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
{huang, estrin, johnh}@isi.edu

Abstract

Due to the complexity and scale of the current Internet, large-scale simulation is 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. We investigate a complimentary solution -- simulation abstraction. Just as a custom simulator includes only details necessary for the task at hand, a general simulator can support configurable levels of detail for different simulations. We demonstrate two abstraction techniques in multicast simulations and show that they each help to gain one order of magnitude in performance. Although abstraction simulations are not identical to more detailed simulations, in many cases these differences are small and result in minimal changes in the conclusions drawn from simulations.

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 [11]) 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, and local agents. 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. Ahn and Danzig estimated 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." [3]

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 that is not widely available or expensive.

A complimentary solution is to slim down simulations by abstracting out details. The idea is to analyze simulations, identify the bottleneck, and eliminate it by abstracting unnecessary details. The risk of abstraction is that simulation results may be distorted; 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 through the general guidelines provided, it can then be used for larger simulations.

Because multicast protocols potentially have more complicated scaling problems, we apply our abstraction techniques, centralized computation and abstraction packet distribution, on multicast routing and packet transmission 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 enables protocol evaluation for large-scale scenarios.

After discussing related work, we present the techniques used to abstract details, particularly for multicast simulations, and the general guidelines 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 guidelines and the abstraction techniques.

2. Related work

There has been a great deal of work on network simulation. However, most of the simulators focus on providing protocol modules and user-friendly interface. Only few address the scaling problem. Here we briefly summarize priori work in parallel and distributed simulation, abstraction, and hybrid simulation.

Parallel and distributed simulation 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 synchronization mechanisms to maintain the correct event ordering [1,2]. Consequently, the simulation efficiency may be degraded due to the overhead associated with these mechanisms. Although parallel and distributed simulation is useful when large computers are available, alternative techniques such as abstraction are also needed to make very large simulations.

Ahn and Danzig [3] proposed to abstract packet streams (Flowsim) for packet network simulations and proved that Flowsim can 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 techniques for a wider range of network protocols.

In hybrid simulation models [4], 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 whereas the analytical queuing model is used for the scheduling 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 and memory consumption. Through abstraction we reduce resource consumption, enabling large-scale simulations. Abstraction is possible because Internet protocols are layered. In evaluation of a level-n protocol, we need information provided by level n-1, 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 message exchange that is required to generate the routing tables. If we abstract away unnecessary details, the memory and time consumption can be conserved and used to simulate larger-scale scenarios. Here we briefly introduce two abstraction techniques using multicast simulations as examples and the general guidelines to abstract. Implementation details of the abstraction techniques and their effect on simulation performance and distortion are presented in the next section.

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 [5] and PIM-DM [6]), membership advertisement (e.g., MOSPF [9]), and rendezvous-based (e.g., CBT [7] and PIM-SM [8]). Much of the current IP multicast infrastructure uses broadcast and prune protocols, so we use a DVMRP-like implementation as the standard detailed multicast for the comparison. The broadcast and prune multicast protocols are suitable for groups with dense member distribution, and, thus, also called dense mode multicast protocols.

Our detailed implementation of 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 computation, avoids this network-level message exchange by centrally computing protocol states. Centralized multicast, an example of applying the centralized computation on multicast routing, computes the result of routing message exchange and updates routes in multicast trees instantly. The same approach can be applied to other network-layer protocols (e.g., unicast routing). The centralized computation technique conserves a significant amount of memory and time at the cost of a slight difference in control message overhead, and route convergence latency when group membership or topology changes.

Centralized 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 avoids transport-level node by node, link by link packet transmission by direct packet scheduling at the receiving ends. Consequently, nodes and links become lightweight data structures. Session multicast, an example of applying abstract packet distribution on multicast transmission, sets up a direct connection (i.e., a session) between a source and its receivers, computes loss and delay characteristics for each source and receiver pair, and schedules packets receive events accordingly. The same technique can be applied to other transport-layer protocols (e.g., TCP). 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.

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 and measurement metrics
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 one of the following conditions holds, proceed with large simulations using the abstract version.
    - The defined measurement metrics are not affected by the introduced distortion at all.
    - Distortion will not be enlarged when scaling factors grow.

We demonstrate the above general guidelines in the case study.

4. Systematic Comparison

We start this section by providing details of the original dense mode multicast, centralized multicast, and session multicast implementations in ns-2. After presenting the simulation resource consumption by the three implementations, finding that the session multicast scales the best and centralized multicast the second, we illustrate the distortions introduced by centralized multicast and session multicast.

4.1 Mechanism Detail

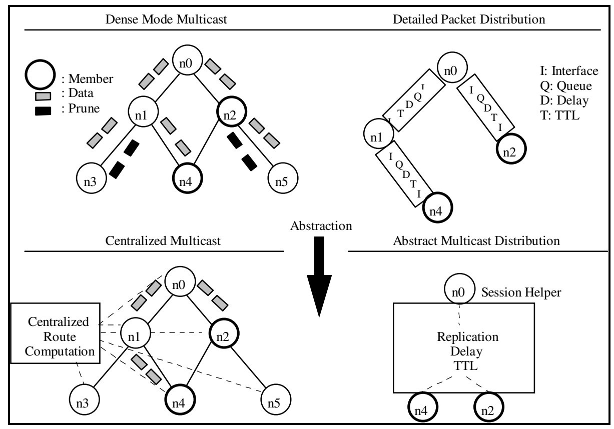

Figure 1. Illustration of the Detailed and Abstract Simulation Mechanisms

Dense Mode Multicast & Detailed Packet Distribution. The ns-2 implementation of dense mode multicast closely follows real-world implementations in terms of message exchanges (Figure 1, upper left). Each dense mode multicast agent maintains a parent-child relationship to all other agents by a triggered checking to neighbors' routing tables. When a node does not find any forwarding entry for a packet, the node upcalls its local dense mode agent to install a multicast forwarding entry according to the source and group addresses in the packet header. The dense mode multicast agent inserts only the child links indicated in its parent-child relationship as the outgoing interfaces. Packets are then forwarded to the outgoing interfaces.

When a 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 onto 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 (Figure1, upper right). 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 multicast address (mask and mask length), packets are forwarded to a multicast classifier which maintains a hash table of routing entries (source, destination, outgoing target, incoming interface). These entries are installed by the multicast routing protocol. The classifiers then hash on the source, destination, or incoming interface, to decide which next object to forward the packets. In unicast classifiers, the next object is typically the network interface. However, in multicast classifiers, next object is a replicator. The replicator copies the packets and then forwards 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 (time to live) 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.

Centralized Multicast. The detailed protocol implementations are helpful in validating the detail protocol design, but they often are unnecessarily specific. Our centralized multicast abstraction eliminates much of its 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 each source 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 new 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, and other message exchange in the dense mode multicast are eliminated in centralized multicast. As a result, some detailed characteristics may not be captured.

Session Multicast. Another area with possibly unnecessary detail is hop by hop packet transmission and heavyweight link and node structures. Session multicast reduces these overheads.

Session multicast 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.

It is important that the session multicast and detailed multicast transmission share the same loss patterns. In particular, losses on a link will correlate to losses to any recipients downstream of that link. Session multicast retains this dependency. To illustrate, 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 in session multicast, where n6 is dependent on e2, n3 is dependent on e1, and n1, n4 and e1 are dependent on e0.

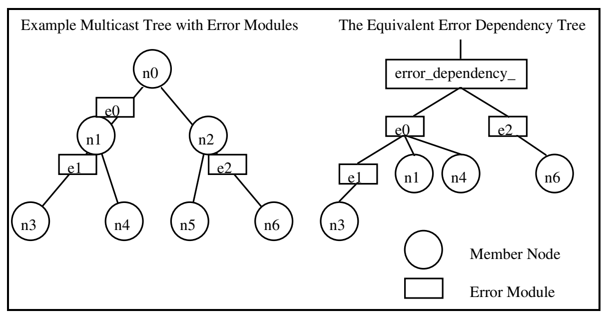

Figure 2. An Example of Error Dependency in Abstract Multicast Distribution (Session Multicast)

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 objects) is reduced to numbers, delay and bandwidth; the original node structure, a combination of classifiers, is reduce to two numbers, node id and port id. Our simple delay calculation entirely ignores queuing delay and therefore session multicast 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. Three sets of tests explore each of the three dimensions from a standard case, 300 nodes (average degree 1.77), 30 groups, and 30 members. We use 2-level hierarchical network topologies with a constant degree of connectivity (~1.77), created by GT-ITM (Georgia Tech - Internetwork Topology Model) [10]. Sources and members are randomly chosen for all simulations. Memory and running time are measured to compare the performance among the three versions of simulations. All simulations are conducted on a FreeBSD Pentium PC with 128 MB real memory.

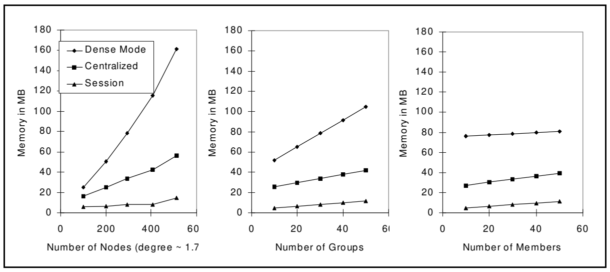

Figure 3. Comparison of Memory Usage

Memory Consumption. From all three graphs in Figure 3, we observe the dramatic effect of periodic flood and prune in detailed multicast on the memory consumption. 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. However, centralized multicast still experiences a significant growth along with the topology size because of heavyweight links and nodes. Session multicast scales the better than the other two.

One interesting point observed from the right most graph is that three simulations grow at similar rates. This is because of the random selection of members. If the member distribution is diverse (fairly random), 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.

In summary, we show that abstraction can save substantial amounts of memory, allowing larger simulations. 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 changes.

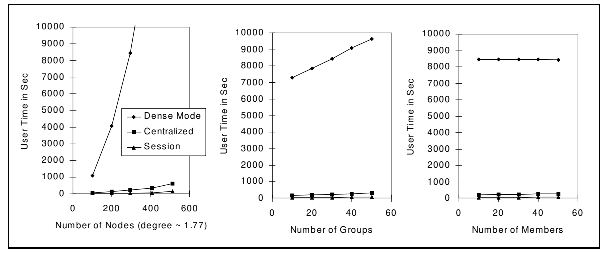

Figure 4. Comparison of Time Usage

Time Consumption. Dense mode scales even worse in CPU time, looking at all three graphs in Figure 4, due to the flood and prune over the entire topology. In particular, the drastic growth of detailed multicast scaling number of nodes is also contributed by the extra CPU cycles spent for swapping after the memory consumption exceed the available real memory. 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. Distortion

The performance gains are possible because abstractions remove some protocol details, consequently introducing distortions. Users must follow the general guidelines (Section 3) and understand if the distortions associated with the applied abstraction techniques will affect the defined measurement metrics. In this section we characterize the distortions introduced by centralized multicast and session multicast.

Distortion by Centralized Multicast. The centralized multicast replaces routing message exchange with a centralized computation agent. Although the resulting routes by the two implementations are identical, their transient behavior can be different. In detailed dense mode multicast, messages propagate topology changes through the tree at link speeds, while with centralized abstraction, topology changes are globally known the instant they occur.

To demonstrate the difference, we simulate various group events (join, leave, link down, link up) on a sample topology. We periodically count the numbers of data packets on the entire topology and draw Figure 5, we referred to as the behavior chart. This behavior chart roughly represents the accuracy of the simulations.

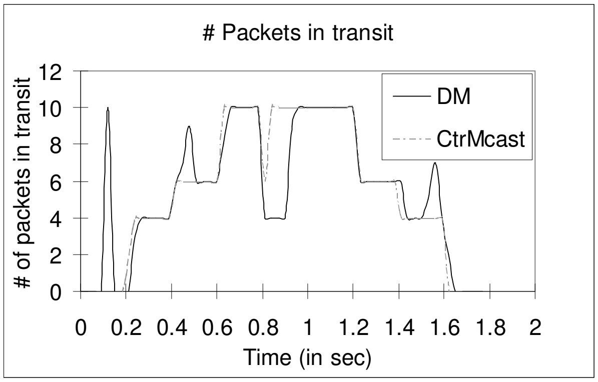

Figure 5. Difference Between Dense Mode and Centralized Multicast

The dark line represents dense mode multicast behavior. The light dashed 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. We find that steady states of the lines rise by a few packets when members, not currently in the multicast tree, join the group (e.g., at time 0.2, 0.4, and 0.6), and vice versa. Furthermore, when a link on the tree fails (for example, at 0.8), we find that the lines drop a few packets lower and, when new routes converge, 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 differs depending on the update sensitivity of the two implementations.

Concluding from the above observations, 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 end results when control message overhead are transient delay are not critical issues.

Distortion by Session Multicast. Session multicast replaces hop-by-hop message passing with direct end-to-end delays, which ignore 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 model queuing delays when cross traffic is added. To demonstrate this effect we compare end-to-end delay in abstract and detailed simulations and draw the mean difference with 1 to 80 multicast groups (roughly represents degree of congestion). See Figure 6.

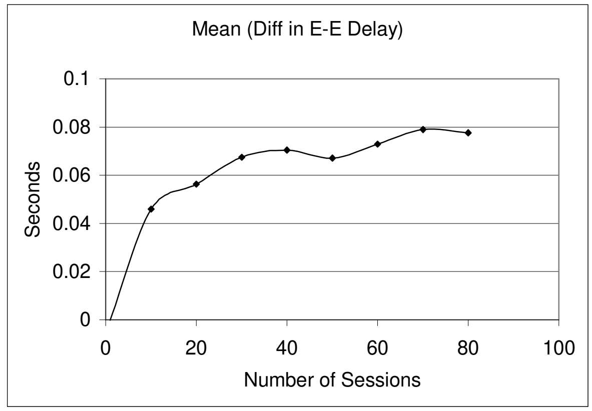

Figure 6. Difference Between Detailed Packet Distribution and Abstract Multicast Packet Distribution

When there is only one multicast flow, 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, we see almost no difference in end-to-end delay. However, when we increase the number of multicast flows, the network starts to experience 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

Abstraction techniques improve performance, but in some cases it produces different results. It is important that users follow the general guidelines to create efficient and accurate abstract, large-scale simulations. To demonstrate, we examine the use of the general guidelines on an important, active research area, reliable multicast. In particular, we choose Scalable Reliable Multicast (SRM) [12], one of the pioneer research in reliable multicast.

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. A timer based on the distance between members is used to suppress duplicate requests and recoveries. Therefore, the time to recover and amount of duplicate messages are important measurement metrics to SRM experiments. Scaling factor is group size.

5.2. SRM Simulations

We profile SRM simulations and find one of the bottleneck is hop-by-hop packet transmission. Thus, we apply session multicast on the SRM simulations and perform the following set of experiments. Members are randomly selected on a 100-node 2-level hierarchical topology. The time and memory consumption is measured to observe the improvement. Recovery delay, duplicate repair, and duplicate request are measured to compare the simulation results.

Figure 7 shows that abstraction improves memory and CPU time consumption. The abstract SRM simulations experience a slower increase in memory consumption but similar increase in time consumption when group size grows. In Figure 8, we plot the 95% confidence interval for each of the measurement metrics and place the results of detailed and abstract simulations side by side. As we can see, these intervals are very close to each other suggesting that abstract simulations produce about the same results.

SRM is an error recovery scheme, so simulations do not concern operationally congested networks. Hence, the session multicast work well for SRM simulations and will continue to work for larger-scale simulations as long as there is no congestion in the simulated network. However, there are other protocols (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 mixed mode technique that will allow the critical areas to run in detailed mode while the rest in abstract mode. With minor performance degrade, we will be able to improve accuracy in simulation results.

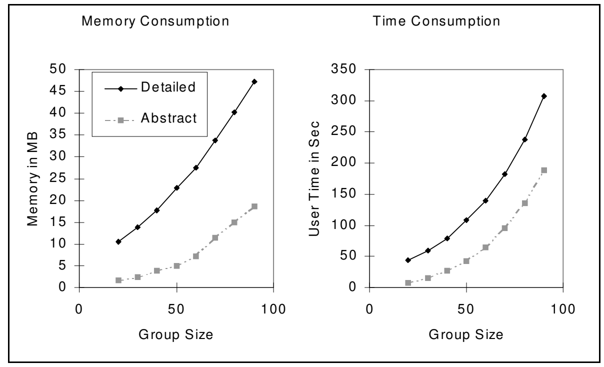

Figure 7. Performance Comparison for SRM and Session SRM

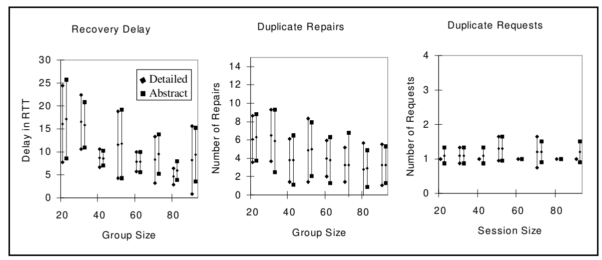

Figure 8. Accuracy Comparison for SRM and Session SRM

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) [13] and RLM (Receiver-Driven Layered Multicast) [14]. Then, we would like to experiment our mixed mode technique to include queuing characteristics so that we could apply abstract multicast distribution on congested network scenarios. For longer term, we would like to further expand simulation scale to 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 control traffic overhead and convergence latency. Higher level multicast protocol simulations do not necessarily need the transient details, so in many cases using centralized multicast does not affect the end-results. However, if a particular set of simulations requires the transient details, we suggest run these simulations at the detailed mode and avoid centralized multicast.

Although centralized multicast scales well with increased group size and number of groups, it does not scale as well with increased topology size. Therefore, we apply the abstract packet distribution technique on multicast packet distribution. The resulted session multicast 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. Session multicast ignores queuing delays, so mechanisms such as multicast congestion control should avoid the abstract packet distribution. However, we are working a mixed mode simulation technique that will allow simulations for congested networks. These abstraction techniques will improve the scale the research community studies protocols.

Acknowledgements

The authors would like to thank Jon Postel, Joseph Bannister (ISI), Sally Floyd (LBNL), and the anonymous reviewers for their useful comments and insightful feedback on the paper.

References

[1] David R. Jefferson. Virtual Time. *ACM Transactions on Programming Languages and Systems*, 3(40):404-425, July 1985

[2] K. M. Chandy and J. Misra. Asynchronous Distributed Simulation via a Sequence of Parallel Computations. *Communications of the ACM*, 24(11):198-205, April 1981

[3] J. Ahn and P. B. Danzig. Speedup vs. Simulation Granularity. *IEEE/ACM Transaction on Networking*, 4(5):743-757, October 1996

[4] D. Schwetman. Hybrid Simulation Models of Computer Systems. *Communication of the ACM*, September 1978

[5] D. Waitzman, S. Deering, and C. Partridge. Distance Vector Multicast Routing Protocol. *RFC1075*, November 1988

[6] D Estrin. D. Farinacci, A. Helmy, V. Jacobson, and L. Wei. Protocol Independent Multicast - Dense Mode (PIM-DM): Protocol Specification. *Proposed Experimental RFC*, September 1996

[7] A. J. Ballardie, P. F. Francis, and J. Crowcroft. Core Based Trees. *In Proceedings of the ACM SIGCOMM*, San Francisco, 1993

[8] 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

[9] J. Moy. Multicast Extensions to OSPF. *RFC1584*, March 1994

[10] E. Zegura, K. Calvert, and M. Donahoo. A Quantitative Comparison of Graph-based Models for Internet Topology. *To Appear in IEEE/ACM Transactions on Networking*, 1997

[11] S. McCanne and S. Floyd. UCB/LBNL/VINT Network Simulator - ns (version 2). http://www-mash.CS.Berkeley.EDU/ns/

[12] S. Floyd, V. Jacobson, S. McCanne, C. Liu, and L. Zhang. A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing. To Appear in IEEE/ACM Transactions on Networking, 1997

[13] StarBurst Communication Corporation. StarBurst MFTP: An Efficient, Scalable Method for Distributing Information Using IP Multicast. http://www.starburstcom.com/while.htm

[14] S. McCanne, V. Jacobson, and M. Vetterli. Receiver-Driven Layered Multicast. ACM SIGCOMM, Stanford CA, August 1996