Enabling Large-scale Simulations: Selective Abstraction Approach to The Study of Multicast Protocols

Polly Huang, Deborah Estrin, John Heidemann

USC/Information Science Institute

University of Southern California

4676 Admiralty Way, Suite 1001

Marina del Rey, CA 90291

Phone: (310)822-1511 ext. {222, 253, 708}

Fax: (310)823-6714

{huang, estrin, johnh}@isi.edu

Abstract

Due to the complexity and scale of the current Internet, large-scale simulations are an increasingly important tool to evaluate network protocol design. Parallel and distributed simulation is one appropriate approach to the simulation scalability problem, but it can require expensive hardware and have high overhead. In this study, we investigate a complimentary solution to large-scale simulation -- simulation abstraction. Just as a custom simulator includes only details necessary for the task at hand, we show how a general simulator can support configurable levels of detail for different simulations. We develop two abstraction techniques, centralized computation and abstract packet distribution, to abstract network and transport layer protocols. We demonstrate these techniques in multicast simulations and derives centralized multicast and abstract multicast distribution (session multicast). We show that our abstraction techniques each help to gain one order of magnitude in performance improvement (from tens to hundreds to thousands of nodes). Although abstraction simulations are not identical to more detailed simulations, we show that in many cases these differences are small. We show that these differences result in minimal changes in the conclusions drawn from simulations in reliable multicast simulations.

Keywords: Abstraction Techniques, Packet Network Simulation, Simulation Scalability, Multicast Simulation, Protocol Evaluation

1. Introduction

Modeling and simulation traditionally have been the two approaches to evaluate network protocol designs. However, modeling is often intractable with today's large networks and complex traffic patterns, so researchers have turned increasingly to simulation. In order to evaluate wide-area protocols with thousands of nodes, we must be able to perform large-scale simulations.

General purpose network simulators (such as ns-2 [25]) make simulation easier by capturing characteristics of real network components and providing a modular programming environment, often composed by links, nodes and existing protocol suites. For instance, a link may contain transmission and propagation delay modules, and a node may contain routing tables, forwarding machinery, local agents, queuing objects, TTL objects, interface objects, and loss modules. These composable modules provide a flexible environment to simulation network behavior, but depending on the level of details a particular simulation is investigating, these details may or may not be required. Unfortunately, this modular structure can result in significant resource consumption, especially when the simulation scenarios grow.

One approach to the scalability problem is to apply parallel and distributed simulation, dividing tasks into parts coordinated over numbers of machines. Parallelism can improve simulation scale in ratio to the number of machines added, but this linear growth is not sufficient to add several orders of magnitude scaling needed. Parallel simulation can also require hardware which is not widely available or expensive.

A complimentary solution is to slim down simulations by abstracting out details. The basic idea is to analyze a particular set of simulations, identify the bottleneck, and eliminate it by abstracting unnecessary details (i.e., keeping the simulator slim). The risk of abstraction is that simulation results may be distorted due to the abstraction; users must be careful that their simulation results are not changed. We address this problem by providing identical simulation interfaces for detailed and abstract simulations, allowing users to make side-by-side comparisons of their simulations at small scales. When the abstraction is validated, they can then use it for very large-scale simulations.

Simulation scaling involves a process of selective abstraction which is often application specific, and raises two interesting questions -- "what to abstract to make simulations scaleable?" and "what is the effect on simulation results?". We examine these questions, using multicast simulations as examples.

This work shows that abstracting details can speed up and reduce memory consumption significantly while the results of simulations are not crucially affected. Application of our abstractions and the techniques may allow the research community to perform large-scale simulations that were impossible before, and to perform previously achievable large-scale simulations at much lower cost to the same user. As a result, this work can enable the Internet community to debug designs and to evaluate protocol performance in large-scale scenarios.

After discussing related work, we present the techniques used to abstract details, particularly for multicast simulations, and a guideline to perform abstract simulations. We then discuss results comparing performance and accuracy for the abstract and detailed versions, and, finally, SRM (Scalable Reliable Multicast) simulations are studied to demonstrate the use of guideline and the abstraction techniques.

2. Related work

There has been a great deal of work on simulation techniques. Here we briefly summarize priori work in data structure, parallel and distributed simulation, abstraction, and hybrid simulation.

Various scheduling algorithms and event queue structures (e.g., calendar queue, heap, and hybrid event list) have been proposed to speed up sequential simulation. Unfortunately, they do not scale to the number of events (packets) triggered by a complex and large-scale network like the current Internet. Ahn and Danzig estimated [11] that “five minutes of activity on a network the size of today’s Internet would require gigabytes of real memory and months of computation on today’s 100 MIP uniprocessors.”

An alternative to sequential simulation is parallel and distributed simulation. It exploits the cost benefits of microprocessors and high-bandwidth interconnections by partitioning the simulation problem and distributing executions in parallel. The distributed simulations require techniques such as conservative and optimistic [4-7] synchronization mechanisms to maintain the correct event ordering. Consequently, the simulation efficiency may be degraded due to the overhead associated with these techniques . In addition, the typical simulation algorithm does not easily partition for parallel execution. Ohi and Preiss [2,3] investigated several block selection policies and found limited speedup and possibly degraded performance when there were a large number of unique event types. Although parallel and distributed simulation is useful when large computers are available, alternative techniques such as abstraction are also needed to make very large scale simulations.

Ahn and Danzig [11] proposed to abstract packet streams (Flowsim) for packet network simulations and proved that Flowsim could be adjusted to the desired simulation granularity and help to study flow and congestion control algorithms more efficiently. However, Flowsim only abstracts one aspect of network simulation. We present two other abstraction techniques and hope to generalized them for a wider range of network protocols.

In hybrid simulation models [12], both discrete-event simulation and analytic techniques are combined to produce efficient yet accurate system models. There are examples of using hybrid simulations on hypothetical computer systems. Discrete-event simulation is used to model the arrival and activation of jobs, and a central-server queuing network models the use of system processors. The accuracy and efficiency of the hybrid techniques are demonstrated by comparing the result and computational costs of the hybrid model of the example with those of an equivalent simulation-only model. Our abstract simulation can be thought of as an application of hybrid simulation to networking.

3. Abstraction techniques

Large-scale simulations are prevented because of resource constraints, typically in CPU consumption and memory usage. Through abstraction we reduce resource consumption, enabling large-scale simulations. Abstraction is possible because most protocol architectures are layered such that one protocol makes use of another. In design or evaluation of a level-n protocol, we need information provided by level n-1 and below, but not necessarily (depending on the research questions) all the details of the lower level protocols. For instance, a multicast transport protocol may need multicast routing tables in order to forward multicast packets, but not the detailed exchange of messages required to generate those routing tables in an actual network. If we abstract away unnecessary details, the memory and time consumption can be conserved and used to simulate larger-scale scenarios. In this section, we present two abstraction techniques and our general approach to abstraction. We use multicast simulations as our example.

Multicast supports transfer of information from one or more sources to multiple receivers (multicast group members). Current Internet multicast protocols transfer information along distribution trees rooted at the group sources and spanning the group members. Various multicast routing protocols establish these multicast trees in different ways: broadcast and prune (e.g., DVMRP [13]and PIM-DM [14]), membership advertisement (e.g., MOSPF [22,23]), and rendezvous-based (e.g., CBT [15], PIM-SM [16-21], and BGMP[35]). Broadcast and prune multicast routing protocols flood data packets all over the network and then prune back branches that do not have members. They are suitable for groups with dense member distribution, and, thus, also called dense mode multicast routing protocols. Membership advertisement multicast routing protocols advertise membership and topology information everywhere in the network and calculate the multicast tree routes according to this information. Due to the domain-wide advertisement of membership and topology information, these multicast routing protocols do not scale to large networks and are most suitable for intra-domain multicast. Rendezvous-based multicast routing protocols build multicast trees by sending explicit membership join messages toward a well-known rendezvous point and sources. These protocols have good scaling properties and are, therefore, suitable for inter-domain multicast routing and sparse member distribution. Much of the current IP multicast infrastructure uses dense mode multicast protocols such as DVMRP and PIM-DM, so we use a DVMRP-like dense mode as the standard detailed multicast for the simulations.

Our detailed implementation of dense-mode multicast does not scale well as numbers of nodes and group members rise due to the overhead of flood-and-prune message processing. Our first abstraction technique, centralized multicast, avoids this message exchange by centrally computing multicast routes. The approach can be thought of as an example of abstracting network-layer protocol details that otherwise create network state in a distributed manner. Instead of setting timers, sending messages and calculating states distributively, centralized multicast computes the result of this exchange and updates routing in multicast trees instantly as described in the next section. Although we applied this centralized computation to multicast routing, the same approach can be used for similar protocols (for example, unicast routing protocols). The centralized computation technique conserves a significant amount of memory and time at the cost of a slight difference in route convergence latency when group membership or topology changes. See the next section for details.

Centralized route computation allows us to scale to 100s of nodes, but at thousands of nodes the overhead of general purpose per-node and link data structures becomes very large. To reduce the node and link structure, we developed abstract packet distribution, our second abstraction technique. Abstract packet distribution can be used to abstract transport-layer protocols which transmit packets over established routes. Instead of sending packets through a series of queues and links, abstract packet distribution computes delay and loss characteristics and directly schedules receive events at the group members according to these characteristics. Consequently, nodes and links become lightweight data structures instead of very general purpose (but flexible) objects. Applying this technique to multicast sessions, abstract multicast distribution sets up a direct connection between a source and its receivers, computes loss, delay, and TTL characteristics for each source and receiver pair, and schedules packets delivery events accordingly. Details of the abstract multicast distribution is presented in the following section. We hope to apply the same abstract packet distribution to other transport protocols (e.g., TCP) to reduce resource consumption. The abstract packet distribution technique significantly improved the simulation performance with increased topology size, but with observable difference in packet end-to-end delay when the network is congested as described in the next section.

Our experience in preparing and designing the two abstraction techniques can be generalized into the following guidelines to help users conduct their own abstract simulations:

1. Define scaling factors/dimensions(varying input) and measurement metrics(wanted output)
2. Start from small-scale detailed simulations
3. Profile simulations to find bottleneck
4. Adapt techniques to avoid simulation bottleneck
5. Verify simulations in small-scale in detailed mode
6. If the difference between the two versions is not crucial to the research question, simulate larger-scale simulations using the improved version

We demonstrate these techniques in the next section.

4. Systematic Comparison

In this section, we describe the original dense mode multicast implementation in detail, as well as our two abstraction approaches (centralized route computation and abstract packet distribution). Then, we present the simulation results to compare the scaling behavior of all three, finding that the abstraction techniques substantially improve the scaling properties of the simulations . Dense mode with detailed packet distribution scales the worst, centralized multicast with detailed packet distribution scales better, and the abstract multicast distribution scales the best. Abstract simulations have small differences from detailed simulations, so finally, we illustrate some of the simulation relevant differences between dense mode and centralized multicast, and between detailed packet distribution and abstract multicast distribution.

4.1 Mechanism Detail

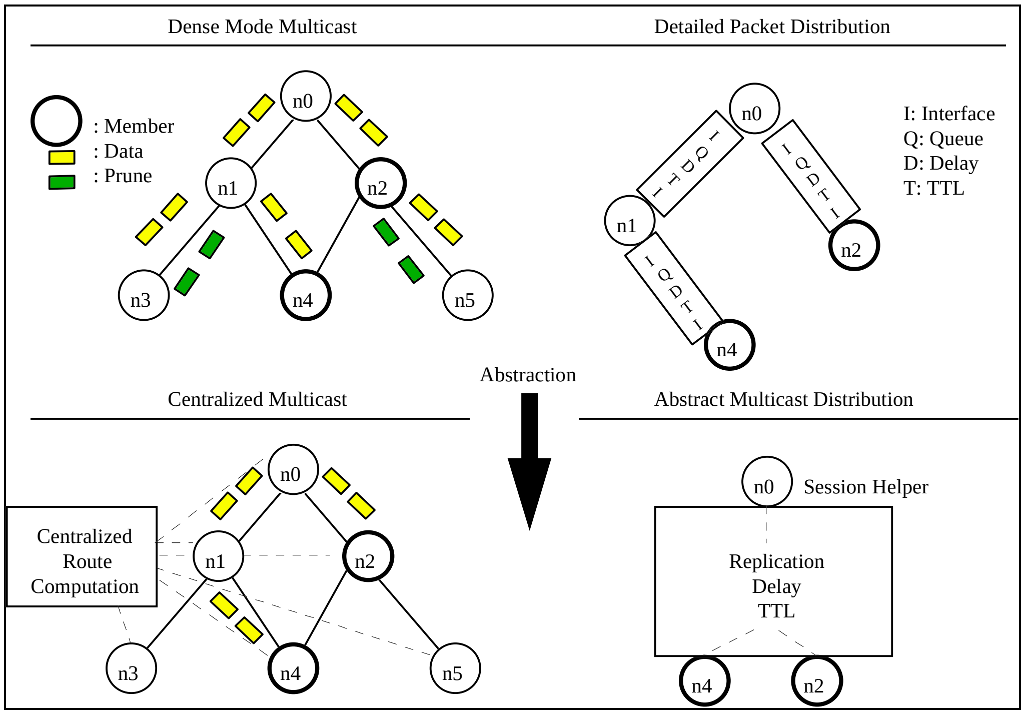


Figure 1. Illustration of the Detailed and Abstract Simulation Mechanisms

4.1.1. Dense Mode Multicast & Detailed Packet Distribution

The ns-2 implementation of dense mode multicast closely follows real-world implementations in terms of message exchanges. Each dense mode multicast agent maintains a parent-child relationship to all other agents by a triggered checking to neighbors' routing tables. When a cache-miss occurs in a node (e.g., a source starts to send packets), this node upcalls its local dense mode agent to install a multicast entry according to the source and group addresses in the packet header. The dense mode multicast agent will insert only the child links indicated in its parent-child relationship to be the outgoing interfaces. After the multicast forwarding entry is installed, packets are forwarded to the outgoing interfaces. When a multicast packet reaches a leaf node that does not have any local member, a prune message is sent upstream to start a prune timer in the upstream dense mode agent for the particular outgoing interface. Within this prune timeout period, multicast packets will not be forwarded on this outgoing interface. When a member joins a group and there exists a multicast forwarding entry for the group but no local or downstream members, a graft message is sent upstream to cancel a prune timer (if there's any) so multicast packets can be received at this member. Similarly, when a member leaves a group and there are no local or downstream members, a prune message is sent upstream to prune off this branch.

Packets are forwarded through a series of objects that are linked to mimic detailed packet forwarding machinery in the real network. When packets enter a node, an entry classifier decides which next classifier to go to depending on the destination address in the packet header. If the address is a unicast/multicast address (mask and mask length), packets are forwarded to a unicast/multicast classifier which maintains a hash table of (source, destination, outgoing target, incoming interface if multicast) tuples. These tuples are installed by the unicast/multicast routing protocol. The classifiers then hash on the source, destination, or incoming interface if multicast, to decide to which outgoing target to forward the packets. In unicast classifiers, outgoing targets are typically the network interfaces that the packets are supposed to be forwarded onto. However, in multicast classifiers, outgoing targets are replicators, and the replicators copy the packets and forward on to several outgoing network interfaces. Each network interface is connected to a queue object which holds packets inside the queue if the link is occupied. When the link is clear, packets are forwarded onto a transmission and propagation delay module which delays the packets with the propagation delay and transmission delay given in the simulation configuration. Finally, the packets reach a TTL checker which decrements the TTL field in the packet header and passes to the receiving network interface. The receiving network interface labels the incoming interface field in the packet header and forwards to the entry classifier of the receiving node.

4.1.2. Centralized Multicast - Abstracting Route Computation

The detailed protocols described in Section 4.1.1 were helpful in developing these protocols and validating the simulator, but they often are unnecessarily specific. Our centralized multicast abstraction eliminates much of this message exchange.

The centralized multicast computation agent keeps track of source and member lists for a group. Therefore, when a member joins a group, a multicast tree branch is installed toward all sources for the group until it merges with the original tree. Similarly, when a member leaves a group, a multicast tree branch is pruned until it reaches the original tree. When a source starts to send packets to a group, the entire multicast tree is installed according to the member list of the group. There are no prune timers associated with outgoing interfaces, and whenever there is a topology change, all the multicast trees are re-installed.

The periodic broadcast and prune, parent-child relationship maintenance, and message exchange used in the dense mode multicast are eliminated in centralized multicast.

4.1.3. Session Multicast - Abstracting Packet Distribution

Another area with possibly unnecessary detail is ns' link and node structures. Our second abstraction, abstract packet distribution, eliminates this overhead.

The abstract multicast packet distribution avoids set-up and maintenance of multicast forwarding entries altogether. Instead, source and members are directly connected with appropriate delay and loss characteristics. In the simulated version, the propagation delay and bandwidth are calculated and represented between a source and each of its members.

Users may introduce loss into a simulation (through the use of *loss modules* in ns-2) to study protocol behavior. Packet loss in the detailed simulation much be reflected in the abstraction. In particular, losses on a link will correlate to losses to any recipients downstream of that link. We reflect this dependency in our abstract simulation. For example, the left diagram in Figure 2 presents a multicast tree with source n0 and members n1-n6, and an error modules is inserted in each of link n0-n1, n1-n3, and n2-n6. The right diagram in Figure 2 shows the equivalent error dependency retained for the multicast tree where n6 is dependent on e2, n3 is dependent on e1, and n1, n4 and e1 are dependent on e0.

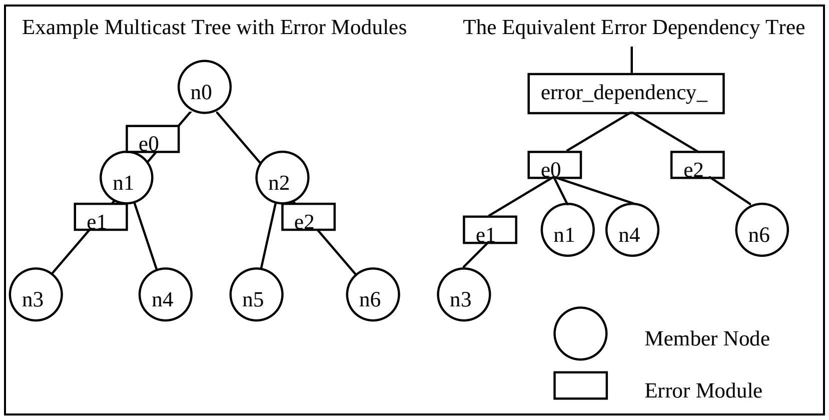

Example Multicast Tree with Error Modules
The Equivalent Error Dependency Tree
n0
e0
n1
n2
e1
e2
n3
n4
n5
n6
error_dependency_
e0
e2
e1
n1
n4
n6
n3
Member Node
Error Module

Figure 2. An Example of Error Dependency in Abstract Multicast Distribution (Session Multicast)

In abstract multicast packet distribution, all the replication, delay, loss and forwarding machinery are combined into one compact multicast session helper agent. As a result, the original link structure (a sequence of interface, loss, queuing, delay, and TTL object) is reduced to delay and bandwidth values, and the original node structure, a combination of classifiers, is reduce to node id and port id. Of course, our simple form of abstract multicast packet distribution is not suitable for studying queuing behavior.

4.2. Performance Comparison

The purpose of our abstractions is to improve performance in the face of large scale simulations. The numbers of nodes, members, and groups are the three factors that affect the scale of multicast simulations. The following three sets of tests explore each of the three dimensions of a standard case, 300 nodes (average degree 1.77), 30 groups, and 30 members. Memory and running time are measured to compare the performance among the three versions of simulations.

Nodes: 30 group, 30 members, and 100 - 500 nodes (average degree 1.77, transit-stub[1])

Members: 300 nodes, 30 group, and 10 to 50 members

Groups: 300 nodes, 30 members, and 10 to 50 groups

In the set of simulations with increasing numbers of nodes, we use fixed numbers of groups and members. 30 group sources are randomly selected and for each group 30 members are randomly selected as well. Several member agents may co-exist in the same node. In addition, the number of links increases proportionally to the number of nodes to maintain a constant degree of connectivity 1.77, and, using GT-ITM (Georgia Tech - Internetwork Topology Model) [24], we are able to create random transit-stub topologies that are closer to the real world network structure. In the second set of simulations, the number of nodes and groups are fixed to 300 and 30, and, for each of these 30 groups, 10 to 50 members are selected randomly. Similarly, the number of nodes and members are fixed in the third set of simulations, and 10 to 50 group sources are randomly selected to send packets to 30 randomly selected members in a 300 node topology.

[1] Transit-stub topologies are formed by connecting many stub domains through few transit domains, closely model the structure of real networks.

4.2.1. Memory Usage

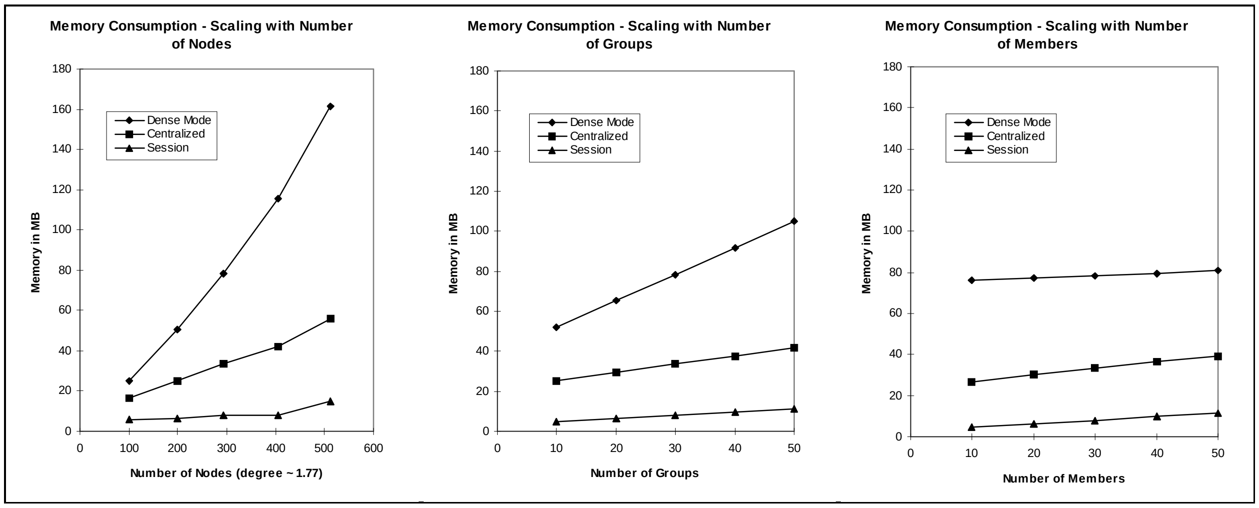


Figure 3. Comparison of Memory Usage

In the first set of simulations (Figure 3), the periodic flood and prune in dense mode multicast has a dramatic effect on the memory consumption (30 topology-wide flood and prune per period). Especially, when the number of nodes increases, the area for flood and prune becomes larger as well, which contributes to the high memory usage for dense mode multicast. Centralized multicast replaces flood-and-prune messages, eliminating this source of overhead (compare memory usage between dense mode and centralized in all three cases). However, centralized multicast still experiences a significant growth in memory consumption with the growth of topology size because of ns' detailed representation of links and nodes. By abstracting this representation we get both better absolute memory usage (compare the difference between session and the other simulations) and incrementally less cost as scale increases (compare the slopes of session to the others as number of nodes or groups change).

An exception to reduced incremental costs is shown as we increase the number of members. All three simulations grow at similar rates here (although the abstractions have far less absolute memory usage). This is due to the random selection of the members. If the member distribution is diverse (fairly random), then increased number of members does not necessarily imply the multicast tree spans a wider range. In another words, the area of flood and prune in dense mode does not necessarily grow or shrink. On average, the flood and prune area should stay constant. However, we expect centralized multicast to consume slightly more memory than dense mode when all or most nodes are members, because the flood and prune overhead in dense mode is reduced and centralized multicast carries extra global states required for route computation. The slight increases we observe in all three versions of simulations is due to the multicast states in the extra members.

In summary, we have shown that abstraction can save substantial amounts of memory, allowing larger simulations to be run. The benefits of these approaches vary depending on what dimensions of scale is being pushed. We see large absolute and incremental benefits when numbers of nodes and multicast groups rise, but only absolute benefits when the number of group members change.

4.2.2. Time Usage

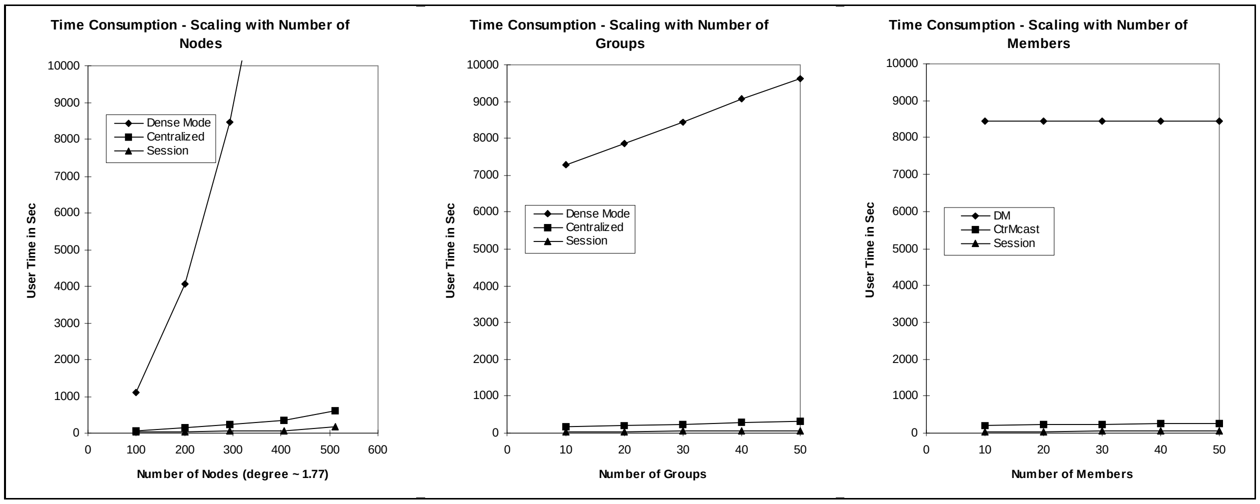


Figure 4. Comparison of Time Usage

Dense mode scales poorly in CPU usage (Figure 4) due to the flood and prune over the entire topology which causes an enormous amount of packet/event scheduling (proportional to the number of nodes and groups). Centralized multicast scales much better because it avoids the scheduling of packets that flood everywhere. Abstract multicast distribution does even better by avoiding hop-by-hop packet processing overheads.

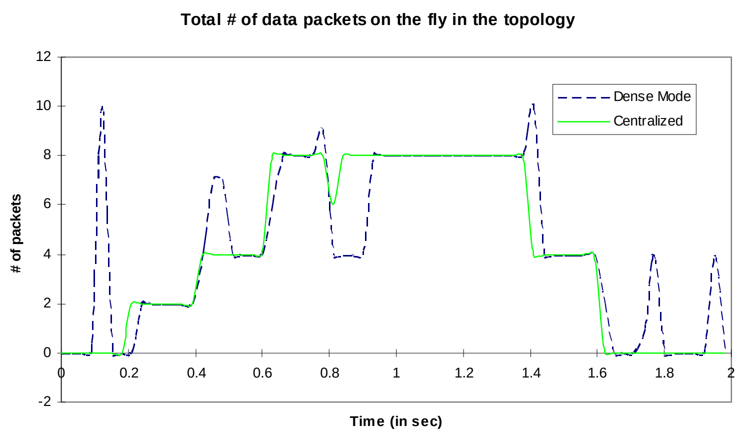
4.3. Behavior Difference

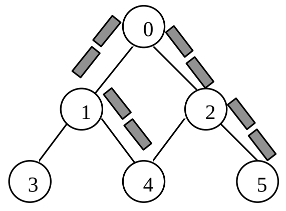
The performance gains described in Section 4.2 are possible because abstractions remove some protocol details. The simulation user must understand if these details affect the accuracy of the simulation's end-result. In this section we characterize the differences our abstractions introduce, and then in Section 5 we examine the effect these differences have on one application (SRM).

4.3.1. Route Computation - Dense Mode vs. Centralized Multicast

The centralized multicast abstraction replaces routing message exchange with a central computation. Although the end results of the algorithms are identical (since both implement the same reverse shortest-path tree algorithm), transient behavior can be different. In a detailed dense mode simulation messages propagate topology change information through the tree at link speeds, while with centralized abstraction topology changes become global known the instant they occur.

To demonstrate the difference between the protocols, we simulate various group events (join, leave, link down, link up) on a simple topology just to demonstrate the difference in the two multicast implementations' behavior. During the simulation, we periodically count the numbers of data packets on the entire topology and show the collected data on the following figure, we referred to as the behavior chart. This behavior chart allows us to observe the protocol dynamics, traffic overhead as well as the event convergence latency, all in one 2-D graph.





time	events
0	source 0 starts
0.1	DM's first periodic flood & prune
0.2	member 1 joins
0.4	member 4 joins
0.6	member 5 joins
0.8	link 0-1 goes down
1	linke 0-1 comes back on
1.2	member 1 leaves
1.4	member 4 leaves
1.6	member 5 leaves

Figure 5. Difference Between Dense Mode and Centralized Multicast

The dark dashed line represents dense mode multicast behavior. The light solid line represents centralized multicast behavior. Dark spikes (for example, at time 0.1, 0.4, and 0.8) nicely demonstrate the periodic flood and prunes. In both lines, we can see that the steady state of the lines rise by a few packets when members not currently in the multicast tree join the group (for example, at time 0.2, 0.4, and 0.6), and the lines drop a few packets lower when members leave the group (for example, at time 1.2, 1.4, and 1.6). Furthermore, when a link on the tree fails (for example, at 0.8), we can see that the lines drop a few packets lower as well, and, when the link comes back on, the lines jump back to the original level. Comparing the two lines, other than the extra flood and prune overhead in dense mode, we also observe that the route convergence latency for join/leave and network events are quite different depending on the update sensitivity of the two multicast implementations.

From the above, we conclude that the centralized multicast is not appropriate when examining detailed multicast behavior during transients (e.g., amount of loss during transients and amount of bandwidth consumption by flood and prune). However, centralized multicast should give no difference to simulation results when the transient delay/behavior is not an issue.

4.3.2. Packet Distribution - Detailed packet vs. Abstract Multicast Distribution

Abstract multicast distribution replaces hop-by-hop message passing with direct end-to-end estimated propagation delays. Simulations with abstract packet distribution therefore will not include network queuing delays. We therefore expect the abstract simulations to be the same as detailed simulations when there is no cross traffic but to fail to module queuing delays when cross traffic is added. To demonstrate this effect we compare end-to-end estimated delay in abstract and detailed simulations of the same network with 1 and 10 multicast groups. These results can be seen in Figure 6.

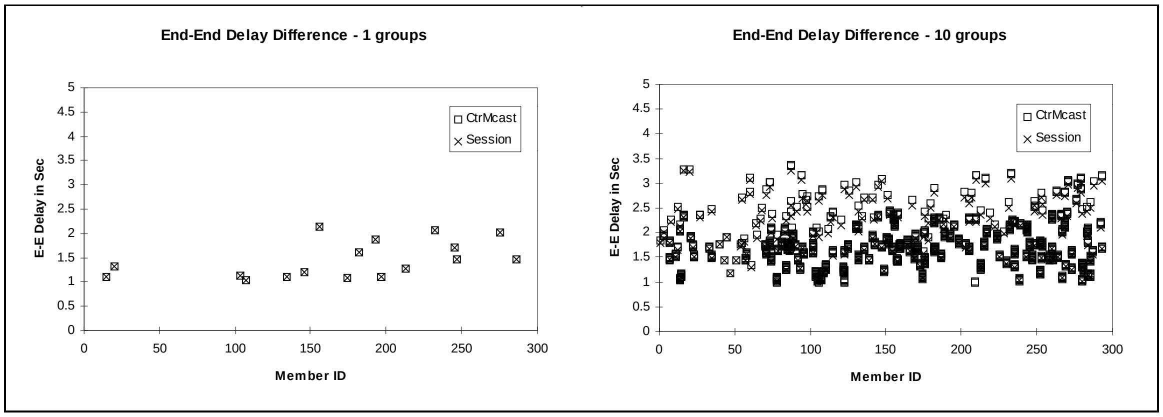


Figure 6. Difference Between Detailed Packet Distribution and Abstract Multicast Packet Distribution

In the uncongested scenario on the left, link capacity is high enough to carry the incoming traffic smoothly, so there is no queuing contributing to the end-to-end delay which is closely modeled by the abstract multicast distribution. Therefore, the results show no difference in end-to-end delay. However, in the scenario on the right there is cross-traffic and so queuing delay. Queuing delay is omitted by current abstract multicast distribution, causing differences in end-to-end delay when queue builds up.

This example suggests that abstract multicast distribution must be used carefully when simulations involve very high source rates or cross traffic(i.e., congested network). For example, abstract multicast distribution should not be used with congestion control protocols because there must be congested network components in order to exercise the congestion control mechanism.

5. Impact on Simulation Studies - Case Study SRM

The last section showed that abstraction can give much better performance for large simulations, but that in some cases it produces slightly different results. To determine if the differences caused by abstraction would affect the end-results of a real protocol study, this section examines SRM timer behavior (previously explored in [26, 31-34]) with both detailed and abstract simulations.

5.1. SRM Mechanism

SRM (Scalable Reliable Multicast) [26] is a reliable multicast transport protocol. Each member detects losses individually and issues multicast requests for retransmission per loss. Any member who successfully receives the packets may multicast a retransmission to recover losses for the requesting node.

If members just fire requests upon detecting losses, there will be duplicate requests multicast over the entire tree. To reduce the number of duplicate requests, members wait for some time before sending requests. The timers have both deterministic and probabilistic parts. The deterministic part is decided according to the distance between the source and requester, so that the requester closest to the source tends to fire requests first. The requests arriving at other losing members suppress duplicate requests (e.g., string topology). The Probabilistic part is to suppress request under the circumstances that several requesters are equally distant from the source (e.g., star topology). Recoveries are handled analogously.

The distance information used to decide the deterministic part of timers is obtained by periodic exchange of session messages, similar to the NTP (Network Time Protocol) algorithm. Member A records T_{sendB} when sending a session message to member B. Member B records T_{rcvA} when receiving the session message from A. After a while, member B records T_{sendA} when sending another session message to member A, and, similarly, member A records T_{rcvB} upon receiving the session message. The one-way distance is estimated by:

$$(T_{rcvB} - T_{sendA} + T_{rcvA} - T_{sendB}) /2$$

5.2. SRM Simulations

We simulate the SRM mechanism on top of detailed packet distribution with centralized multicast (referred to as SRM/detailed) and abstract multicast distribution (referred to as SRM/abstract). Session participants are randomly selected on a 100-node transit-stub topology. To study scaling, we increase the session size from 20 to 90 and record the time and memory consumption. To study possible behavior differences, we measure the common SRM metrics - recovery delay, duplicate repair, and duplicate request. To make sure our results are reproducible, we examine 10 runs at each session size, randomly distributing the session participants in the same 100 node network.

Figure 7 shows that abstraction improves memory and CPU usage for SRM simulations. Session SRM simulations experience a slower slope increase in memory consumption and a constantly lower time consumption when session size grows.

To evaluate if abstraction changed the simulation results, we look at the three common metrics of reliable multicast mechanisms. In Figure 8, we plot three common metrics of SRM performance. For each metric, we plot the mean observed value for detailed and abstract simulations and indicate the 95% confidence interval around this mean. As can be seen, these means are very close to each other suggesting that abstraction produces about the same results.

SRM is an error recovery scheme and so most simulations do not concern operationally congested networks. This suggests that the abstract multicast distribution works well for SRM simulations, and indeed our simulations field useful results. However, there are other network mechanisms (e.g., congestion control) that clearly need to be studied in the presence of congestion. Can these mechanisms be studied in abstract packet distribution? The answer may be yes, depending on the levels of details the mechanism requires and the availability of the models for congested links and nodes (e.g., input rate, output rate, => queuing time). As mentioned earlier, to scale simulations is application specific. Currently, we are working on hybrid simulation techniques to model queuing behaviors.

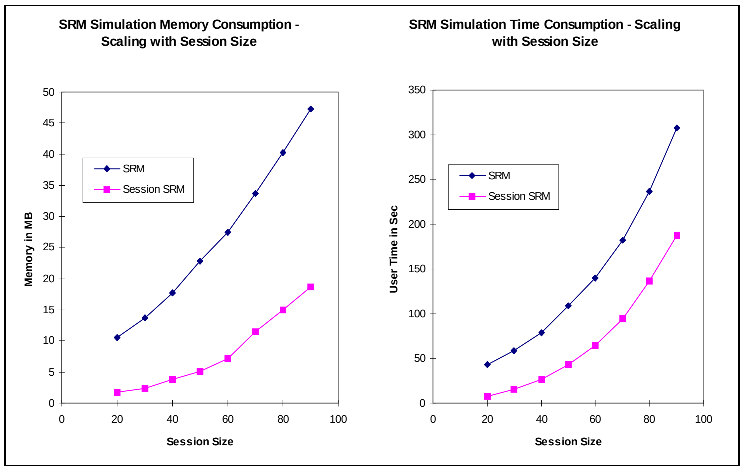


Figure 7. Performance Comparison for SRM and Session SRM

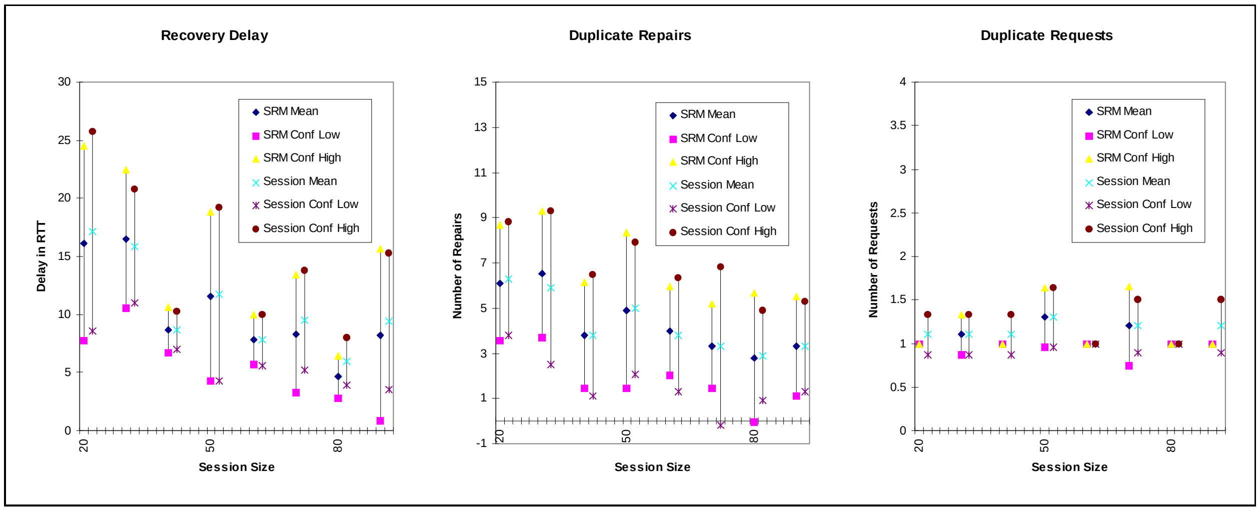


Figure 8. Accuracy Comparison for SRM and Session SRM

6. Future Work

Our short term objective is to apply our abstraction techniques on other multicast transport protocols such as MFTP (Multicast File Transfer Protocol) and RMTP (Reliable Multicast Transport Protocol). Then, we would like to extend our delay model to include queuing characteristics so that we could apply abstract multicast distribution on congested network scenarios. For the longer term, we would like to further expand the scalability of ns-2 to perform simulations on the order of tens of thousands.

7. Conclusion

We found that the general purpose dense mode multicast with detailed packet distribution is not scalable with increased topology size, group size, and number of groups. Therefore, we applied the centralized computation technique on multicast routing. The resulted centralized multicast avoids periodic flood and prune but exhibits difference in convergence latency. Most higher level multicast protocol simulations do not need the transient details, so using centralized multicast does not affect the end-results. However, if a particular set of simulations requires the transient details, we suggest to run these simulations at the detailed mode. Although centralized multicast scales well with increased group size and number of groups, it does not seem to scale as well with increased topology size. Therefore, we apply the abstract packet distribution technique on multicast packet distribution. The resulted abstract multicast distribution significantly reduces link and node structures, and the amount of packet scheduling, but exhibits difference in end-end delay when the simulated network is congested. Many multicast error recovery mechanisms do not require simulation on congested networks, so using abstract multicast distribution does not affect the end-results. In the case study, we use SRM as an example to demonstrate that our abstraction techniques conserve resources and retain accuracy. Currently, the delay model for the abstract multicast distribution has not been extended to model congested network yet, so mechanisms such as multicast congestion control should avoid the simple abstract multicast distribution. However, we believe that the queuing delay can be modelled and included into abstract multicast distribution for simulating congested network in a foreseeable future. These abstraction techniques will improve the way the research community studies protocols.

References

1. Roger McHaney, Computer Simulation A Practical Perspective, Academic Press Inc., 1991
2. Bruno R. Preiss, Ian D. MacIntyre, and Wayne M. Loucks, On the Trade-off between Time and Space in Optimistic Parallel Discrete-Event Simulation, In Proceedings of. 1992 Workshop on Parallel and Distributed Simulation, pages 33-42, Newport Beach, CA, January 1992. Society for Computer Simulation

3. James F. Ohi and Bruno Richard Preiss, Parallel Instance Discrete-Event Simulation Using a Vector Uniprocessor, In Proceedings of. 1991 Winter Simulation Conference., pages 93-601, Phoenix, AZ, December 1991. Society for Computer Simulation
4. Jayadev Misra, Distributed Discrete-Event Simulation, ACM Computing Surveys, Vol. 18 No. 1, March 1986, pp. 39-65
5. David R. Jefferson, Virtual Time, ACM Transactions on Programming Languages and Systems, Vol. 3 No. 40, July 1985, pp. 404-425
6. K. M. Chandy and Jayadev Misra, Asynchronous Distributed Simulation via a Sequence of Parallel Computations, Communications of the ACM, Vol 24 No. 11, April 1981, pp. 198-205
7. Rajie L. Bagrodia and Wen-Toh Liao, Maisie: A Language for the Design of Efficient Discrete-Event Simulations, IEEE Transactions of Software Engineering, Vol. 20 No. 4, April 1994, pp. 225-238
8. Scalable Self-Organizing Simulation (S3), Web Page http://www.dimacs.rutgers.edu/Projects/Simulations/darpa/
9. DARPA Global Mobile (GloMo) Information Systems program, Web Page http://glomo.sri.com/glomo/
10. The REAL Network Simulator, Web Page http://www.cs.cornell.edu/skeshav/real/overview.html
11. Jong-Suk Ahn and Peter B. Danzig, Speedup vs. Simulation Granularity, Technical Report, USC CS TR92-528, University of Southern California
12. D. Schwetman, Hybrid Simulation Models of Computer Systems, Communication of the ACM, September 1978
13. D. Waitzman, S. Deering, and C. Partridge, Distance Vector Multicast Routing Protocol, RFC1075, November 1988
14. Estrin, D. Farinacci, A. Helmy, V. Jacobson, and L. Wei, Protocol Independent Multicast - Dense Mode (PIM-DM): Protocol Specification, Proposed Experimental RFC, September 1996
15. A. J. Ballardie, P. F. Francis, and J. Crowcroft, Core Based Trees, In Proceedings of the ACM SIGCOMM, San Francisco, 1993
16. S. Deering, D. Estrin, D. Farinacci, M. Handley, A. Helmy, V. Jacobson, C. Liu, P. Sharma, D. Thaler, and L. Wei, Protocol Independent Multicast - Sparse Mode (PIM-SM): Motivation and Architecture, Proposed Experimental RFC, September 1996
17. S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C. Liu, and L. Wei, An Architecture for Wide-Area Multicast Routing. In Proceedings of the ACM SIGCOMM, London 1994
18. S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C. Liu, and L. Wei, The PIM Architecture for Wide-Area Multicast Routing. ACM Transactions on Networks, April 1994
19. D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, M. Handley, V. Jacobson, C. Liu, P. Sharma, and L. Wei, Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification, Proposed Experimental RFC, September 1996
20. S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C. Liu, L. Wei, P. Sharma, and A. Helmy, Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification, Internet Draft, December 1995

21. Deborah Estrin, Mark Handley, Ahmed Helmy, Polly Huang, David Thaler, A Dynamic Bootstrap Mechanism for Rendezvous-based Multicast Routing, Technical Report USC CS TR97-644, University of Southern California, 1997
22. J. Moy, MOSPF: Analysis and Experience, RFC1585, March 1994
23. J. Moy, Multicast Extensions to OSPF, RFC1584, March 1994
24. E. Zegura, K. Calvert, and M. Donahoo, A Quantitative Comparison of Graph-based Models for Internet Topology, To Appear in Transactions on Networking, 1997
25. Steven McCanne and Sally Floyd, UCB/LBNL/VINT Network Simulator - ns (version 2), http://www-mash.CS.Berkeley.EDU/ns/
26. Sally Floyd, Van Jacobson, Steven McCanne, Ching-Gung Liu, and Lixia Zhang, A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing, Extended Report, LBNL Technical Report, pages 1-37, September 1995, also submitted for publication in IEEE/ACM Transactions on Networking.
27. S. Armstrong, A Freier, and K. Marzullo, Multicast Transport Protocol, RFC1301, February 1992
28. John C. Lin and Sanjoy Paul, RMTP: A Reliable Multicast Transport Protocol, Proceedings of IEEE INFOCOM ‘96, pp. 1414-1424, April 1996
29. R. Yavatkar, J. Griffioen, and M. Sudan, A Reliable Dissemination Protocol for Interactive Collaborative Applications, Proceedings of ACM Multimedia ‘95, 1995
30. D. Mills, Network Time Protocol (v3), RFC 1305, April 1992, Obsoletes RFC 1119
31. Ching-Gung Liu, A Scalable Reliable Multicast Protocol, PhD. Dissertation Proposal, September 1996
32. Ching-Gung Liu, Deborah Estrin, Scott Shenker, and Lixia Zhang, Local Error Recovery in SRM: Comparison of Two Approaches, Submitted for Publication in IEEE/ACM Transactions on Networking, February 1997
33. Puneet Sharma, Deborah Estrin, Sally Floyd, and Lixia Zhang, Scalable Session Messages in SRM, Submitted for Publication, http://netweb.usc.edu/vint/papers/ssession.ps, University of Southern California, August, 1997
34. Kannan Varadhan, Deborah Estrin, and Sally Floyd, Impact of Network Dynamics on End-to-End Protocols: Case Studies in Reliable Multicast, Submitted for Review to the Third IEEE symposium on Computers and Communications, http://netweb.usc.edu/vint/papers/dynamics.ps, University of Southern California, August, 1997
35. D. Thaler, D. Estrin, and D. Meyer, Border Gateway Multicast Protocol (BGMP): Protocol Specification, Internet Draft, IDMR Working Group, draft-ietf-idmr-gum-01.txt, October, 1997