2000

Online Multicast Grouping for Dynamic Data Distribution Management

Morse, Katherine; Zyda, Michael

Katherine Morse and Michael Zyda Online Multicast Grouping for Dynamic Data Distribution Management, Proceedings of the Fall 2000 Simulation Interoperability Workshop, September 2000.

https://hdl.handle.net/10945/41535

ONLINE MULTICAST GROUPING FOR DYNAMIC DATA DISTRIBUTION MANAGEMENT

Katherine L. Morse
Science Applications International Corporation
10260 Campus Point Drive, MS B-1-E
San Diego, CA 92121
858-826-5442, 858-826-5112
katherine.l.morse@saic.com

Michael Zyda
MOVES Academic Group
Naval Postgraduate School
Monterey, CA 93943-5118
zyda@acm.org

KEYWORDS: HLA, Data Distribution Management, multicast, RTI

ABSTRACT

The High Level Architecture's Data Distribution Management (DDM) (Morse 1997, DoD 1998) services are the most recent in a succession of systems designed to reduce the amount of data received by individual simulations in large-scale distributed simulations. A common optimization in these interest management systems is the use of multicast groups for sending data to a selected subset of all potential receivers. The use of multicast has met with considerable success in this application. However, its use to date has relied on a priori knowledge of communication patterns between simulations and static assignment of multicast groups to these patterns. As larger, more complex, and less predictable simulations are built, the need has arisen for more efficient use of multicast groups as they are a restricted resource. This paper describes an implementation of online multicast grouping, compares the message delivery time of the resulting groupings against an offline grouping algorithm, and projects the performance impact on a production RTI.

1 INTRODUCTION

(Morse 2000b) describes two algorithms for performing offline grouping for dynamic DDM and analyzes their performance in terms of their ability to deliver required updates in a timely manner. In this paper we describe an online implementation of one of those algorithms and evaluate both the goodness of the groupings it produces and the potential impact of incorporating the algorithm in a production RTI. In section 2, we analyze the potential performance improvements for using multicast grouping. Section 3 describes the offline and online versions of the grouping algorithm. Section 4 compares the goodness of the results generated by both implementations and analyzes the potential impact of implementing online grouping in a production RTI. Section 4.2 outlines future work[1].

[1] Initial work on this project was funded under DARPA ASTT contract MDA9972-97-C-0023.

2 MULTICAST GROUPING

The most promising optimization identified to date is the use of multicast groups for routing data to a controlled subset of all member simulations in a simulation (Abrams, Watsen, and Zyda 1998; Calvin *et al.* 1995; Macedonia *et al.* 1995; Mastaglio and Callahan 1995; Rak and Van Hook 1996). The ultimate measure of effectiveness of any interest management system is the latency between sending a piece of data and an interested receiver getting it. Broadcast makes sending fast, but at the expense of time spent by the receiver discarding irrelevant data. Point-to-point ensures that receivers only get relevant data, but it requires determining the destination for the data and requires sending multiple copies of that data, slowing transmission. The use of multicast strikes a balance between broadcast and point-to-point by reducing the time to send and the amount of data received. Broadcast and point-to-point represent opposite ends of the send/receive time spectrum with various applications of multicast occupying the area in between. Even though multicast has the potential of improving communication time, it is not without its own challenges: multicast hardware currently supports a limited number of multicast groups, on the order of a couple thousand; the time to reconfigure multicast routers can be of the same order as the total allowable latency for message delivery (Mastaglio and Callahan 1995). As a result, most implementations using multicast to date have used static assignment of multicast groups, usually to fixed geographic regions[2]. These implementations have achieved good results, but ultimately they are limited in scale as well because they do not account for changing connection patterns between senders and receivers. The next step in optimization is dynamic multicast grouping that adapts to connection patterns.

[2]See (Macedonia *et al.* 1995) for an exception.

2.1 Connection Graphs

By virtue of regions, we know the destination of attribute updates before they are sent. Figure 2-1 illustrates the problem with a connection pattern graph. Federate f_1 sends the attribute update(s) represented by connection path c_1 to federates f_2 and f_3. Federate f_3 sends the attribute update(s) represented by connection path c_2 to federates f_2 and f_4. Federate f_1 also sends c_3 to f_4. The connection set, C, represented by this graph is $\{<f_1, c_1, (f_2, f_3)>, <f_1, c_3, (f_4)>, \{<f_3>, c_2, (f_2, f_4)\}$. Note that a connection path doesn't represent a single message, but all updates of some set of object attributes.

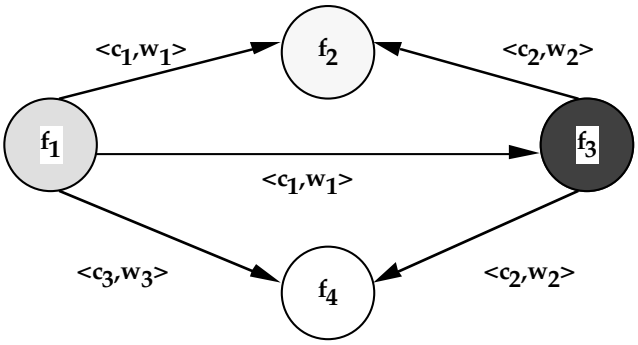

Figure 2-1. Example Connection Pattern Graph

This problem is similar to the clique-covering problem (Papadimitriou 1994) Intuitively, if a clique exists in the connection pattern between a set of nodes for a set of connections, assigning a multicast group to these nodes for these connections results in optimal routing. In general, however, we cannot expect to be fortunate enough to have many cliques in the connection pattern graph. The algorithm's accuracy is augmented by weighting the arcs in the graph with the data transfer frequency over the arc, known as the connection weight.

The multicast grouping algorithm uses maximum tolerable latency, t_{max}, as its cost measure. The goal is to group m connections, $c_1, \ldots c_m$, into no more than n multicast groups,

g_1, ... g_n, such that no communication arrives at its receiver in greater than t_{max} time, if physically possible. The parameters to the algorithm are:

- Number of available multicast groups (n)
- Maximum tolerable latency (t_{max})
- Time to send (t_s) - We assume that t_s is roughly the same for all federates.
- Time to receive (t_r) - We assume that the time to discard an irrelevant message is the same as the time to receive a relevant one; also that t_r is roughly the same for all federates.
- Time to propagate message through the network ($t_p(f_i,f_j)$) - measured between federate f_i and f_j, the publisher and subscriber of the connection, respectively.

The algorithms begin by assuming point-to-point communication for all messages and falls back to this position when the network and scenario make multicast grouping impossible, i.e. when $t_s + t_p + t_r + t_q > t_{max}$ for some connections.

The decision to add a connection, c, to a multicast group is based on the expected impact on t_{max} of all connections and federates already in the group. If connection c is sent to k receivers point-to-point, t(c), the time from the beginning of c's sending to the end of the last receiver's receipt is bounded by:

$$k \bullet t_s + \max(t_p(f_i,f_j)) + \max(t_q(f_j)) + t_r$$

Equation 2-1. Time Bound for k Point-to-point Communications

This bound is based on the worst case assumption that the k^{th} communication has the longest t_p and the longest t_q.

We assume t_s and t_r are fixed, uniform, and roughly equal. Time to propagate the update, $t_p(f_i,f_j)$, is assumed to be fixed, but different and measurable between source and destination, federates i and j. Time in the queue, t_q, varies depending on the number and frequency of messages received.

The use of multicast improves latency for a message by reducing the average serial delay send time, t_{ds}, for the k copies as given in Equation 2-2. In fact, t_{ds} becomes exactly t_s when c is put in a multicast group. Furthermore, this improvement is achieved each time a message is sent for a connection.

$$t_s \le t_{ds}(c) \le k \bullet t_s$$

Equation 2-2. t_{ds} for c not in a Group

However, adding connections to an existing multicast group may cause extraneous communication at some receivers, increasing t_q for some valid communications. For example, putting all the connections in Figure 2-1 would result in the following extraneous communications:

- c_3 to f_2 and f_3
- c_1 to f_4
- c_2 to f_1

This simple observation illuminates a much larger point. We are addressing *potential* performance improvements. There are some circumstances under which we cannot improve over the performance of point-to-point.

3 GROUPING ALGORITHMS

3.1 The Input-Restricted Largest Outgoing Connection (IRLOC) Algorithm

The input-restricted largest outgoing connection (IRLOC) (Morse 2000b) algorithm seeks to minimize both t_{ds} and t_q to produce better average results than its predecessor, the largest outgoing connection (LOC) algorithm. This algorithm recognizes three facts about adding a connection to an existing group:

- Any receivers of the connection who are not already in the group will receive additional connections equal to the sum of the connections already in the group, the group weight.
- Any group members who are not receivers of the connection will receive the additional weight of the connection.
- Assuming that $t_s = t_r$, improvements in average message delivery time created by sending a connection via multicast are directly offset by the "negative weight" created by the first two facts.

The IRLOC performs the following steps:

1. Calculate the positive cumulative effect of each connection, $(k - 1) \bullet w$.
2. Add the receivers of the connection with the largest cumulative effect; in the event of a tie, add the lowest numbered such connection.
3. Add the next largest connection such that a) the input weight of the current group members does not exceed t_{max} by the addition of the connection weight, b) the input weight of the connection's receivers not already in the group does not exceed t_{max} by the addition of the group weight, c) the positive cumulative effect is greater than the negative weight. Note that the positive cumulative effect is only a function of the connection, while the negative weight is a function of the connection and the current state of the group.
4. Repeat step 3 with the remaining connections. Halt when all connections are assigned or all multicast groups are used.

3.2 The Online Grouping Algorithm

The online, distributed IRLOC algorithm is built on top of a baseline prototype (Morse 2000b) implemented in the MESSENGERS mobile agents system (Bic 1996). The baseline prototype implements a minimal subset of RTI necessary to test DDM. The online IRLOC algorithm integrates the baseline prototype with the basic structure of the IRLOC algorithm. The distributed algorithm operates with degraded information for several reasons. First, the information about connections and incoming weights is distributed among the federates, and collecting it would be prohibitive in a very large scale distributed simulation. If the simulation were small enough to be able to collect all this data at a central point and still make timely grouping decisions, it wouldn't need multicast grouping. Second, this same information is changing in real time. Regions and region intersections are changing while the grouping decisions are being made. Even if the algorithm had access to global information when it started grouping, there is a non-zero probability that the information would be out of date by the time the grouping completed. Finally, the MESSENGERS system doesn't provide a straightforward, timely mechanism for passing dynamic data structures to Messengers. Some simplifying assumptions have been made which account for this. In a production system this final constraint could be relaxed.

The online grouping algorithm is triggered by the discovery of a connection or connections. A grouping Messenger is injected which begins searching for a potential group. The grouping Messenger searches three places in the following order:

1. on the init node on the local machine;
2. at the multicast server where it may find an unused group;
3. at most one hop from the multicast server at another machine.

If the grouping Messenger finds an unused group at the multicast server, it marks the group as taken to the requesting federate's machine and "carries" the group home. This is how groups migrate away from the multicast server. If the group is unused, there's no need to check for overflow and the group can be used immediately. Before a partially used group can be taken from another federate, it must be

checked for overflow. The grouping Messengers carries the connection weight and connection receivers with it. The current group weight and members is always stored with the group at its current init node. All of this information is consistent with the IRLOC algorithm and is always up to date. However, the IRLOC algorithm also makes use of the current incoming connection weights of both the current group members and the connection's receivers. Here is where slightly degraded information is used. Instead of having the current incoming weights of all the connection's receivers, the grouping Messenger carries the last known, largest incoming weight of all the receivers. The incoming weights of receivers are piggybacked on subscription regions, so they may be out of date due to subsequent subscriptions. Instead of the current incoming weights of all the group's members, the group is stored with the last known, largest incoming weight of any of all the group's members.

If the grouping succeeds, the group's weight and member list is updated. The grouping is reported to the requesting federate which changes its connectivity and adjusts its outgoing connection weight down. It also injects "join" Messengers for all the connection receivers who were not previously members of the group. In this system, this Messenger only informs the receiver to adjust its incoming weight upward to account for other traffic from the group and to add to a reference count for this group. If multicast hardware were available, this Messenger would also be the trigger for the receiver to issue the appropriate system calls to join the multicast group.

4 RESULTS

There are two measures of merit for a DDM implementation, latency and overhead. As described in Section 2, we wish to minimize the average message delivery time while not incurring too much overhead cost.

4.1 Grouping Results

The first experiments performed were to determine the goodness of the groupings generated by the online grouping algorithm relative to the offline version. These experiments were conducted with random connection sets. For each desired random connection set we generated 10 sets of the given size, using different random seeds, and averaged the 10 results. This is to minimize the impact on the results of randomly pathological connection sets. Table 4-1 lists the parameters of the random connection sets generated.

Table 4-1. Random Connection Set Tests

f	10
$t_r = t_s$	10 milliseconds[3]
t_p	10 milliseconds between all federates
t_{max}	1000 milliseconds
(n, m)	(5, 2), (5, 3), (6, 2)

Since the online grouping algorithm can only be run in real time with the MESSENGERS system underneath, this comparison test required manually generating regions and DDM API calls whose resulting region intersections produce the connection sets listed in Table 4-1. The groupings produced in this way were manually edited into connection set files and run through the offline simulator. In the online configuration there is no way to create the entire connection set statically. As soon as a connection or connections are detected at any federate, the RTI component at that federate triggers grouping. This required writing auxiliary Messengers which locate existing multicast groups and add new receivers to the group when they subscribe for a connection which has already been assigned to the group in question.

[3] Hoare and Fujimoto (Hoare and Fujimoto 1998) measured these values for RTI 1.3.

Even with degraded information about the input weights of the federates, the online grouping algorithm compared quite favorably to the offline IRLOC algorithm. In over half the cases, seventeen cases, the online algorithm generated the same solution as IRLOC. In six of the cases, it generated a better average message delivery time. In one case, the degraded information about input weights caused the online algorithm to generate too conservative a solution. In two cases, the order in which connections were discovered affected the order in which they were added to groups, i.e. early addition of a connection prevented later addition of a different connection which would have produced a better result. In one case, the fact that the grouping Messenger was restricted to only looking one hop from the multicast server prevented it from putting a potential connection into an existing group. In three cases, the grouping was affected by both of the previous two factors, ordering and restricted hops.

4.2 Runtime Performance Results

The experiments described in Table 4-2 are designed to test the potential impact of integrating online grouping with a production RTI using relative measures between the online grouping algorithm, the baseline prototype, and an actual RTI implementation with DDM, RTI 1.3 v4. The experiments use the benchmark algorithm described in (Morse 1999b). When interpreting the results, it's critical to remember that the baseline prototype and the online grouping algorithm implement the barest minimum of HLA functionality necessary to test the hypothesis with almost no error checking. RTI 1.3 is a robust, fully-compliant HLA 1.3 implementation with all the specified service groups and error checking, and the overhead implied by that.

Table 4-2. Runtime Experiments

#	(federates, regions)	Δr /min.	i	Δi /min.
1	(2,50), (5, 200), (10,1000)	0	r/25, r/10	0
2	(2,50), (5, 200), (10,1000)	r	r/25, r/10	0
3	(2,50), (5, 200), (10,1000)	r	r/50, r/25	i

Experiment 1 tests the impact of intersection calculations on initialization time. It establishes a basis for projection of performance of the online grouping algorithm in an actual implementation. The baseline prototype and RTI 1.3 are used because the online grouping algorithm is built on top of the baseline prototype, while the baseline prototype has an architecture for DDM which closely models the architecture of RTI 1.3.

The average per federate initialization times for each of the federates in the RTI 1.3 tests are listed in Table 4-3. Separate tests verified that the growth in initialization times is due primarily to a larger number of federates, not to a larger number of regions. The change in initialization time for grouping was calculated as the difference in initialization time between the online grouping algorithm and the baseline prototype. Given that average per federate initialization time increase is more than three orders of magnitude smaller than the initialization time without grouping, using grouping has no appreciable impact on initialization.

Table 4-3. Initialization Times

f	r	Δr/ min	i	Δi/ min	RTI 1.3 (sec.)	Change for Grouping
2	50	0	2	0	13.190171	.012
2	50	0	5	0	13.103029	.008881
5	200	0	8	0	24.895595	.015031
5	200	0	20	0	24.134336	.010996

10	1000	0	40	0	132.050392	.009019
10	1000	0	100	0	129.133358	.005554
Averages					56.251147	.010247

Experiment 2 tests the impact of region changes without any intersection changes. As discussed in Section 3.2, this should impact CPU usage, but not severely since the system should recognize that connectivity hasn't changed. Since regions are uniformly assigned to federates, Δr per federate = r/f which ranges from 25 to to 100. Here the methodology is to determine if the RTI can do its job without robbing the federates of the CPU cycles they need to do their job. Although the Sun Sparc 5s used in the experiment are slightly underpowered compared to platforms typically used for HLA-based simulations, the RTI performed fairly well. Each experiment is run for 7 minutes with 10 loops per second for a total of 4200 loops. During each loop, the federate code performs all the calculations it requires and the remainder of the time in the loop allocated for the RTI to perform its functions. A "bad" loop is one in which the RTI fails to complete all its processing in the remaining loop time allocated to it. Across all six tests in experiment 2, the RTI only suffered an average of 2.6% bad loops. Running the benchmark algorithm with the online grouping algorithm and the baseline grouping algorithm only resulted in an average of 2173 msec more time taken by the RTI across a 7 minute period; approximately .5 msec per .1 sec loop. That's less time than it takes to receive a single extraneous message that would have been delivered without multicast!

Experiment 3 tests the impact of region changes with intersection changes. This should impact CPU usage more severely than experiment 2 since connectivity changes must be made. Predictably, the RTI produced more bad loops for experiment 3 than for experiment 2, 5.7% vs. 2.6%. However, the grouping algorithm only resulted in an average of 384 msec more time. The fact that this is lower than the time for experiment 2 are initially surprising, but the numbers are so small compared to the measurable resolution that even small perturbations in the CPU load or network load on these non-dedicated machines can result in proportionally large differences.

For the sake of completeness, the additional time for all the experiments with the online grouping algorithm and the baseline algorithm were recorded and averaged. The average additional time was 2596 msec or .62 msec per loop.

All of this is overshadowed by the time it takes to reconfigure multicast groups in routers. According to (IETF 1997) and (Cisco 1999), joining a multicast group across a LAN can take no time at all, while leaving a multicast group across a WAN may take on the order of 260 seconds. In summary, it is not the time it takes to calculate the multicast groups which is the impediment to dynamic multicast grouping as has been asserted in the past, but the time it takes to change the groups in the routers.

5 FUTURE WORK

5.1 Maintaining Incoming Weight Information

The offline grouping algorithms always have perfect global knowledge of the incoming weights of all federates. And the algorithms update this information as they build groups. The distributed online grouping algorithm has a much less consistent view of this information. A receiver could form many new connections and be added to other groups in between the time when it sends its incoming weight to a sender with its subscription region and the time when the sender begins to form a group. The accuracy of the offline grouping algorithm could be improved by periodically updating

incoming weight information to receivers. One possibility is to keep track of the incoming weight sent with each subscription region as well as the lowest reported such weight. When the difference between the lowest reported weight and the current incoming weight reaches a threshold, an auxiliary Messenger with a new incoming weight would be dispatched to update all senders holding the subscription region. The threshold would have to be determined experimentally, trading off the cost of the additional overhead of tracking reported incoming weights against the cost of the extraneous messages resulting from inaccurate incoming weight information.

5.2 Ungrouping

The focus of this research has been on developing groups. Connections are dropped from groups and groups are dissolved when all the receivers have dropped out. However, the departure of only one or two receivers may adversely affect the effectiveness of the grouping. In the extreme case the cost function could be recalculated every time a receiver drops out. Doing so would require keeping complete information about the connections in the set, including the identities of all the required receivers of each connection. The recalculation would be very expensive. Like the grouping algorithms, ungrouping could almost certainly benefit from heuristic approaches.

5.3 The Real World

Finally, the true test of this research would be to implement the distributed IRLOC algorithm in a production RTI and test it in a very large scale, dynamic virtual environment.

6 CONCLUSIONS

As the size of distributed simulations grow, unwanted data received by member simulations will continue to grow as a limiting factor. Multicast has been identified as a highly effective and efficient tool for controlling the delivery of unwanted data, but multicast groups are a limited resource. Static assignment of multicast groups to particular geographic regions and data types have yielded positive results, but may not be extensible to very large simulations or simulations which exhibit a large degree of chaotic clustering. We have taken major steps toward dynamic assignment of multicast groups in the context of the HLA's DDM services. We have demonstrated that it is feasible to implement dynamic multicast grouping in a production RTI, but use of dynamic grouping rests on the ability to quickly reconfigure multicast hardware.

7 REFERENCES

Abrams, H.; K. Watsen; and M. Zyda. 1998. "Three-Tiered Interest Management for Large-Scale Virtual Environments." In *Proceedings of 1998 ACM Symposium on Virtual Reality Software and Technology (VRST'98,* (Taipei, Taiwan).

Bic, L.; M. Fukuda; and M. Dillencourt. 1996. Distributed Computing using Autonomous Objects. IEEE Computer, 29(8), August 1996.

Calvin, J.; D.P. Cebula; C.J. Chiang; S.J. Rak; and D.J. Van Hook. 1995. "Data Subscription in Support of Multicast Group Allocation." In *13th Workshop on Standards for the Interoperability of Distributed Simulations* (Orlando, FL, September) 367-369.

Cisco IOS 12.0 Solutions for Network Protocols Volume 1: IP. Cisco Press. 1999.

Department of Defense High Level Architecture Interface Specification, Version 1.3, DMSO, April 1998, available at http://hla.dmso.mil.

Hoare, P.; and R. Fujimoto. 1998. "HLA RTI Performance in High Speed LAN Environments." In *Proceedings of the 1998 Fall Simulation Interoperability Workshop*. (Orlando, FL, September). 501-510.

IETF Network Working Group. Internet Group Management Protocol, Version 2, RFC 2236. Available at http://rfc.fh-koeln.de/rfc/html/rfc2236.html, November 1997.

Macedonia, M.; M. Zyda; D. Pratt; and P. Barham. 1995. "Exploiting Reality with Multicast Groups: a Network Architecture for Large Scale Virtual

Environments." In *Virtual Reality Annual International Symposium '95*. 2-10.

Mastaglio, T.W.; and R. Callahan. 1995. "A Large-Scale Complex Virtual Environment for Team Training." *IEEE Computer 28*, no. 7 (July): 49-56.

Morse, K.L.; and J.S. Steinman. 1997. "Data Distribution Management in the HLA: Multidimensional Regions and Physically Correct Filtering." In *Proceedings of the 1997 Spring Simulation Interoperability Workshop* (Orlando, FL, March). 343-352.

Morse, K.L.; L. Bic; M. Dillencourt; and K. Tsai. 1999. "Multicast Grouping for Dynamic Data Distribution Management." In *Proceedings of the 1999 Society for Computer Simulation Conference*. (Chicago, IL, July).

Morse, K.L; L. Bic; and M. Dillencourt. 1999b. "Characterizing Scenarios for DDM Performance and Benchmarking RTIs." In *Proceedings of the 1999 Spring Simulation Interoperability Workshop*, March 1999.

Morse, K.L; L. Bic; and M. Dillencourt. 2000. "Interest Management in Large Scale Virtual Environments." MIT Presence, March 2000.

Morse, K.L. 2000b. "An Adaptive, Distributed Algorithm for Interest Management." Ph.D. Dissertation, University of California, Irvine, May 2000.

Morse, K.L.; M. Zyda. 2000b. "Multicast Grouping for Dynamic Data Distribution Management." In *Proceedings of the 2000 Distributed Simulation - Real Time Workshop*. (San Francisco, CA, August).

Papadimitriou, C.H. 1994. Computational Complexity. Addison-Wesley, New York. Pg. 190.

Rak, S.J.; and D.J. Van Hook. 1996. "Evaluation of Grid-Based Relevance Filtering for Multicast Group Assignment." In *14th Workshop on Standards for the Interoperability of Distributed Simulations* (Orlando, FL, September) 739-747.

AUTHOR BIOGRAPHIES

KATHERINE L. MORSE is a Senior Computer Scientist with SAIC. She received her B.S. in mathematics (1982), B.A. in Russian (1983), M.S. in computer science (1986) from the University of Arizona, and M.S. (1995) and Ph.D. (2000) in information & computer science from the University of California, Irvine. Dr. Morse has worked in industry for over 20 years in the areas of simulation, computer security, compilers, operating systems, neural networks, speech recognition, image processing, and engineering process development. Her Ph.D. dissertation is on dynamic multicast grouping for Data Distribution Management, a field in which she is widely recognized as a foremost expert.

MICHAEL ZYDA is a Professor in the Department of Computer Science at the Naval Postgraduate School, Monterey, California. Professor Zyda is also the Chair of the NPS Modeling, Virtual Environments and Simulation Academic Group. Since 1986, he has been the Director of the NPSNET Research Group. Professor Zyda's research interests include computer graphics, large-scale, networked 3D virtual environments, computer-generated characters, video production, entertainment/defense collaboration, and modeling and simulation. He is known for his work on software architectures for networked virtual environments.
Professor Zyda was a member of the National Research Council's Committee on "Virtual Reality Research and Development". Professor Zyda was the chair of the National Research Council's Computer Science and Telecommunications Board Committee on "Modeling and Simulation: Linking Entertainment & Defense". From that report, for the Deputy Assistant Secretary of the Army for Research and Technology, Professor Zyda drafted the operating plan and research agenda for the USC Institute for Creative Technologies (ICT).

Professor Zyda is a member of the National Research Council Committee on Advanced Engineering Environments. Professor Zyda is

also a Senior Editor for Virtual Environments for the MIT Press quarterly PRESENCE, the journal of teleoperation and virtual environments. He is a member of the Editorial Advisory Board of the journal Computers & Graphics. Professor Zyda is a member of the Technical Advisory Board of the Fraunhofer Center for Research in Computer Graphics, Providence, Rhode Island.