, we relied on these message transfers using TCP connections. This entailed a TCP connection for each message and each tier in the information exchange. With the TCP connection, each message transfer generated 4 messages at the network layer for each element of the tier. Using the XOM multicast service, a message transfer entails a single message at the network layer and also provides reduced message flow among the elements of the overlay across the open network. At the small scale of this demonstration, the gain in using multicast is minimal. However, if we consider a federation operating over ten sites, the unicast approach generates 90 times as much WAN traffic as a single site, while the multicast approach generates only ten times as much traffic. In general the multicast traffic scales linearly with the number of sites, where unicast traffic scales as the square of the number of sites.

Another important observation from our experiments is that the message flow rate is never zero. This observation, along with the observed uniform distribution, provides important characteristics to be considered as we develop a class of service (priority) capability for the XOMR. As described above, our current draft architecture includes priority queuing mechanisms. These observations also impact global optimization of the overlay resources such as the network access capacity for each of the XOMs, link capacities, and number of intermediate XOMs in an overlay. We recognize that this experiment only represents an initial example of potential simulation environments, however, it does give us confidence that the XOMR has great potential for efficiently supporting distributed simulations over an open WAN.

The service available will of course be dependent of the underlying networks. Experience to date suggests that, across networks such as Internet2 and DREN, an XOM system using the current prototype XOMR could support tens of subnets with aggregate group traffic up to 5000 messages per second and 1 Megabit per second, while exhibiting latency under 100 milliseconds and jitter under 20 milliseconds. Scaling to higher traffic levels should be possible if the load is distributed across multiple multicast groups, assuming the underlying network has adequate capacity. Whether this projected performance is actually achieved remains to be demonstrated.

5. Conclusions

This paper describes an overlay multicast system that provides a many-to-many multicast service for distributed reality simulation environments. We have described our motivation for overlay multicast and provided an overview of the architecture of the XOM protocol. We also have presented the software design of our prototype, which has been used to demonstrate some of the key aspects of the architecture.

We deployed the XOMR prototype in our laboratory, using a network of four XOMRs to evaluate throughput characteristics under various loading conditions. We measured throughput in two different scenarios in order to evaluate throughput under routing conditions where the XOMR was required to perform message replication. Our results indicated performance levels showing great promise that overlay multicast can meet the needs of the RT-DVS environment, including Web-based distributed simulations.

We also have demonstrated a five node overlay across the public Internet supporting a Web services based real time simulation application. Our results from these experiments and demonstrations give us confidence that our approach will provide a networking capability that supports the use of Web services for RT-DVS and will allow these applications to be independent of underlying network protocols.

Many challenges remain in order to produce overlay networks with adequate efficiency and scalability as well as overlay management mechanisms that respond to the dynamic nature of underlying open networks. We need to better understand the network characteristics of the collaborative and distributed virtual simulation environments, as these characteristics must be accommodated in the XOMR. For the future, we plan to continue to refine and improve our Java based prototype with a Registry and graphic performance displays. We will also use the prototype for experimenting with various routing algorithms and service optimization strategies.

6. References

[1] Simulation Interoperability Standards Organization (SISO), http://www.sisostds.org/

[2] George Mason University Center for Command, Control, Communications, and Intelligence Network Laboratory (NetLab) XOM web site, http://netlab.gmu.edu/XOM/.

[3] OPNET Technologies, Inc., http://www.opnet.com/products/modules/ace/home.html

[4] Brutzman, D., M. Zyda, M., J.M. Pullen, and K.L. Morse, "Extensible Modeling and Simulation Framework (XMSF): Challenges for Web-Based Modeling and Simulation", US Naval Postgraduate School, 2002

[5] Pullen, J., "Reliable Multicast Network Transport for Distributed Virtual Simulation", *Proceedings of the 1999 IEEE Distributed Interactive Simulation and Real-Time Systems Workshop*.

[6] Braudes, R. and S. Zabele, "Requirements for Multicast Protocols", *IETF RFC 1458*, 1993.

[7] Moen, D., J. Pullen, and F. Zhao, "Implementation of Host-based Overlay Multicast to Support of Web Based Services for RT-DVS", *Proceedings of the Eighth IEEE Workshop on Distributed Simulation and Real-Time Applications,* 2004, pp. 4-11.

[8] Moen, Dennis and J.M. Pullen, "A Performance Measurement Approach for the Selectively Reliable Multicast Protocol for Distributed Simulation," Proceedings of the Fifth IEEE Workshop on Distributed Simulation and Real-Time Applications, 2001, pp.30-34.

[9] Bollobas, Bela, *Random Graphs*, 2nd Edition, Cambridge Press, NY, 2001.
[10] Morse, Katherine L., Ryan Brunton, J. Mark Pullen, Priscilla McAndrews, Andreas Tolk, and James Muguira K., “An Architecture for Web-Services Based Interest Management in Real Time Distributed Simulation”, *Proceedings of the Eighth IEEE Workshop on Distributed Simulation and Real-Time Applications,* 2004, pp. 108-1
[11] Pullen, J.M, R. Simon, F. Zhao and W. Chang, NGI-FOM over RTI-NG and SRMP: Lessons Learned, *Proceedings of the IEEE Fall Simulation Interoperability Workshop*, paper 03F-SIW-111, Orlando, FL, September 2003
[12] Chu, Yang-hua, S. G. Rao, S. Seshan, and H. Zhang, “A Case for End System Multicast,” *Proceedings of the ACM Sigmetrics*, June 2000.
[13] Wang, Wenjie, David Helder, Sughi Jamin, and Lixia Zhang, “Overlay Optimizations for End-host Multicast,” *Proceedings of the ACM NGC 2002,* p154-161.
[14] Morse, K., , J.M. Pullen, R. Brunton, and D. Drake, Web Services Interest Management, *Proceedings of the IEEE Fall Simulation Interoperability Workshop,* paper 04F-SIW-037, Orlando, FL, September 2004

Author Biographies

DENNIS MOEN is pursuing a PhD in Information Technology at George Mason University. He received his M.S. degree in electrical engineering from the University of Arizona. He is a licensed Professional Engineer and Chairperson of the Northern Virginia Chapter of the IEEE Communications Society. He is a Systems Engineer Principal at Lockheed Martin Corporation working in the area of survivable network architectures, network design and performance modeling. His research interests are network performance modeling and survivable network architectures.

J. MARK PULLEN is Professor of Computer Science at George Mason University, where he heads the Networking and Simulation Laboratory in the C3I Center. He holds BSEE and MSEE degrees from West Virginia University, and the Doctor of Science in Computer Science from the George Washington University. He is a licensed Professional Engineer, Fellow of the IEEE, and Fellow of the ACM. Dr. Pullen teaches courses in computer networking and has active research in networking for distributed virtual simulation and networked multimedia tools for distance education.