Calhoun: The NPS Institutional Archive

DSpace Repository

Faculty and Researchers | Faculty and Researchers' Publications

2000

Multicast Grouping for Data Distribution Management

Katherine Morse; Zyda, Michael

Katherine Morse and Michael Zyda “Multicast Grouping for Data Distribution Management,” Proceedings of the Computer Simulation Methods and Applications Conference, October 2000.
http://hdl.handle.net/10945/41543

Downloaded from NPS Archive: Calhoun

http://www.nps.edu/library

Calhoun is a project of the Dudley Knox Library at NPS, furthering the precepts and goals of open government and government transparency. All information contained herein has been approved for release by the NPS Public Affairs Officer.

Dudley Knox Library / Naval Postgraduate School
411 Dyer Road / 1 University Circle
Monterey, California USA 93943

MULTICAST GROUPING FOR 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 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[1]. This paper presents two algorithms for performing grouping, and the message delivery time improvements resulting from applying the algorithms to selected data sets.

1 INTRODUCTION

The primary goal of the High Level Architecture (HLA) (DMSO 1998; Morse and Steinman 1997) Data Distribution Management (DDM) services is to reduce the amount of data received by federates. Using Declaration Management (DM) services, federates specify classes of data they wish to receive. However, in federations with large numbers of objects or objects generating large numbers of updates, DM services may be insufficient to reduce data delivery to the degree necessary for federates to receive and process updates in a timely manner. In these circumstances federates may use DDM services to further limit the individual updates received. Section 2 provides a more detailed overview of DDM.

[1] (3Com) lists a limit of 6K, the highest number identified while STOW (VanHook) had a hardware limit of approximately 1,000. A typical workstation NIC has only a few (Abrams).

The DDM services specified for the HLA are the latest in a succession of data reduction mechanisms for large scale distributed simulations. These mechanisms are referred to variously as interest management, relevance filtering, and data subscription. See (Morse 2000) for an extensive survey of previous systems. In (Morse 2000) we also demonstrate that most interest management systems to date have been purposely built with relatively static architectures and static specification of filtering capabilities. The current trends for interest management systems are toward:

- Distributed and dynamic architectures
- Flexible, general purpose specification of filtering expressions
- Optimization to improve overhead of the interest management itself, especially through the use of multicast.

This paper focuses on improving the performance of DDM through assignment of multicast groups based on a cost function. In section 3 we analyze the potential performance improvements for using multicast grouping. Section 4 describes our first multicast grouping algorithm, the various simulation tools we built to test our hypotheses, and the results of experiments with the algorithm and tools. Section 6 outlines the remaining work on this project[2].

2 DDM OVERVIEW

The fundamental Data Distribution Management construct is a *routing space*. A routing space is a multidimensional coordinate system through which federates[3] either express an interest in receiving data (subscribe) or declare their intention to send data (publish). These intentions are expressed through:

- *Subscription Region*: Bounding routing space coordinates that narrow the scope of interest of the subscribing federate[4].
- *Update Region*: Bounding routing space coordinates that are guaranteed to enclose an object's location in the routing space.

Both subscription and update regions can change in size and location over time as a federate's interests change or an object's location in the routing space changes.

An object is discovered by a federate when at least one of the object's attributes comes into scope for the federate, i.e. if an only if:

- the federate has subscribed to the attribute
- the object's update region overlaps the federate's subscription region.

DDM enable federates to specify by object class and attribute name the types of data they will send or receive, while also narrowing the specific instances of data. Each federate decides which of the federation routing spaces are useful to them and defines the portions of those routing spaces that specify *regions,* or logical areas of interest particular to the federate, by putting bounds (*extents*) on the dimensions of the selected routing space.

Specifying a subscription region, the federate tells the Run Time Infrastructure[5] (RTI) it is interested in data which fall within the extents of the region specified by that federate. Specifying an update region and associating that update region with a particular object instance is a contract from the federate to the RTI that the federate will ensure that the

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

[3] A federate is a member simulation of a distributed simulation referred to as a federation.

[4] Regions in a multidimensional routing space do not necessarily map to physical geographical regions. A region in a routing space should be thought of as an abstract volume with any number of dimensions, e.g. radio channels.

[5] An RTI is an implementation of the High Level Architecture.

characteristics of the object instance which map to the dimensions of the routing space fall within the extents of the associated region at the time that the attribute update is issued. This implies that the federate is monitoring these added characteristics for each of the attributes owned by the federate. As the state of the objects change, the federate may need to either adjust the extents on the associated regions or change the association to another region.

Figure 2-1 shows one update region (U1) and two subscription regions (S1, S2) within a two dimensional routing space. In this example, U1 and S1 overlap so attribute updates from the object associated with U1 will be routed to the federate that created S1. In contrast U1 and S2 do not overlap so attributes will not be routed from the federate that created U1 to the federate that created S2.

When an update region and subscription region of different federates overlap, the RTI establishes communications connectivity between the publishing and subscribing federates. The subscribing federates each receive only the object class attributes to which they subscribed, although they may receive individual updates outside their subscription region depending on the precision of the routing space implementation. In figure 1, S1's federate will receive attribute updates from the object associated with U1 because their regions overlap, even though the object itself is not within S1.

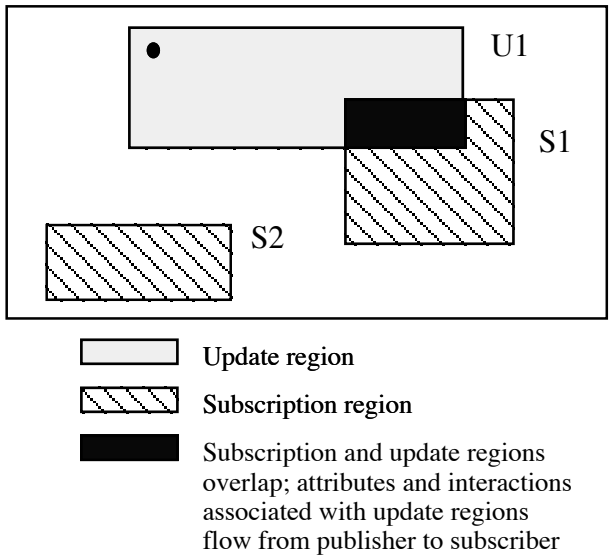

Figure 2-1. Two-dimensional Routing Space Example

Each federate can create multiple update and subscription regions. Update regions are associated with individual objects that have been registered with the RTI. A federate might have a subscription region for each sensor system being simulated.

For the sake of simplicity we have described regions as n-dimensional rectangles up to this point[6]. In fact, they are defined as sets of extents, or sets of n-dimensional rectangles. Regions which are not logically rectangular can be approximated by sets of smaller rectangles.

3 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

[6] The most common application of regions is to geographical 3-space, but the concept of regions is not limited to this.

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[7]. These implementations have achieved good results, but ultimately they are limited in scale as well because they do not account for changing connection graphs between senders and receivers. The next step in optimization is dynamic multicast grouping that adapts to connection patterns.

3.1 Connection Graphs

By virtue of regions, we know the destination of attribute updates before they are sent. Figure 3-1 illustrates the problem with a connection graph. A connection represents an attribute update stream originating from one federate and received by one or more other federates, that is updated with some frequency[8]. A connection may also be thought of as the result of the overlap of a single update region with one or more subscription regions. Connections are labeled to differentiate between potentially multiple update streams between the same set of sender and receivers. Federate f_1 sends the attribute update(s) represented by connection c_1 to federates f_2 and f_3. Federate f_3 sends the attribute update(s) represented by connection 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 $\{< c_1, w_1, f_1, (f_2, f_3)>, < c_2, w_2, f_3, (f_2, f_4)>, < c_3, w_3, f_1, (f_4)>\}$.

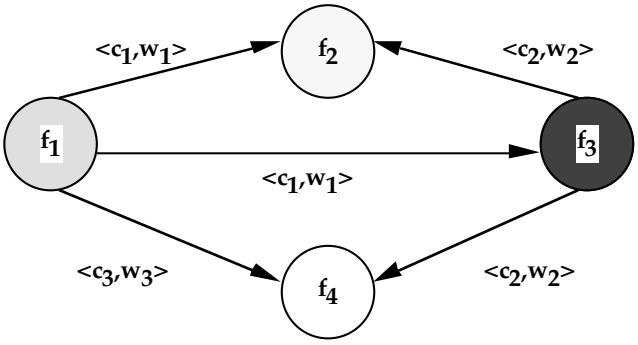

Figure 3-1. Example Connection Graph

Note that a connection doesn't represent a single message, but all updates of some set of object attributes.

This problem is similar to the clique-covering problem. Intuitively, if a clique exists in the connection graph 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 graph. The algorithm's accuracy is augmented by weighting the arcs in the graph with the data transfer

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

[8] In fact, connection graphs are hypergraphs because a connection's edges connect with multiple nodes. However, an illustration of such a hypergraph is more complex and obscures the input weight to the individual node federates. The importance of the latter will be explained later.

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 n connections, c_1, ... c_n, into no more than m multicast groups, g_1, ... g_m, 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 (m)
- 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 $t = t_{ds} + t_p + t_r + t_q > t_{max}$ for some connections, where tds represents the delay in sending caused by sending point-to-point, and t_q represents the time in the receiver's queue.

The decision to add a connection, c, to a multicast group is based on the expected negative impact on t of all connections and federates already in the group, as well as the positive impact on t(c) accrued by sending c via multicast, where t(c) is the time from the beginning of c's sending to the end of the last receiver's receipt. If connection c has k receivers and individual updates are sent point-to-point, t(c) is bounded by:

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

Equation 3-1. 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. If c is sent via multicast, t_{ds} is reduced to $t_{\backslash s}$.

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.

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 3-1 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.

4 GROUPING ALGORITHMS

4.1 The Largest Outgoing Connection (LOC) Algorithm

The first algorithm built and tested is referred to as the "largest outgoing connection" (LOC) algorithm. The LOC of any node is the connection originating at that node whose

connection weight, w, multiplied by the number of receivers, k, is the largest. We use this measure because $t_{ds} = k \bullet t_s \bullet w$ is the total sending delay time for point-to-point, which the quantity we wish to reduce across the entire federation.

The algorithm performs the following steps for each available multicast group:

1. Set the group's weight to 0.
2. Calculate the LOC of each node.
3. Select the LOC for the entire graph; in the event of a tie, select the lowest numbered such connection.
4. Test that adding the weight of the selected connection to the current weight of the group will not exceed t_{max}. If it does, remove this connection from consideration for this multicast group and return to step 3.
5. Add the selected connection's sender and all its receivers to the multicast group. Add the connection's weight to the group weight.
6. Calculate a new LOC for the sender's node.
7. Repeat steps 3 through 6 with the modification that new connections are only considered which originate with current group members. Halt when all connections are assigned or there are no connections left whose addition will not cause the group to exceed t_{max}.

4.2 Simulating Algorithm Goodness

To test the goodness of the results of the algorithms we built a discrete event simulation that takes as input a configuration file specifying:

- f (the number of federates)
- t_p (matrix of the propagation times between each pair of federates)
- t_r
- t_s
- t_{max}
- a connection set with connections assigned either to multicast groups or to point-to-point communication.

The simulation simulates 10 seconds of updates at millisecond resolution according to the connection set and measures:

- Average message queue time
- Average message delivery time
- Number of messages
- Number of late messages
- Average queue length

If an update is sent point-to-point, individual copies of the update are sent to multiple receivers at intervals of t_s. Updates sent via multicast are sent simultaneously.

4.3 Testing with Random Connection Sets

We performed experiments with the random connection sets listed in
Table 4-1. n and m were necessarily kept small to compensate for the combinatorial growth of the possible combinations. Each connection set was generated using the Unix rand function to generate the sender's federate number, connection weight, number of receivers, and the receivers' federate numbers.

Table 4-1. Random Connection Set Test

f	10
$t_r = t_s$	10 milliseconds[9]
t_p	10 milliseconds between all federates
t_{max}	1000 milliseconds
(n, m)	(5, 2), (5, 3), (6, 2)

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, of which there were a few.

The results of the LOC algorithm compared to point-to-point and to all possible groupings were reported in (Morse99). These results demonstrated that our initial LOC algorithm

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

makes grouping decisions designed to minimize t_{ds} and the simulation results show that it's successful. The next version must also seek to minimize t_q to produce better results, i.e.

- ❖ We must find empirical measures for predicting t_q for the grouping algorithm to make accurate predictions about the impact of sending extraneous messages.

In addition, while the LOC algorithm takes t_{max} into consideration for the group as a while, it doesn't prevent individual federates from being swamped, i.e.

- ❖ The grouping algorithm should take into account all incoming connections as well as outgoing connections because an incoming connection not included in the multicast grouping can overwhelm a receiver.

Detailed analysis of three cases revealed that one or more federate was output-throttled by using point-to-point. An output-throttled connection is one for which k•w• $t_s$exceeds t_{max}. The first effect of such a connection is that the sender physically cannot send all the required updates in time without using multicast. The second side effect is that the average message delivery time at the receivers may appear lower because many messages never leave the sender. When these connections were added to multicast groups, the sending federate was able to send all messages at the desired rate, resulting in send rates as much as four times higher. Of course this resulted in slower average receive rates since the receiving federates had to receive four times as much data.

4.4 The Input-Restricted LOC (IRLOC) Algorithm

The input-restricted largest outgoing connection (IRLOC) algorithm seeks to minimize both t_{ds} and t_q to produce better average results than the 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 modifies the LOC algorithm in response to these facts. It performs the following steps:

- Calculate the positive cumulative effect of each connection, (k - 1) • w.
- Add the receivers of the connection with the largest cumulative effect; in the event of a tie, add the lowest numbered such connection.
- 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.
- Repeat step 3 with the remaining connections. Halt when all connections are assigned or all multicast groups are used.

4.5 Comparing LOC and IRLOC

The discovery of the effects of output-throttled connections lead to the realization that the

average message delivery time reported by the offline simulation does not capture all the important measures of goodness when the limits of message delivery are tested. And these are precisely the cases of most interest. As a result, the evaluation criteria were refined beyond just average message delivery time to include the effects of output throttling and overflowing the receiving queue. Output throttling and exceeding t_{max} are boolean failure conditions; any algorithm which produces either effect for a connection set is considered to have failed. If an algorithm succeeds on these two criteria, it is compared to other successful algorithms on the basis of average message delivery time. So, the four evaluation criteria are:

1. No receiving federate exceeds t_{max} for its average message delivery time (Success or Failure)
2. No sending federate is output throttled for any connection or set of connections (Success or Failure)
3. Average message delivery time
4. Number of extraneous messages

The 30 random test cases were re-run for just LOC, IRLOC, point-to-point, and broadcast. The percentage of extra messages, both positive and negative, was calculated for LOC, IRLOC, point-to-point, and broadcast based on the incoming weight at each federate using point-to-point. The point-to-point incoming weight reflects the number of messages which should be received, but not necessarily the number that are received with point-to-point owing to output throttling effects.

Any grouping which results in output-throttled connections or any receiver having an average message delivery time greater than t_{max} are considered to have failed. 26 of the point-to-point experiments and 8 of the broadcast experiments failed on one or both of these criteria. These are precisely the types of test cases to which grouping is applicable.

Among test cases that pass the first two criteria, a lower average message delivery time is preferable. In all cases, LOC produced lower average message delivery time than point to point, but often at the predictable cost of delivering extraneous messages. In 24 of the 30 test cases, LOC produced lower average message delivery time than broadcast. However, in all cases it delivered less data, as much as 50% less data. The analysis of the remaining 6 test cases reveals that the LOC algorithm could be more aggressive about adding connections to groups because the time to discard messages is much lower than the time saved by reducing sends. In all cases where LOC failed the t_{max} or output-throttling criteria, no solution exists to prevent these conditions because either some receivers had point-to-point input weights which cause them to exceed t_{max} or the sender had multiple connections with output weights that they could not all be sent, even if they were all assigned to multicast groups.

The IRLOC algorithm generates consistently good results for all cases where a good solution exists. In 23 of the 30 cases, IRLOC produced average message delivery times as low or lower than IRLOC, and usually with more accurate delivery of the correct data. In three of the seven cases in which IRLOC had higher message delivery times, it delivered exactly the correct set of data in 34.82, 34.11, and 34.64 milliseconds vs. 32.03, 32.07, and 33.24 milliseconds for LOC where the fastest possible delivery time is 30 milliseconds given $t_s = t_r = t_p = 10$ milliseconds.

There are two important points about these three cases. First, while IRLOC delivered exactly the right data for all three cases, LOC delivered 252%, 102%, and 268% **extraneous** messages for the same cases. Second, all seven cases had fairly light connection sets, i.e. nearly all of the connections could have been put in a single group without exceeding t_{max}. The

IRLOC algorithm balances the positive effect of adding a connection to a group against the **potential** negative effect of adding it. When grouping light connection sets, this approach is overly conservative because the receivers have a lot of spare time to throw away extraneous messages, i.e. the potential negative effect is higher than the actual negative effect. Under such circumstances, the more aggressive LOC grouping algorithm works better and the additional overhead of IRLOC is not warranted.

5 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 identified the critical performance impacts and incorporated them into two algorithms for performing multicast grouping. And we have shown that these algorithms can be expected to perform favorably in terms of data delivery against point-to-point delivery and broadcast.

6 FUTURE WORK

In this paper we have only presented results of using the LOC and IRLOC algorithms on static snapshots of connection sets. Clearly the real challenge is to perform grouping on dynamically changing connection sets. A distributed version of the IRLOC algorithm has been implemented and tested relative to RTI 1.3. The small change in overhead bodes well for ultimately incorporating dynamic multicast grouping into the DDM implementation in a production RTI.

7 REFERENCES

3Com Corporation. "Scaling Performance and Managing Growth with the CoreBuilder 3500 Layer 3 Switch." Available at http://www.3com.com/products/dsheets/400347a.html.

Howard Abrams. Extensible Interest Management for Scalable Persistent Distributed Virtual Environments. Ph.D. Dissertation, Naval Postgraduate School, December 1999.

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).

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.

James O. Calvin, Duncan C. Miller, Joshua Seeger, Gregory Troxel, and Daniel J. Van Hook. Application Control Techniques System Architecture. Technical Report RITN-1001-00, MIT - Lincoln Labs, February 1995.

James O. Calvin, Carol J. Chiang, and Daniel J. Van Hook. Data Subscription. In 12th Workshop on Standards for the Interoperability of Distributed Simulations, pages 807-813. March 1995.

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.

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. 1999; L. Bic; M. Dillencourt; and K. Tsai. "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. 2000. "Interest Management in Large Scale Virtual Environments." MIT Presence, March 2000.

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.

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.

Daniel J. Van Hook. RITN IM and IM History. Personal Communication, January 1996.