An agent architecture for network support of distributed simulation systems

Robert Simon, Mark Pullen and Woan Sun Chang
Department of Computer Science
George Mason University
Fairfax, VA, 22032
U.S.A.
{simon, mpullen, wchang}@cs.gmu.edu

Abstract

Continued research into distributed agent-based systems and evolving web based technologies are opening up tremendous possibilities for the deployment of large scale and highly extensible and flexible simulation systems. However, the question remains as to how well agent and web based simulation systems can use underlying network protocols. This is especially true for complex distributed simulations that have multiple participants and real-time data delivery requirements. This paper addresses the issue of interconnection patterns between web-based simulation agents. We first describe a simple agent-oriented architectural extension to an existing distributed simulation system. Next, we present a set of interconnection algorithms between the agents in the system, under the assumption that only end-host multicast is available. In other words, we assume native IP multicast will not be deployed in the near future on a large scale. We experimentally evaluate our approach under a number of different scenarios. Our results show that by using a proper connection strategy it is possible to get good performance for agent-based systems that use end-host multicast.

Index terms

Agent-based Simulation, Performance of Agent-based Simulation Systems, Overlay Multicast Support for Distributed Simulation

A. INTRODUCTION

One of the problems currently facing distributed simulation system designers is how to best take advantage of evolving web-based technologies to provide for simulation service interoperability, composability, extensibility and discovery. Since many distributed simulation systems require real-time performance, a closely related problem is how web-enhanced simulation environments can effectively use network level QoS services for reliability, bounded latency and group-based multicast. The challenge in addressing both issues is that web-based simulation technologies and network-level control polices operate at different system levels, and the policies and semantics at each level do not readily translate into efficient support at another level. For instance, software agents are often the entities that control web-based simulation programs. However, application-level agents may not make efficient use of available network service choices such as end-host multicast, whereby the multicast function is shifted from the network into the application [3]. End-host multicast is required when IP multicast is not available. The focus of this paper is the design and implementation of an agent-based architecture and set of interconnection algorithms to efficiently use end-host multicast in the Next Generation Internet (NGI).

We have previously addressed the issue of supporting a web-based distributed virtual simulation (DVS) system over the NGI [9]. Portions of this earlier work described our NGI Internet Federation Object Model (NGI FOM) and the Selectively Reliable Multicast transfer Protocol (SRMP). The NGI FOM provided a structured FOM to support a variety of multicasting styles within a distributed simulation system such as the HLA[13]. SRMP is a transport level protocol that supports a mix of reliable and best-effort multicast by taking advantage of the specific requirements of the DVS. For instance, many data streams are refreshed frequently and thus may be distributed by best-effort transport, while a small fraction of traffic consists of data change rarely and thus requires reliable transport.

Our original work was implemented by modifying an RTI written in TAO. This paper extends our approach to take advantage of several proposed web-based technologies, specifically distributed agents using Web Services [8] and the Semantic Web[5]. Our goal is to develop and evaluate an "agentified" version of our work in the NGI FOM and SRMP, and to use them to effectively support an end-host DVS application. Using this framework, we design interconnection algorithms between agents that make efficient use of the network.

We have defined three basic agent classes; one to represent individual running federations and their associated FOMs, a second class concerned with end-to-end data transport, and a third class of network-level agents. Our agents and their capabilities are described using an ontology markup language based on DAML-S, and is therefore compatible with current work in web services and other agent ontologies [8].

Previous related work in Web Services for distributed simulation includes research into techniques for composable services [2][10]. However, to date, there has been little work addressing the relationship between mobile and composable simulations and end-host multicast, which is increasingly becoming an accepted alternate to direct IP multicast.

B. BACKGROUND

The use of agent-based computing in designing flexible and extensible simulation systems has long been recognized[1]. Agents are a powerful software engineering unit that can encapsulate complex rule sets, can be composed from other agents and can communicate semantically rich messages via agent communication languages. Messages passed between agents are described by knowledge representation schemas called ontologies[5].

Agent computing is particularly well suited for deployment in the context of recent work in Web services and the Semantic Web. Web services are applications or software components that are deployed on the World Wide Web. Web services have an associated suite of enabling protocols, such as the WSDL [7](Web Services Definition Language). Web simulation services must be described along multiple dimensions, including both syntactic (how to use it) and semantic (what does the service mean and what does it offer). While methods for syntactic descriptions are generally well understood for distributed simulation, the issue of semantic description is still considered an open question. One promising approach to this problem is through ongoing efforts in Semantic Web service description.

A potential drawback in using web-enabled agent-oriented simulation is that system performance will suffer. This is particularly true in the case of group-based communication systems that use end-host multicast. End-host multicast is where group management and packet replication is performed on end host systems, rather then within the network, as in the original IP multicast model. As the end-host multicast model increases in importance, there is growing concern about its impact on system performance, especially for systems such as time sensitive DVS. For instance, since agent composition and location can cause communication bottlenecks, it is essential to design effective agent placement algorithms for end-host multicast.

C. AGENT DESCRIPTION

We use three classes of agents. The first class is *federate* agents, and is used to represent and control individual simulation programs associated with a particular NGI FOM. The function of these agents is to allow for automatic service discovery, service invocation, service composition and service execution monitoring for simulation federations. The second agent class is *fed_com* agents, short for federation communication. These agents are used to specify and control communication level reliability and group communication issues, such as multiple types of reliability for different data objects in the same federation by providing agent control over SRMP functionality. The third agent class is *serv_comp* agents, short for service composition agent. These agents interact with the network level to specify the construction of multicast overlay trees. *Serv_comp* agents determine the interconnection pattern between *fed_com* agents. They also possess the necessary functionality to perform data conversions, such as lowering the data rate to certain members of a group, due to compression techniques, data distribution management, etc.

These three classes of agents communicate by exchanging web-pages. Each web page contains ontological information based upon DAML-S. In the DAML-S model, each ontological framework consisting of three subonotologies: a *service profile*, a *process model* and a *grounding.* The agent service profile describes what the service can do, for purposes of service advertisement, service discovery and service *matchmaking*. Matchmaking is the act of matching services with other services. The process model describes how simulation services work in terms of control and data flow in terms of inputs and outputs, along with the service's preconditions and side effects. The grounding specifies the details of how an agent can access a service. In our case we use WSDL as the format to specify *fed_com* communication modes and associated transport-level port numbers.

D. ARCHITECTURE

Our approach is to define two communication paths through the system. The first is direct agent-to-agent communication, and is used for setup and control purposes. This occurs as agents exchange web pages. The second type is used directly for data transfer, bypassing direct agent control. This occurs when direct network paths are setup to go between *fed_com* agents. The reason for two paths is that agent communication methods, including ours, operate by exchanging XML messages. This technique is too slow for time sensitive data transport. The *serv_comp* agents will also participate directly in data transfer, using direct network connections.

A simulation is initiated when a *federate* agent finds other *federate* agents via the use of a system matchmaker. Under the direction of the initiating agent, the agents perform an all-to-all exchange of web page descriptions, which include their FOMs and requirements for runtime simulation parameters, based on the RTI model for simulation specification. Using this shared information, the runtime simulation requirements are encoded in the form of a XML description. Each *federate* agent then discovers a local *serv_comp* agent to determine network connectivity and data management requirements. This includes a specification of the *federates* expected data rate and the selection of the desired multicast service, either IP Multicast or end-host.

Serv_comp agents manage multicasting and the need to support both heterogeneous data rates and data reliability modes. Data rates are determined by network connectivity and application needs. For instance, in the case of DVS systems requiring streaming data or media streams different rates can be supported, due to different compression technologies or network connectivity. Interest Management techniques in the HLA also can result in different required data rates. Data reliability is a function of the FOM.

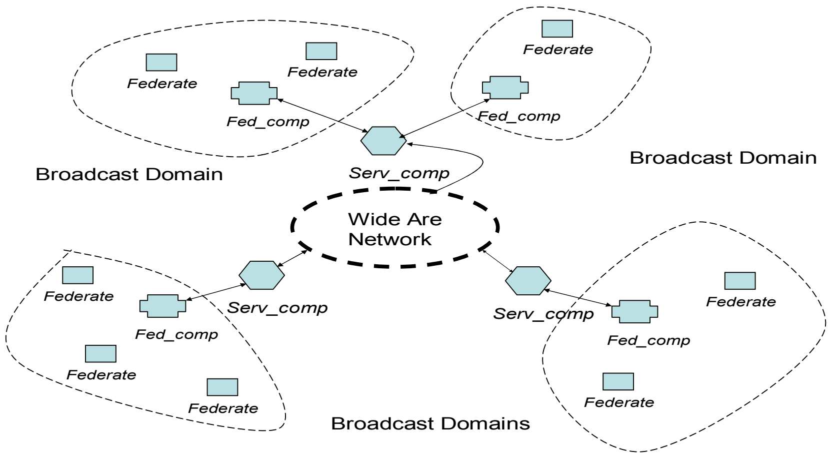

Figure 1: Simulation agent architecture

From the architectural viewpoint, *serv_comp* agents manage local domains that represent connection points between *federate* agents running on end-hosts and the Wide Area Network (WAN). A local domain may be one or more Local Area Networks (LAN), a single high-end computer connected directly to a router, etc. We characterize the local domain as a *broadcast* domain. In effect, each broadcast domain is the end-point of a logical multicast distribution tree.

Based on the simulation specification and the location of *federates* inside each broadcast domain, each *serv_comp* discovers and assigns *fed_comp* agents to provide transport level mixed data delivery mode support. Thus, the number and location of each *fed_comp* agent represents the endpoints in the logical data distribution tree. An overview of this architecture is shown in Figure 1. If native IP Multicast were possible, *fed_comp* agents could be placed directly in a spanning tree. However, under the end-host scenario each *fed_comp* agent needs to perform data forwarding and packet replication. Therefore, the interconnection pattern between *fed_comp* agents is of central importance to system performance.

The purpose of defining and analyzing *fed_comp* interconnection algorithms is to reduce communication latency and system load, thus improving end-to-end performance. Notice that all end-host overlay techniques cause extra packet transmissions over the network. Our goal is to minimize this effect and reduce actual bandwidth utilization. Our approach to multicasting is for single-source tree construction. That is, each sender is the root of a tree that consists of all the receivers. Further, we dynamically place *serv_comp* agents within the tree at specific places in order to reduce data traffic transmission, as explained below.

The algorithm has three phases. In phase one, we create a mesh of *fed_comp* agents ordered by data rate. In phase two, we create a series of single source overlay multicast trees. For phase two, we add application-level admission control, in an attempt to limit congestion over any particular portion of the WAN. We also use a load-balancing path selection heuristic. Our algorithm is flexible, and uses several different approaches to tree creation. In phase three, we place extra *serv_comp* agents in the system to support the data rate of receiving *fed_comp* agents, as required. This allows us to further reduce bandwidth utilization. The details follow.

Phase 1: Create a mesh of *fed_comp*s per source's different data rate. The mesh is a complete virtual graph whose vertices are the *fed_comps*. First, find the shortest path between each *fed_comp* using link delay as the link cost. Then add a uni-directional virtual link between the two group members, (i.e. *fed_comp*) to the mesh, *only* when there is a path that starts from a sender to a receiver *fed_comp*, the data rate requested by the sender is greater than or equal to the data rate requested by the receiver, and the physical links of the shortest path have the available bandwidth greater than or equal to the data rate requested by the receiver.

Algorithm for Mesh Creation

```
Let $\hat{M}$ be a set of fed_comp agents.
For all $\hat{m}_i \in \hat{M}$
{ // Required in case of newly assigned fed_comp
        if ( $\hat{m}_i$.data_rate > (max{federate.data_rate})
                $\hat{m}_i$.data_rate = (max{federate. source.data_rate};
}
For all $\hat{m}_i \in \hat{M}$
{
  Find the shortest path from $\hat{m}_i \in \hat{M}$ to $\hat{m}_j \in \hat{M}$ where $i \neq j$ using a shortest path
  algorithm and the delay as the link cost.
}
For all $\hat{m}_i \in \hat{M}$
```

```
{
        if (($\hat{m}_i \neq \hat{m}_j$) and ($\hat{m}_i$'s data rate $\geq \hat{m}_j$'s data rate) and
        (the bottleneck bandwidth of the path from $\hat{m}_i$ to $\hat{m}_j$ $\geq \hat{m}_j$'s data rate))
        {
            Add a virtual link from $\hat{m}_i$ to $\hat{m}_j$ to the mesh.
        }
}
```

We repeat this procedure to create a mesh per each source's different data rate. If the sources' data rates are the same, a mesh can be used to create Overlay Multicast Trees rooted on the sources with the same data rate.

Phase 2: Create a source-based Overlay Multicast Tree over the mesh that the source's data rate matches. We do application-level admission control, insuring that no particular link or path is overloaded. Our algorithm accommodates either a shortest path or a spanning tree routing tree construction. We use a load balancing heuristic based upon avoiding paths for which there is a large imbalance of utilization along specific links. Phase 2 requires this extra complexity because we are trying to minimize multiple routing metrics, including delay, bottleneck bandwidth and overall tree cost.

During construction of source-based multicast tree, at each step, a *largest data rate edge* connecting an off-tree node to the partial tree is added to the tree. When a node decides its parent, it chooses the node connected with the mesh link that has the least value of $(1 - ILBR(P_{\hat{m}_i,\hat{m}_j}))$, where $ILBR(P_{\hat{m}_i,\hat{m}_j})$ (*Independent Least-balanced Routing* [11]) is defined as

$$ILBR(P_{\hat{m}_i,\hat{m}_j}) = \frac{\prod_{\forall z \in P_{\hat{m}_i,\hat{m}_j}} ec_z}{\sum_{\forall z \in P_{\hat{m}_i,\hat{m}_j}} ec_z}$$

Excess capacity ec_z at each link in the system is link z is defined as the bandwidth requirements of the simulation connections routed through the link as a percentage of total link capacity. This information is obtaining using standard network monitoring and tracing tools.

Only when the available bandwidth of the bottleneck link of the path between two group members supports the receiver's data rate, the path is added to the overlay multicast tree. If a path is added to the overlay multicast tree, the available bandwidth of each link on the path is reduced by the receiver's data rate.

Each parent *fed_comp* keeps only the *fed_comp* with the highest data rate request among the children candidates as its children. This condition is enforced by placing only one *fed_comp* in each domain. The algorithm shown below implements finding a minimal cost spanning tree.

Algorithm for routing overlay creation

Let $\hat{M}$ be a group of *fed_comps* and $\hat{m}_i \in \hat{M}$ be a multicast group member.

Let L be a list of fringe *fed_comps*. The status of $\hat{m}_i \in \hat{M}$ can be UNSEEN, FRINGE, and IN-TREE. UNSEEN means the node is never visited, FRINGE means the node was visited, and IN-TREE means the node is added to the multicast tree.

```
Set the status of all $\hat{m}_i \in \hat{M}$ to UNSEEN;
Set source_node's status to IN_TREE;
current_node = source_node;

For all $\hat{m}_i \in \hat{M}$
{
    For all $w_i$ node adjacent to the current_node
    {
        next_node = $w_i$;
        If (next_node.status is UNSEEN)
        {
            //admission_control returns true if the available bandwidth of the
            //bottleneck link of the path from current_node to next_node is
            //greater than or equal to the next_node's data rate, otherwise
            //returns false.
            If (admission_control(current_node, next_node) == true)
            {
                next_node.status = FRINGE;
                next_node.cost = $1 - ILBR(P_{current\_node,next\_node})$;
                next_node.parent=current_node;
                Add next_node into the list of fringe node;
            }
        }
        If (next_node.status is FRINGE)
        {
          new_cost =$1 - ILBR(P_{current\_node,next\_node})$ ;
          If (next_node.cost > new_cost)
          {
```

```
                    release_path_bandwith(next_node.parent, next_node);
                    If (admission_control(current_node, next_node) == true)
                    {
                        next_node.cost = new_cost;
                        next_node.parent=current_node;
                    }
                    else
                        //reserve the bandwidth again for the path between next_node
                        //and its parent.
                        admission_control(next_node.parent, next_node);
                }
                If (the parent of next_node already has a child whose data_rate is
                greater than that of next_node)
                {
                    release_path_bandwith(next_node.parent, next_node);
                    If (admission_control(current_node, next_node) == true)
                    {
                        next_node.delay = new_delay;
                        next_node.parent=current_node;
                    }
                    else
                        admission_control(next_node.parent, next_node);
                }If
        }//For

        Add the off-tree node with the largest data rate among the fringe nodes to the
        multicast tree (IN-TREE);
        current_node = the selected node;

}//end For
```

<u>Shortest Path– based Heuristic Variant</u>

This is similar to the above Minimum Spanning Tree-based algorithm. The difference is that it uses the cost of the path from the source group member to another member to choose a parent of a node instead of the cost just between the parent_node and child node to determine a parent_node of a node. We omit the details because of space considerations.

Phase 3: Place *serv_comps* to adapt the data rate to users' requests at the nodes on the overlay multicast tree that have the children whose requested data rates are smaller than the node's. Finally, at the end of Phase 3, direct network connections are instantiated between the *fed_coms* in the system, along with connections between any added *serv_comps* and their associated *fed_coms.*

The result of the three phases of this algorithm is direct transport level network connections between *fed_comps*, the agents that control SRMP functionality. Further, if as a result of Phase 3 additional *serv_comp*s were added for data reduction, there are transport level connections setup between these *serv_comps* and other *fed_comps* in the system. Recall that for data transfer all connections use regular transport level protocols, not agent communication protocols.

E. Evaluation

We have conducted an experimental evaluation of our approach for constructing *fed_comp* interconnection patterns. We compare the performance of the Shortest Path-based heuristic and the Minimum Spanning Tree-based heuristic with the performance of IP Multicast, using DVMRP, along with pure unicast. We determine the IP Multicast tree based on the unicast paths from the source to each receiver. The evaluation metrics for the performance of each algorithm includes the number of *serv_comp* agents needed to be placed for a simulation, the average link stress, the worst link stress, the resource usage, the total bandwidth usage, and the relative delay penalty (RDP) of each path.

In reference [5], the stress of a physical link is defined as the number of identical copies of a packet carried by a physical link. For our evaluation, the physical link stress is ratio of the number of packets that a link processed to the number of packets sent by all members. Resource usage is the sum of the product of the link stress and the link delay over all links used. We assume that links with higher delay values tend to be associated with higher cost. The total bandwidth usage is the sum of all bandwidth reserved along the paths from all sources to all receivers. The resource usage and total bandwidth are metrics to measure the network resources consumed in the process of data delivery to all receivers. RDP is the ratio of the latency experienced when sending data using the overlay to the latency experienced when sending data directly using the underlying network.

In our simulation, all *fed_comp* agents send and receive packets. We created a separate source-based multicast tree for each member to send out packets to all other multicast group members in IP Multicast, the Shortest Path-based overlay heuristic and the Minimum Spanning Tree-based heuristic. The admission control is performed when a path is added to the multicast trees in IP Multicast, the shortest-tree-based heuristic and the minimum-tree-based heuristic, and also for a path between the source and receiver for unicast. A source-based tree is accepted only when there is bandwidth enough to meet the data late requested by each member for all the paths from a source to all other multicast group members. Otherwise, the tree is rejected.

We used CSIM package to write the simulation programs, and Georgia Tech Internetwork Topology Models (GT-ITM) [12] to generate transit-stub graphs with 100 nodes. The multicast group size varies from 5, 10, 15, 20, 25, and 30. Notice that 30 group members implies that 30% of the routers are part of the group, which is quite a high percentage compared to a typical WAN scenario. We included this worst case scenario as a stress test for our approach. The members are uniformly assigned to each node on the graph. We ran the simulation program for each heuristic 30 times per group size. Each time, we used different graph with 100 nodes. We used the number for each link in the graph generated by GT-ITM as the link delay. The delay of the local link is randomly distributed between 1 and 15. We used 1 Gbps for the bandwidth of the backbone links and 20 Mbps for the local link between the router and the end host. The data rates requested by the *fed_comp* members are randomly distributed among 150 Kbps, 60 Kbps, and 20 Kbps.

The first set of results in Figure 2 shows the effect of data rate reduction, by comparing the number of *serv_comp* agents placed in the two overlay techniques to what would happen in IP Multicast. In the latter case, data reducing gateways are required at the end-points of the IP multicast tree. Notice that this comparison is not required in unicast.

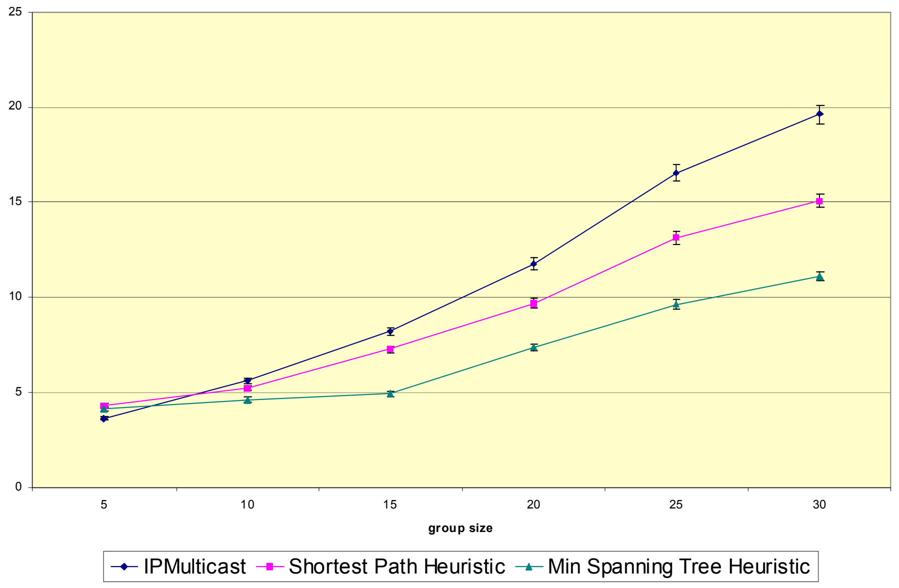

Figure 2 The number of *serv_comp* Agents due to data reduction

The averages are between 9.98 and 10.42 for the Minimum Spanning Tree-based heuristic, between 15.86 and 16.04 for the Shortest Path-based heuristic, and between 19.23 and 20.04 for IP Multicast at 95% confidence level. Notice that the overlay approach actually *reduces* the number of data-reducing agents required over IP multicast, which must be modified to make this type of intelligent data reduction in the data path. This is a strong indication that agent mobility is important.

Figure 3 shows average link stress. The averages are between 0.397 and 0.401 for IP Multicast, between 0.80 and 0.81 for the Minimum Spanning Tree-based heuristic, between 0.99

and 1.0 for the Shortest Path-based heuristic, and between 1.05 and 1.07 for unicast at 95% confidence level.

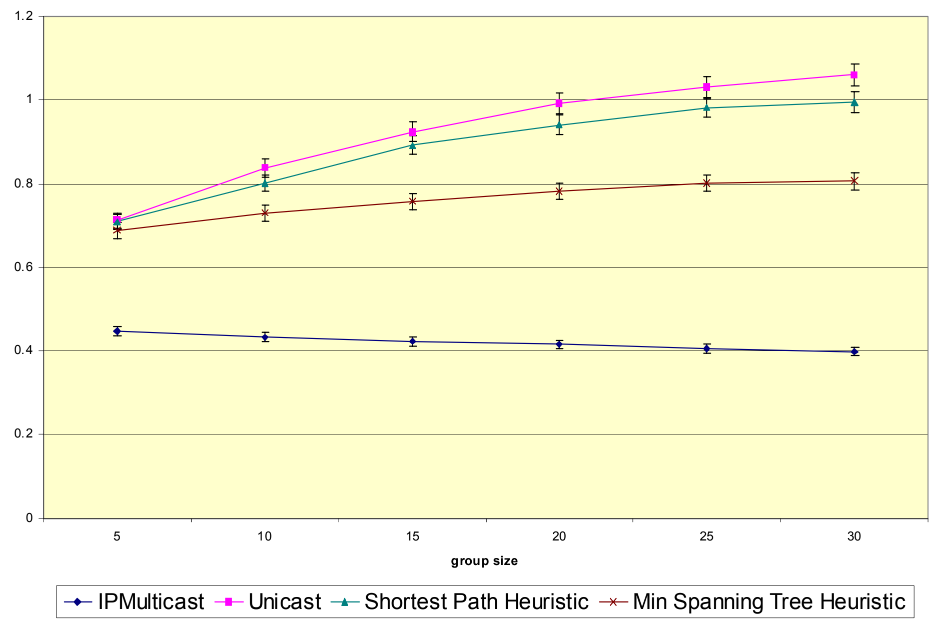

Figure 3 Average Link Stress

Figure 4 shows the maximum, or worst case link stress for the scenarios. This gives us an indication as to the "imbalance" of the various approaches. The minimum-spanning-based heuristic has the larger worst link stress than IP Multicast, but has smaller worst link stress than unicast. However, the Shortest Path-based heuristic has the larger worst link stress than IP Multicast and unicast. The figure 3 shows the result.

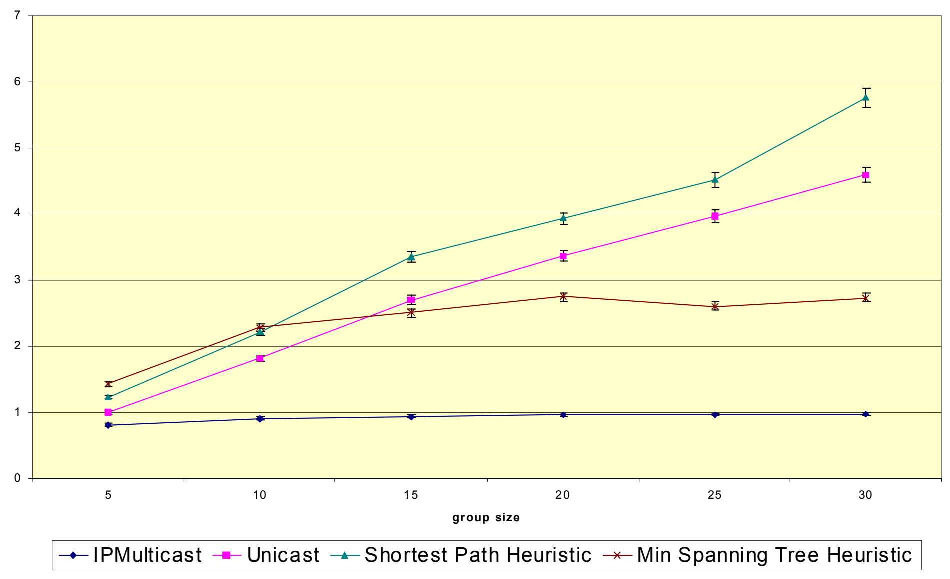

Figure 4 The Worst Link Stress

Figure 5 shows the total bandwidth usage. The Shortest Path-based heuristic and the Minimum Spanning Tree-based heuristic naturally both have larger total bandwidth usage than IP Multicast, but the smaller than unicast. The Minimum Spanning Tree-based heuristic has the smaller total bandwidth usage than the Shortest Path-based heuristic. The results show that our approach scales in linear fashion to the Multicast approach

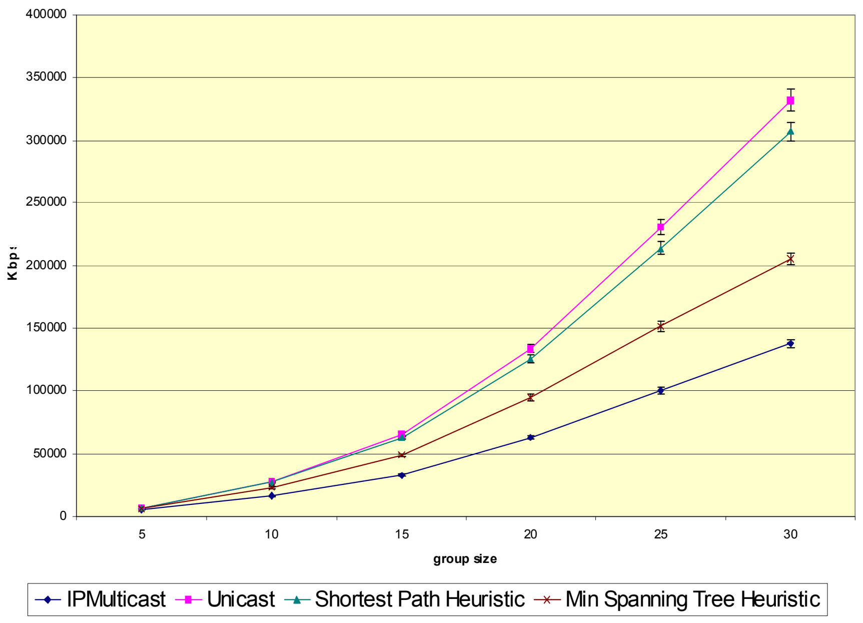

Figure 5 Total Bandwidth Usage

Figure 6 shows the RDP. The Shortest Path-based heuristic has a smaller RDP than the Minimum Spanning Tree-based heuristic. We used 1 for the RDP of unicast and IP Multicast. The RDP penalty for the Shortest Path heuristic is quite competitive as compared to IP multicast and unicast, indicating the viability of our approach for time sensitive communication.

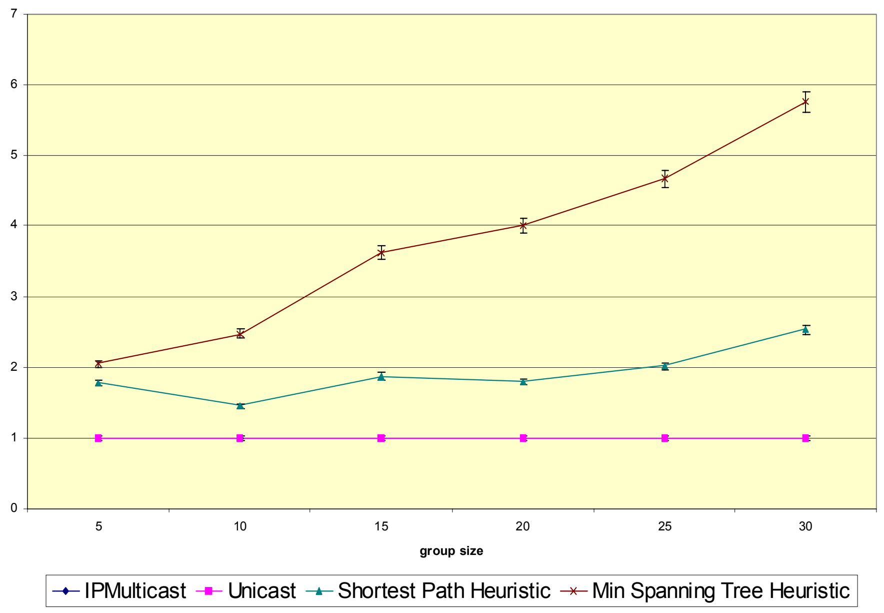

Figure 6 Relative Delay Penalty

Finally, Figure 7 shows Resource Usage. The Minimum Spanning Tree-based heuristic and the Shortest Path-based heuristic both have larger resource usage than IP Multicast, but smaller than unicast. Once again, we see that our approach scales in linear fashion.

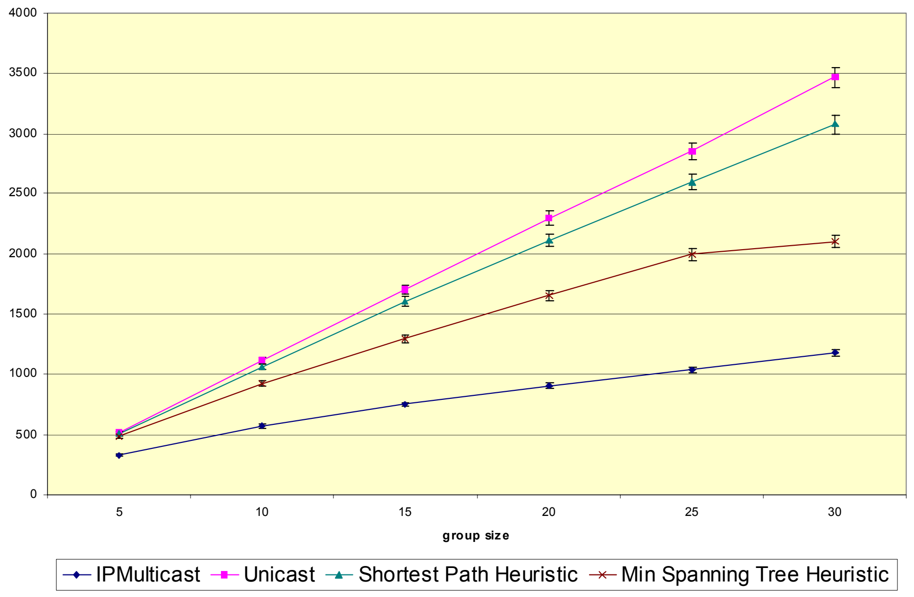

Figure 7 Resource Usage

In summary, we see that for moderate group sizes, the overlay techniques are within a factor of 2 as compared to IP multicast for expected delay, stress, resource usage and bandwidth usage. When group members approach 30% of the available network router size, overlay techniques are less efficient. However, in virtually all cases the overlay techniques outperform IP multicast for number of required data reduction agents. We observe that for networks where IP multicast is not available it is possible to still obtain good network performance, expect in highly saturated networks. Further, since the end-host techniques can make intelligent use of packet rebroadcast, the number of data reducing agents required is reduced in the overlay methods, as compared to IP multicast.

E. CONCLUSIONS

This paper described an agent-based architecture and interconnection strategy for a distributed virtual reality simulation environment. We first discussed a web-based ontology for three classes of agents required to make a previously deployed system compatible with current work in Web Services and the Semantic Web. We then described how these agents initialize and configure themselves into an interconnection pattern. We introduced a new algorithm for agent interconnection, involving three separate phases. The outcome of the third phase involved setting up direct network connections, and may also result in additional agents being placed for purposes of data reduction.

We experimentally evaluated our approach, using two different heuristics, against IP Multicast and pure unicast. Expect in the circumstances of extremely dense group membership, our approach perform in a competitive fashion to IP multicast. Further, the ability to place intelligent agents for packet forwarding on end-hosts reduces the computational burden within the system.

F. REFERENCES

[1] *Agent Based Simulation 2002: 3rd Agent Based Simulation Workshop,* Edited by Christopher Urban, Society for Computer Simulation, April 2002.

[2] S. Chandrasekaran, et. al., G, "Web Service Technologies and their Synergy with Simulation," Proceedings of the 2002 Winter Simulation Conference (WSC'02), San Diego, California (December 2002) pp. 606-615.

[3] Y. Chu, S. Rao, and H. Zhang. "A Case for End System Multicast," In *Proceedings of ACM Sigmetrics*, June 2000.

[4] DAML: The DARPA Agent markup Language Homepage, available at http://www.daml.org.

[5] J. Hendler, "Agents and the Semantic Web" IEEE Intelligent Systems, March 2001, pp. 30-37.

[6] S. Jain, et. al., "A Comparison of Large-Scale Overlay Management Techniques," Technical Report UW-CSE 02-02-02, Computer Science and Engineering, University of Washington, 2002.

[7] U. Ogbuji, "Using WSDL in SOAP applications," available at http://www-106.ibm.com/developerworks/webservices/library/ws-soap/index.html?dwzone=ws

[8] S. A. McIlraith, D. L. Martin, "Bringing Semantics to Web Services," IEEE Intelligent Systems, January/February 2003 (Vol. 18, No. 1) pp. 90-93.

[9] Pullen, J.M, Simon, R., Khunboa, C., Parupalli, M. and Brutzman, D., A Next-Generation Internet Federation Object Model for the HLA," Sixth IEEE International Workshop on Distributed Simulation and Real Time Applications (DSRT) October 2002, Fort Worth, Texas.

[10] A.P. Sheth and John A. Miller, "Web Services: Technical Evolution yet Practical Revolution?" *IEEE Intelligent Systems (IEEEIS),* Vol. 18, No. 1 (January-February 2003) pp. 78-80. IEEE Computer Society Press.

[11] R. Simon and A. Sood, "Load-balanced Routing for Collaborative Multimedia Communication", *6th International Symposium for High Performance Distributed Computing (HPDC '97)*, August 1997, pp. 81 - 90.

[12] M. Thomas and E. W. Zegura. "Generation and Analysis of Random Graphs to Model Internetworks," Technical Report GIT-CC-94-46, College of Computing, Georgia Tech, 1994.

[13] F. K., Weathery, R., and Dahmann, J., *Creating computer simulation systems*, Prentice Hall, New Jersey, 2000.