DYNAMIC INTEREST FILTERING FOR OPTIMAL STATE UPDATE MESSAGING

Dr. Stephen Zabele	Matt Dorsch	Mark Keaton	Dr. Rosemary Kennett
TASC, Inc.	TASC, Inc.	TASC, Inc.	TASC, Inc.
Reading, MA	Reading, MA	Reading, MA	Reading, MA

ABSTRACT

Achieving truly large-scale, real-time distributed simulation has remained a highly elusive goal primarily due to limitations in current generation network technology. While the progression from early, broadcast-based DIS approaches to the current multicast-based publish/subscribe approaches (Van Hook and Calvin, 1994) embodied in the HLA (Defense Modeling and Simulation Office, 1998) has significantly extended the number of simulated hosts that can be supported, limitations in the networking infrastructure still force compromises that historically have constrained simulation exercises to only a few tens of hosts. This constraint is overwhelmingly the result of the networking infrastructure's inability to adequately control the delivery of unneeded and unnecessary traffic to the simulation hosts – a problem so severe that the overhead of reading and discarding unneeded state update messages can critically impair a host's ability to perform its primary simulation tasks. This unintended distributed denial of service problem also results in an enormous waste of network and human resources. In particular, the approach taken for United Endeavor of over-provisioning WAN links with multiple T3 circuits coupled with multiple man-weeks of trial-and-error to find a "workable" multicast grouping is extremely expensive and time-consuming.

Under the Defense Advanced Research Projects Agency (DARPA)-sponsored Specialized Active Networking technologies for Distributed Simulation (SANDS) project, we are developing Active Networks-based capabilities to significantly improve multicast-based distributed simulation performance[1]. In particular, we have created a capability for dynamically and automatically configuring and reconfiguring application-specific content management filters directly within intermediate network routers where they are most effective. With this approach, we have been able to achieve the ultimate goal of being able to eliminate *all* irrelevant network traffic at the earliest opportunity within the network, providing *optimal* use of both network and host resources. Moreover, through the use of filter processing acceleration techniques, such as tree-based filtering mechanisms, information theoretic-based filter complexity reduction techniques, and cooperative, distributed content filtering strategies, we have been able to craft an extremely low-latency (<100 usec) interest filtering capability that provably scales to support millions of simulated entities. This capability is intentionally designed to mesh seamlessly with the High Level Architecture (HLA) Declaration Management (DM) and Data Distribution Management (DDM) services, and a working version has been demonstrated using ModSAF and the Georgia Tech HLA-compatible Run Time Infrastructure (RTI).

This paper reviews our active-networks-based interest filtering architecture, the interrelationships with HLA DM and DDM services, and provides technical details and performance measurements of the various components that clearly demonstrate the performance and scalability claims for our approach.

ABOUT THE AUTHORS

DR. STEPHEN ZABELE is Manager of Advanced Networking and Information Systems at TASC Inc., where he is Principal Investigator for SANDS and other projects in the advanced networking area.

MATT DORSCH is a Member of the Technical Staff at TASC, Inc, where he is responsible for extending and enhancing simulation and HLA/RTI components to leverage SANDS technologies.

MARK KEATON is a Senior Member of the Technical Staff at TASC, Inc, where he is responsible for developing the SANDS kernel filtering and signaling components, as well as managing overall integration.

DR. ROSEMARY KENNETT is a Member of the Technical Staff at TASC, Inc, where she is responsible for the SANDS high-performance filter organization and visualization components.

[1] Defense Advanced Research Projects Agency (DARPA) contract N66001-9117-411V.

Dynamic Interest Filtering for Optimal State Update Messaging

Dr. Stephen Zabele
TASC, Inc.
Reading, MA

Matt Dorsch
TASC, Inc.
Reading, MA

Mark Keaton
TASC, Inc.
Reading, MA

Dr. Rosemary Kennett
TASC, Inc.
Reading, MA

INTRODUCTION

The *content-aware routing* mechanism described in this paper offers an elegant solution to a heretofore quite difficult network resource management problem: efficient delivery of information among a large community of users/collaborators, where each participant can have individual, highly specialized interests. Our work combines a publish-subscribe paradigm with a distributed, multicast-based, data delivery infrastructure that leverages active (or programmable) network technologies to perform content management functions directly within individual network routers. The successful combination of these technologies has enabled a highly scalable solution that has demonstrated the ability to provide optimal delivery efficiency.

Current-generation information dissemination systems commonly employ multicast as a transport mechanism due to multicast's sender-side scalability and efficiency properties. In the absence of loss, each message need only be sent once, and is only replicated by network nodes as paths to members of the multicast group diverge. By itself multicast can only provide a partial solution, however, as few (if any) members of the group will in general want or need to receive all messages sent to the group. In particular, when the number of receive hosts is large, each with highly individualized information needs, the amount of unneeded messages delivered to any given member of the group can quickly and easily outweigh the amount of needed messages, such that both network and receive host resources are wasted.

Our approach improves multicast delivery efficiency by pruning messages from those portions of the multicast tree where they are not needed, allowing messages to be forwarded only along those links necessary to reach hosts where they are needed. Pruning is accomplished using *interest filters* that can extract and use message content in making forwarding decisions at various network nodes along the multicast distribution tree. With the ability to place interest filters at precise locations within the network, pruning occurs at the earliest point possible, and delivery efficiency is optimized from both an end-user and a network perspective. Moreover, through the incorporation of filter acceleration techniques, we have developed a solution that is exceptionally lightweight with respect to network node processing requirements, and yet is intrinsically compatible with current generation router technology.

Although the original motivation for this work was developing a solution to one of the major scalability problems in real-time distributed simulation, we believe our content-aware routing solution has broader applicability to a wide range of applications, including data dissemination to large user communities and multiplayer games. Furthermore, by simply inverting the information request (inclusion) service into an information *exclusion* or *limitation* service, our content-aware routing mechanism can be cast as a dynamic, distributed firewall that can be used, for example, as a means for countering distributed denial of service (DDoS) attacks against servers.

Background

Previous efforts focused on solving the information dissemination problem have met with great difficulty, primarily due to the fact that traditional networks do not provide the flexibility or the degree of control necessary to make delivery efficient. Specifically, network limitations have forced content management functions to be performed by end-hosts on the edges of the network, mandating *approximate* solutions at the cost of substantially degraded performance. In contrast, our approach uses active network technologies (Tennenhouse, Smith, Sincoskie, Wetherall, and Minden, 1997; Kasera, et al 2000) to locate content management functions within the network where they are most effective, providing a high-performance, highly-scalable, and *exact* solution to the delivery problem.

To illustrate the core issue, consider the distributed simulation problem where state updates from individual entities are needed only by a limited number of subscribers – for example, only those entities in (simulated) close geographic proximity. Early on, network delivery of distributed simulation state updates was accomplished using

broadcast (one-to-all) protocols. Since broadcast messages are (in the absence of loss) received by all receivers, broadcast offers send-side scalability since messages then need only be transmitted once rather than repeatedly for each receiver. Broadcast-based systems suffer, however, because broadcast is indiscriminate in terms of which hosts receive which pieces of information. As the number of senders grows, the overhead of processing large numbers of irrelevant messages quickly becomes so large that receivers end up spending nearly all available processing time reading and discarding irrelevant messages, leaving no time for the actual simulation processing itself.

To improve delivery efficiency, publish/subscribe mechanisms were subsequently introduced. Here, each data producer (sender) advertises the types and ranges of data it can provide, and data consumers (receivers) that have need of the producer's data respond by sending subscriptions to the producer. The goal was to develop a delivery system where only data needed by a given receiver are delivered to that receiver, eliminating the receiver side message implosion and improving performance.

Initial attempts at implementing publish/subscribe mechanisms involved the use of unicast protocols. Although somewhat more efficient than broadcast, unicast protocols also suffer scalability problems as each message must be transmitted multiple times, once for each end host needing a particular piece of information. Additionally, use of unicast imposes the added requirement that senders keep track of which receivers need which pieces of information. This presents a significant problem in itself, particularly since information needs can be highly dynamic due to movement of entities within the simulated environment.

More recently, networking innovations such as restricted broadcast protocols (i.e., multicast) have been introduced. Like broadcast, multicast has the attractive property that messages can be sent once but delivered to multiple receivers. Unlike broadcast, however, only receivers subscribing to a particular multicast group receive messages for that group, providing considerably improved levels of discrimination.

In principle, multicast could solve a significant portion of the delivery problem, and its failure within the distributed simulation community has primarily been for pragmatic reasons. Specifically, straightforward analysis shows that the number of multicast groups needed to properly solve the delivery problem can easily reach into the millions or higher; however, the number of multicast groups that can be realistically handled by end hosts is typically less than forty. Hence, application designers have been left with no truly scalable alternatives for large-scale distributed simulation. Moreover, the problems of organizing informational needs, mapping overlapping needs onto individual multicast groups, communicating which groups carry which pieces of information, and having receivers dynamically join/drop membership in potentially large numbers of groups further complicates the use of multicast as a solution.

In order to circumvent the multicast group limitation problem, recent work has focused on channel aggregation, where large numbers of disparate informational needs are mapped onto the small number of available multicast channels. The direct penalty for channel aggregation is decreased delivery efficiency – all information sent to the group is received by all members of the group, even when only a single member of the group requires any given piece of information. The indirect penalty of channel aggregation is the aggregation processing itself. Since calculating "optimal" aggregation solutions has been shown (by members of our team) to have NP-hard complexity, aggregation must be accomplished using suboptimal, approximate methods which generally implies further decreases in delivery efficiency.

Although clearly more efficient than broadcast, in essence multicast channel aggregation approaches decompose the broadcast approach into a small number of somewhat smaller problems. From this perspective, channel aggregation in effect provides a roughly linear improvement in delivery efficiency to a delivery problem that potentially scales exponentially with the number of users in nature (as implied by the NP-complete complexity of the optimal aggregation solution). As such, the inefficiencies and overhead resulting from channel aggregation can be substantial, and is one of the primary reasons that the goal of large scale distributed simulation has remained elusive.

Application-specific Considerations

Several characteristics of the distributed simulation problem have influenced the particular approach described in this paper. Specifically:

Many-to-many Delivery – Within a distributed simulation environment, each end-host will

typically act as both a sender and a receiver on the same multicast group. This requires that the signaling approach be sufficiently sophisticated to accommodate the complexities of what is intrinsically more of a mesh topology.

Best-effort Service Model – For real-time simulations, particularly human-in-the-loop interactive simulations, elimination of simulation lag is of utmost importance. Hence real-time distributed simulation applications typically forgo the use of reliable transport (e.g., TCP), where rate control and/or error recovery/ordered delivery can potentially add large amounts of delay, in lieu of a best-effort delivery model (e.g., UDP) for the majority of the message traffic. Here, use of a best effort service model allows the delivery of only the needed fraction of a transmitted stream to any given receiver without disrupting the underlying transport mechanism.

Scalability and Performance – One of the stated goals of large-scale distributed simulation is supporting large numbers (>1,000) of simultaneous hosts. Hence the signaling and filtering approaches must be sufficiently lightweight to enable network nodes to accomplish filtering objectives without significantly affecting throughput or delivery latency. This constraint, in particular, has motivated the development of acceleration mechanisms that potentially allow network nodes to support millions of simultaneous filters without significantly degrading throughput.

Correctness Property – Due to the evolution of distributed simulation applications, an implicit correctness property for the delivery infrastructure has emerged in that all data needed by an end host must be *deliverable* to the end host, whereas delivery of any unneeded data should be minimized. *Deliverable* is used rather than *delivered* to indicate that the delivery infrastructure must enable and allow needed data to be delivered, rather than requiring guaranteed delivery – which would mandate reliable transport. Here, the correctness criteria allows complex filters or filtering requests to be approximated (if necessary) with simpler expressions that are a superset (i.e., a covering) of the expressed needs in order to accommodate any limitations in signaling or filtering performance within the routers.

Dynamic Content Management – Within distributed simulation environments, simulated entities are continuously moving or otherwise changing state, and as a result the characteristics of their data needs are also continuously changing. Here, the requirement becomes that the signaling and filtering mechanisms must allow each and every host to change its subscription dynamically (i.e., on-the-fly), without needing to suspend or otherwise impede simulation processing as updated subscriptions propagate through the network.

FILTERING

We now describe the characteristics and components of our core filtering approach. Our core filtering approach embodies the following key aspects:

- *Embedded Content Descriptors* – Methods for organizing and encoding information needed for content management functions (e.g., location, frequency, etc.) within messages are defined and agreed to by all participants. Then, for each message sent by a publisher, the associated content information is embedded at the predefined locations (e.g., as a message header). *Note that a perfectly acceptable mechanism here is to define the content descriptors as being comprised of information already embedded within the messages.* In this situation, embedded content descriptors are really virtual entities that impose no additional processing or message overhead.

- *Subscription via Interest Filters* – Subscribers submit, modify, and/or retract interest filters to active nodes within the network. Interest filters are specified as a set of operations performed on content descriptors to determine which update messages within a specific multicast group should be delivered to the subscriber. For our purposes, operations are restricted to boolean combinations of interval tests applied to each content variable independently. This "product space" restriction enables content filters to be expressed in a form compatible with packet filtering mechanisms typically found in current generation routers.

- *Content Filter Distribution* – Active routers propagate interest filter specifications from subscribers (receivers) to publishers (senders) in the reverse direction along the multicast path from sender to receiver. Active routers merge interest filters from all downstream entities (subscribers or other active routers) for the group and send the collective set of filter specifications upstream. A merged filter set may be either an exact representation of the collective set of

subscriber interest filters, or it may be a superset of these filters with a simpler representation if needed to accommodate signaling or filtering constraints.

- *Active Content Filtering* – In addition to standard multicast routing functions, routers that are also active routers will perform any relevant interest filtering to determine whether a received packet should be replicated and forwarded out a given outbound interface. Hence message forwarding for a given interface requires both an admissible route through that interface for the multicast group and that the message contain content descriptor values that are admissible by the associated set of interest filters for that interface.

Note that the collection of bits constituting the content descriptors in each message can well be thought of as a type of extended multicast address information since the bits are used to decide which of a given set of end hosts should be sent the message. Hence, our approach can be viewed as providing a virtual extension to the usable multicast address space, with interest filters providing a convenient and efficient means for subscribing to multiple multicast groups simultaneously.

A Simple Interest Filtering Example

The notional operation of a simple interest filter specification and distribution capability begins with subscribers (receivers) transmitting interest filter specifications towards the publisher (sender) (see Figure 1). (Note: in Figure 1, the content space is represented as larger squares, and needed data regions within the space are represented as shaded cells). Active routers along the path accept interest filters from both subscribers and downstream active routers, merge the filters into a single, more general interest filter, and forward merged filters upstream.

The corresponding operation of the active interest filtering function begins with the sender transmitting a message with embedded content descriptors to the multicast group (see Figure 2). (Note: it is assumed that all subscribers shown in Figure 2 are members of the group). Upon receipt of the message, each active router compares the information in the content descriptors with the interest filter associated with each outbound interface. The message is forwarded only out those interfaces with interest filters that will admit the message.

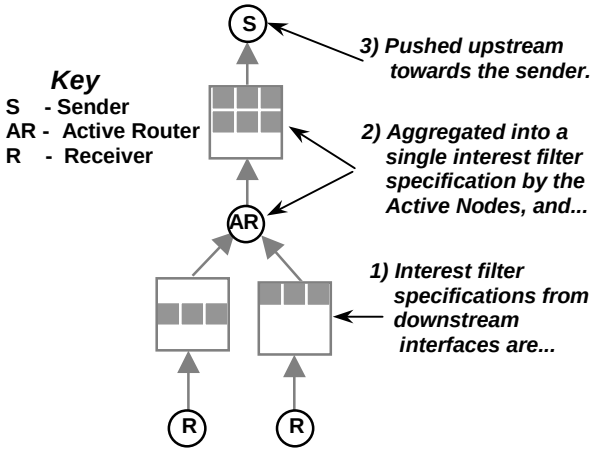

Figure 1. Interest filter specification and distribution example

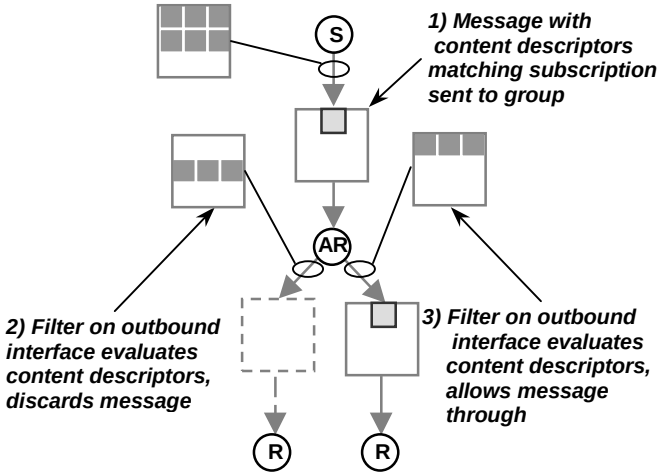

Figure 2. Interest filtering example

Note that in this example, each of the active routers has forwarded an aggregate filter specification that is *exact* in the sense that no simplifications were made by any of the active routers (e.g., as needed to meet performance goals.) Hence, even though all subscribers are in fact members of the single multicast group, only those subscribers needing the data actually receive the message. As such, no irrelevant data are received, and the system behaves as if a larger number of more precisely specified groups are in use.

Active Filtering Performance Improvements

The preceding example, while straightforward architecturally, hints at performance issues that await us as we try to scale the system to support large numbers of users. In particular, it is easy to imagine that the number of filters could quickly become so large that routers may no longer be able to complete filtering operations at line speeds and overall throughput suffers. To address these

issues, we have incorporated several novel performance acceleration techniques in the form of distributing filtering among active routers, use of tree filtering constructs within the active routers, and (as necessary) filter simplification to reduce the overall number of filters to accommodate physical storage or signaling constraints. We now present the basic ideas behind each of these techniques.

Distributed Filtering

An interesting approach for improving overall filter performance arises from the observation that a significant amount of the interest filtering performed at each node in the previous example is in fact highly redundant and therefore unnecessary. As a thought experiment, suppose that all receivers in the simple example have identical interest filters. Then, each router node downstream from the first (uppermost) node is bound to repeat verbatim each of the filtering operations performed by the first node – when in fact *no* additional filtering need be performed after the first node. The key insight here is that if active nodes can be provided with accurate and timely knowledge of the filters being executed upstream, then the overall filtering load can be reduced.

A reasonable approach for distributing the filter load involves using a two-pass filter specification technique. In this approach, the first (upstream) pass is identical to that of the original example. As before, receivers express interests as individual interest filters that are forwarded upstream towards the sender. Similarly, active routers along the path accept interest filters from both receivers and downstream active routers, and merge the filters into a single, more general interest filter that is then forwarded upstream. For clarity, we here refer to filters propagating upstream as *request* interest filters.

The second (downstream) pass begins once a request interest filter reaches the sender. The sender takes the request interest filter and performs an intersection with its advertised interest region (i.e., the region describing what information the sender can actually provide). The resultant, possibly reduced interest filter is then sent downstream towards the receivers as a *response* interest filter. Each active router along the forwarding path intercepts the response interest filter and uses it to calculate a supplementary filter for each downstream interface. A supplementary filter defines what additional filtering is to be done to best support the aggregate needs of downstream constituents for a given interface. The aggregation of the intercepted response interest filter with the supplementary filter for a given interface becomes the response interest filter for that interface, which is subsequently forwarded downstream.

Tree Filtering

A key scalability-enabling mechanism used in our approach involves organizing and executing interest filters as filter trees (i.e., decision trees.) Filter trees are created in typical tree building fashion by recursively dividing sets of interest filters into smaller and smaller subsets. Each division defines a simple branching decision to assess subset membership (e.g., a decision halfspace or a single covering filter containing all members of one subset). Branching decisions are represented as nodes in the tree, and the (disjoint) subsets resulting from each decision are represented as a set of links (i.e., branches) attached to the node (see Figure 3).

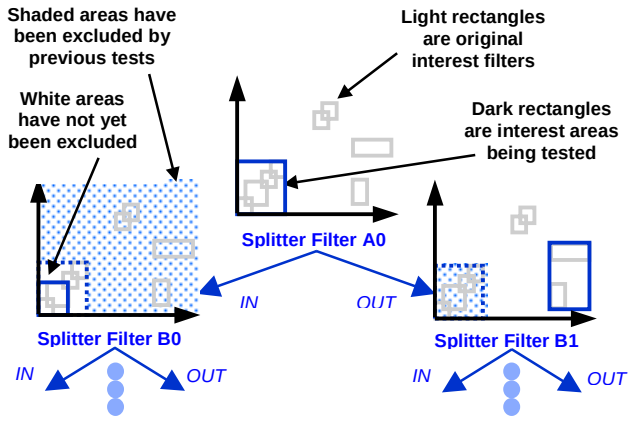

Figure 3. First two levels of a filter tree for a collection of eight interest filters (in gray)

Tree filters are applied by sequentially evaluating the branching decisions using the message content descriptor values and following the resulting links until a terminal node is reached. It is easily shown that the prescribed recursive construction yields a tree height proportional to the $\log_2$ of the number of interest filters, hence tree filtering provides an interest filtering approach that scales logarithmically with the number of filters. This contrasts quite sharply and favorably with the alternative approach of applying each interest filter in sequence – an approach that scales linearly with the number of interest filters.

Experimentally this approach has yielded results that in many cases achieve the theoretical lower bound on the average number of required decisions. Performance results obtained using the

prescribed tree filtering approach against randomly generated sets of interest filters and averaging over multiple runs show the desired logarithmic scaling characteristics (see Figure 4).

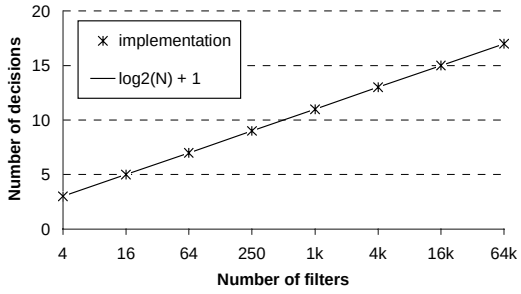

Figure 4. Tree filter logarithmic scaling property

Filter Simplification

As a final mitigation strategy for maintaining or improving throughput, the aggregation process may also elect to reduce the filter complexity with a simpler set of approximate filters. For example, a single larger interest region may "correctly" be used in lieu of a large number of small interest regions as long as the larger region contains all of the smaller regions. Although this potentially allows additional, unwanted network traffic through, it is still an acceptable tactic under the correctness criteria since no needed data are discarded.

Optimal filter simplification is for the most part impractical as the problem has NP-hard computational complexity (Burkard and Miatselski, 1999). To gain insight into the origins of this complexity, consider the number of ways that a set of *M* filters may be partitioned into *K* unique subsets, expressed by the Stirling number of the second kind. As such the computational complexity for optimal merging is then given by

$$O\,(K^M)$$

where $O()$ is the usual order function.

To address the computational complexity problem associated with filter simplification, we have developed a very capable, albeit suboptimal approach based on a technique used in designing Vector Quantizer (VQ) codebooks for data compression applications. In general, the VQ codebook design problem is one of constructing a low cardinality set (the set of VQ code vectors) that is used to represent a much larger cardinality set (the space of all possible input vectors) so that the distortion inherent in the representation is in some sense minimized. Clearly the notion of representing a large number of filters by a smaller number of filters is analogous with the VQ cardinality reduction objective, and the notion of doing so in a way that minimizes excess traffic or excess volume is aligned with the VQ optimization goals.

Drawing from this analogy, we have adapted the Fast Pairwise Nearest Neighbor (PNN) algorithm (Equitz, 1989) to perform filter simplification. The Fast PNN algorithm basically works as follows: the current set of elements (initially, the set of original filters for our purposes) is recursively split into smaller and smaller groups based on proximity (see Figure 5.) Splitting continues until the number of elements in each group reaches some objective value (e.g., 8 filters per group). Within each group, the pair of elements that if merged would produce the smallest amount of distortion (e.g., excess volume) is identified as the candidate pair for the group. Here, merging two elements translates to approximating the pair by a single element containing the pair.

The candidate pairs, one from each of the groups, are then ordered according to the amount of distortion, and some fraction (e.g., 30%) of the pairs are replaced by corresponding single elements that result from the merge process (see Figure 6). All elements are then lumped back into a single group, and the process repeats until the target number of elements is reached.

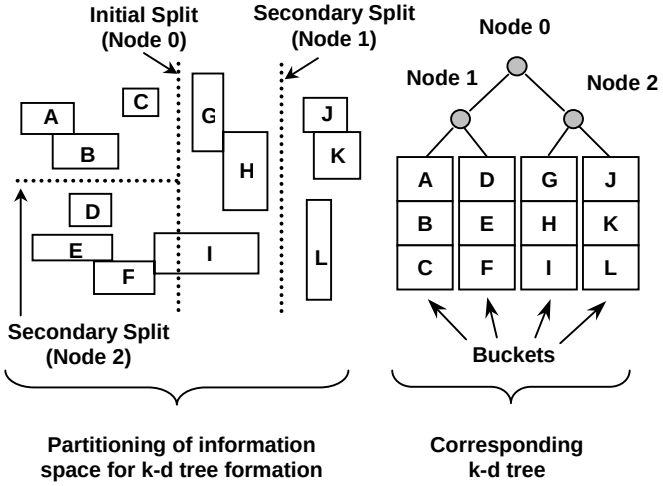

Figure 5. Splitting operations within the FastPNN algorithm

The Fast PNN algorithm is attractive in that not only have its results proven quite good operationally for VQ applications, but it has an *extremely* attractive computational complexity - particularly in comparison with the optimal approach. Specifically, the computational

complexity for reducing a number of elements *M* to some target number *K* using the Fast PNN algorithm is then given by:

$$O\,(M \log (M))$$

and is essentially independent of *K*.

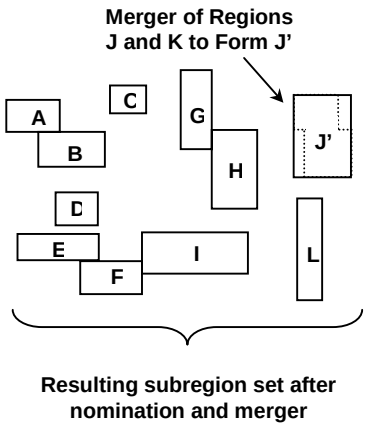

Figure 6. Merging operations within the FastPNN algorithm

Experimentally this approach has yielded excellent results. For example, testing using the Fast PNN algorithm to reduce an eight filter problem (see Figure 3) to a target number between two and seven filters yielded results identical to those produced by the optimal algorithm. Moreover, from a processing speed perspective, the optimal implementation took nearly three hours to reduce a set of 14 filters to a set of four filters. By comparison, the Fast PNN was able to reduce a set of *24* filters to a set of four filters in *under one one-hundredth of a second*. If the optimal algorithm was instead applied to the 24 filter problem, it would take an estimated *350 years*.

HARP: HLA ACTIVE ROUTING PROJECT

The active filtering approach described in this paper was implemented as part of a team effort referred to as the HLA Active Routing Project (HARP). The motivation for this effort was in response to DARPA's challenge problem of leveraging active networks to address distributed simulation performance and scalability limitations. The resulting demonstration architecture (see Figure 8) not only included the prescribed filtering, filter management, and active signaling components, but also included secure signaling support and was integrated with a DoD-standard High Level Architecture (HLA)-compatible distributed simulation Run Time Infrastructure (HLA/RTI). The architecture was used to demonstrate performance gains using the U.S. Military standard Modular Semi-automated Forces (ModSAF) real-time simulation package.

Signaling – The movement of filter information between hosts and routers is accomplished using a filter signaling protocol referred to as AFSP for Active Filter Signaling Protocol. AFSP was developed in cooperation with researchers at the University of Southern California Information Sciences Institute (USC/ISI). AFPS is in essence a derivative of the more commonly known ReSerVation Protocol (RSVP) (Braden, Zhang, Berson, Herzog, and Jamin, 1997) developed by USC/ISI.

Signaling Security – As the Filter Manager and Filtering mechanisms we are developing are fundamentally dynamic, programmable firewalls, such capabilities could cause substantial denial of service outages if used maliciously or even incorrectly. Consequently, provisioning active network security mechanisms to authenticate and authorize filter installation requests is crucial. Towards this end, all communications between AFSP components is handled by SANTS, designed and implemented by Network Associates, Inc. Laboratories (NAI Labs). SANTS is basically a secure version of the Active Network Transport System (ANTS) (Wetherall, Guttag, and Tennenhouse, 1998) developed by MIT.

HLA-compatible Run-time Infrastructure – Although one of our main objectives is developing a capability that requires *no* changes to HLA-compliant simulations, our approach *does* require modest modifications to the supporting RTI. Basically, the modifications provide needed information (i.e., interest region definitions, locations of interest region variables in the protocol data units (PDUs), and multicast session information) to the active network via the AFSP host module. When we began the HARP effort, source code for HLA-compliant RTIs was singly unavailable. Fortunately, researchers at the Georgia Institute of Technology agreed to extend their RTI to include HLA interfaces needed by simulations such as ModSAF and its variants, as well as the Declaration Manager (DM) and Data Distribution Manager (DDM) components needed for our work. It is to their RTI (Fujimoto and Hoare, 1998) that we have added the active filtering enhancements).

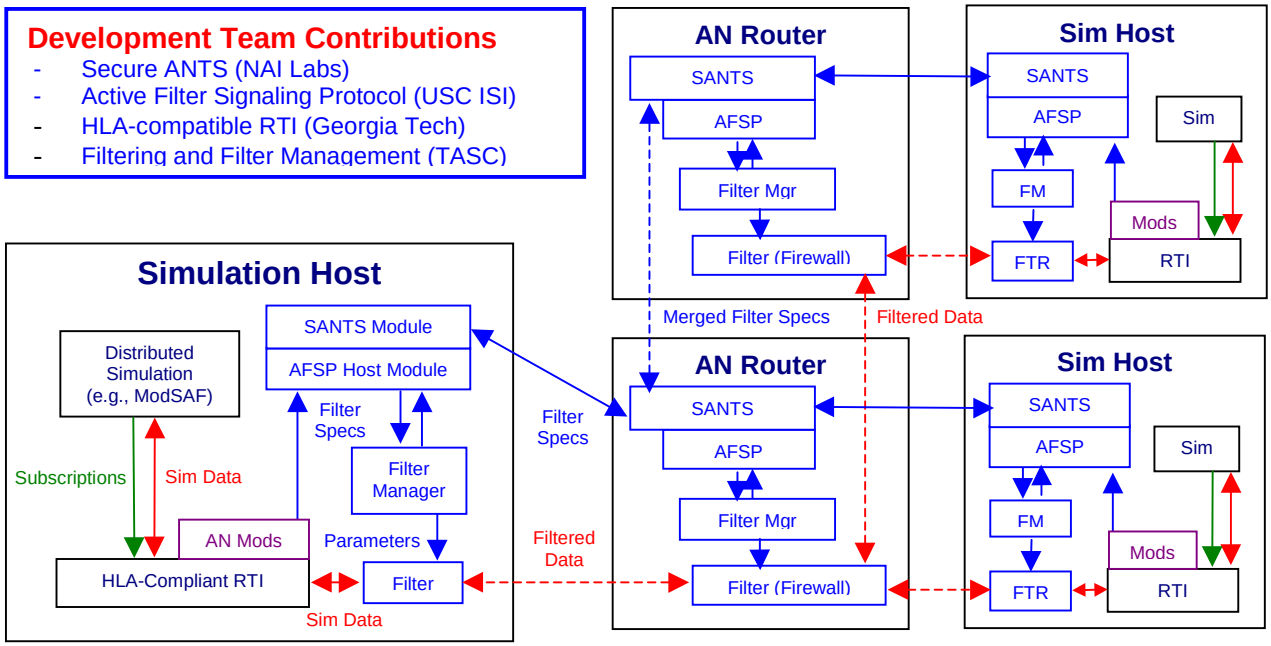

Figure 7. HARP Active Filtering Architecture

HARP Processing Example: ModSAF

The ModSAF user interface (see Figure 8) shows an example simulation run with two tank platoons and an aircraft wing overlaid on a map of the exercise area. The locations of the three groups are each circled on the map for clarity. At this point in the exercise, the group of aircraft (circled in the lower right side of the map) are just entering a tank platoon's engagement area.

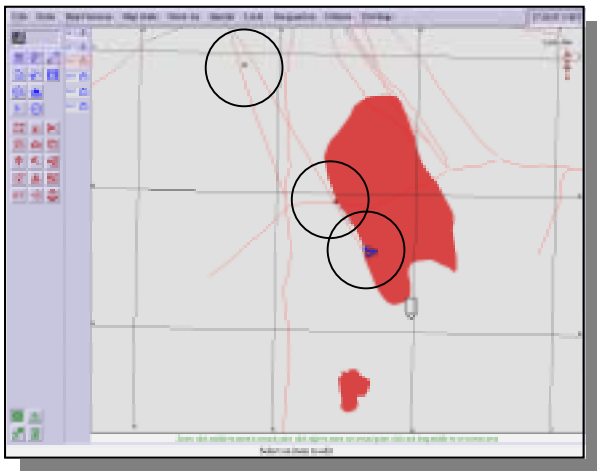

Figure 8. Example ModSAF application window

The corresponding interest filters for each of the simulated entities at the same point in time are illustrated using the HARP filter visualization tool (see Figure 9). The lower right filter, seen as a rectangle on the window, is the input (subscription) filter for the host. The other two filters, seen as rectangles in the upper left and center on the window (see Figure 9), represent output filters on the local host. The filters were installed by the other two ModSAF application hosts, which are simulating the tank platoons. Each of these ModSAF applications are only interested in data packets containing information about entities close to themselves.

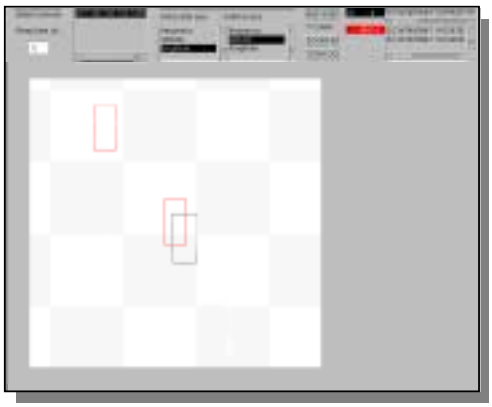

Figure 9. Example HARP region visualization tool window

Gauging Performance Impact: Packet Marking

When the active filtering implementation is functioning correctly, unwanted data packets are discarded at multiple points in the multicast distribution tree. However, this data packet loss complicates collection of performance statistics at active nodes and application hosts. In order to easily collect these statistics, a technique called *Packet Marking* is used. Here, rather than discarding data packets, data packets are instead marked in a prearranged location and forwarded.

Then, using combined packet marking and local filtering information at any point in the network, data packets can be classified in one of the following categories:

- *Wanted and Received (WR)* – The data packet is wanted and has not been marked. This case represents packets that have been correctly forwarded through the network and would not be inadvertently filtered out within the network.
- *Not Wanted and Not Received (NWNR)* – The data packet is not wanted and has been marked. This case represents packets that would have been correctly filtered out within the network if we were not using packet marking to gather statistics.
- *Not Wanted and Received (NWR)* – The data packet is not wanted but has not been marked. While undesirable, this case can arise from inexact filtering and does not technically violate the correctness property.
- *Wanted and Not Received (WNR)* – The data packet is wanted but has been marked. Packets in this category are a result of the correctness property being violated and constitute a failure of the interest filtering system.

A running histogram of these different packets types can then be used to track network performance and behavior as a function of time, as illustrated by the HARP packet marking visualization tool (see Figure 10)[2].

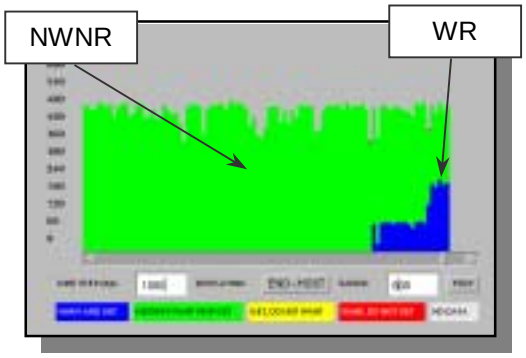

Figure 10. HARP packet marking tool window

In particular, Figure 10 shows the packet marking results captured at the same moment in time and on the same host as for 0 and Figure 9. The left portion of the graph shows that all data packets received on the local host before the group of aircraft entered the tank platoon's engagement area were NWNR data packets. As both tank platoons sending the data packets were not in the aircraft's interest region, this is correct behavior.

The right side of the graph in Figure 10 shows the changes after the aircraft have entered the lower tank platoon's engagement area. Data packets from the upper tank platoon continue to be NWNR data packets. However, data packets from the lower tank platoon are now WR data packets. The reduction in data packet processing load at the local ModSAF client in this example is pronounced and is similar at the other ModSAF applications in this example. Moreover, the improvement in network efficiency is rather remarkable. The area to the left shows that all of the roughly 450 packets per second being pumped into the network by the two tank platoon simulation hosts would in fact not be delivered to the host simulating the aircraft. The area to the right (after the aircraft have entered the first tank platoon's engagement area) still shows a 2:1 reduction in inbound traffic at the receiving host, a ratio that could in fact be as high as *K*:1 were the number of tank simulation hosts to increase to some number *K.*

As a final comment, we note the explicit absence of NWR and WNR packets, showing that the filtering is both correct and exact.

Active Node Filtering Performance

Although our goal is to leverage programmable hardware (e.g., Nortel's Accelar Platforms), equipment capable of supporting the types of user-defined filtering needed for our program is not yet available. In order to evaluate the potential performance impacts of user filtering as well as support interim demonstrations, we therefore developed and integrated the needed capabilities as extensions to the kernel-based firewall code (*fw*) within Linux-based software routers.

The results from our initial experiments using this router capability were extremely encouraging, (see Figure 11). Throughput measurements made using the modified Linux kernel on a 300MHz Pentium II equipped with a pair of 100 Mbps interfaces to route packets between a multicast sender and a multicast receiver connected to the two interfaces respectively. The experiment consisted of the send host multicasting packets with interest information embedded within the

[2] We emphasize that the prescribed packet marking technique is only used to assess the correctness and performance of the developed capability and would not be used operationally since the intent is to discard unneeded packets rather than simply account for them.

packet payload as fast as possible, and the receive host simply counting the number of packets received over the duration of the transmission. The software router between the sender and receiver was configured with N-1 "miss" filters (i.e., filters that did not match the information within the packet) and a final "hit" filter (i.e., a filter that did match the information within the packet) with the rule for the "hit" filter being to forward the packet. Hence the router was forced to evaluate N filters before forwarding the packet, with values of N varying from 10 to 2000.

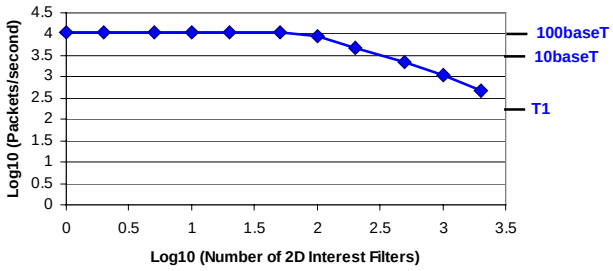

Figure 11. 2D Interest Filtering Performance on 300 MHz PII

Clearly even this relatively slow router was able to sustain full 100 Mbps line rate throughput for values of *N* well beyond 50, and was able to sustain T1 speed or better throughput (1.5 Mbps, typical of WAN links) out to around 5000 filters. The implication of this *linear* filtering test (i.e. where all filters are applied in a single linear sequence) is that there is strong reason to believe that the prescribed *logarithmically* scalable tree-based active interest filtering approach will easily support the needs of large simulations.

CONCLUSIONS

We have demonstrated that distributed simulation performance and scalability can be substantially improved by using Active Networks to implement HLA interest management functions directly in the network where they are most effective. In particular, under this effort we have demonstrated:

- An active network-based dynamic active interest filtering capability that leverages the natural publish and subscribe mechanisms employed within the HLA
- Clean integration with an HLA-compatible RTI requiring *no* changes to the HLA API: code modifications were exclusively limited to the RTI itself.
- Successful operation under ModSAF – a real HLA-based simulation of significant interest to the distributed simulation community.
- Substantial, quantifiably improved simulation performance with no added latency or reduction in throughput
- A suite of performance enhancement techniques that should enable the developed capabilities to scale to large numbers of users

In summary, active interest filtering is a promising, highly attractive approach for achieving large scale distributed simulation.

REFERENCES

Braden, R. Ed, L. Zhang, S. Berson, S. Herzog, and S. Jamin, Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification IETF RFC 2205, September 1997

Burkard, R., and M. M. Miatselski, Volume Maximization and Orthoconvex Approximation of Orthogons, Computing, 63, 1999

Defense Modeling and Simulation Office (DMSO), Department of Defense High Level Architecture Interface Specification, Version 1.3, April 1998

Equitz, W. H., A new Vector Quantization Clustering Algorithm, IEEE Trans. ASSP, Vol. 37, No. 10, October 1989

Fujimoto, R. M., and P. Hoare, HLA RTI Performance in High Speed LAN Environments, Proceedings of the Fall Simulation Interoperability Workshop, September 1998

Kasera, S., S. Bhattacharyya, M Keaton, D. Kiwior, S. Zabele, J. Kurose, and D. Towsley, Scalable Fair Reliable Multicast Using Active Services, IEEE Network Magazine, Vol. 14, No. 1., January-February 2000.

Tennenhouse, D., J. Smith, W. Sincoskie, D. Wetherall, and G. Minden, A Survey of Active Network Research, IEEE Communications Magazine, Vol. 35, No. 1, pp80-86. January 1997.

Van Hook, D., S. Rak, and J. Calvin, Approaches to Relevance Filtering, 94-11-144, Proceedings of the Eleventh Workshop on Standards for the Interoperability of Distributed Simulations, September 26-30, 1994.

Wetherall, D., J Guttag, and D. Tennenhouse, ANTS: A Toolkit for Building and Dynamically Deploying Network Protocols, IEEE OPENARCH'98, San Francisco, CA, April 1998