Proceedings of the 2003 Winter Simulation Conference New Orleans, LA, December 2003, pp. 712-720
S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, eds.

LOOSELY-COORDINATED, DISTRIBUTED, PACKET-LEVEL SIMULATION OF LARGE-SCALE NETWORKS

Boleslaw K. Szymanski
Yu Liu

Department of Computer Science
Rensselaer Polytechnic Institute
Troy, NY 12180, U.S.A.

ABSTRACT

The complexity and dynamics of the Internet is driving the demand for scalable and efficient network simulation. In this paper, we describe a novel approach that partitions the networks into domains and simulation time into intervals. Each domain is simulated independently of and concurrently with the others with only local domain information over the same simulated time interval. At the end of each interval, global routing information, packet delays and drop rates for each inter-domain flow are exchanged between domain simulators. When the exchanged information converges to the value within a prescribed precision all simulators progress to the next simulated time interval. This approach allows the parallelization with infrequent synchronization, and achieves significant simulation speedups. Such a solution supports simulations of large-scale networks on distributed machines with modest memory size.

1 INTRODUCTION

In simulating large-scale networks at the packet level, a major difficulty is the enormous computational power needed to execute all events that packets undergo in the network (Law and McComas 1994). Conventional simulation techniques require tight synchronization for each individual event that crosses the processor boundary (Bhatt et al. 1998). The inherent characteristics of network simulations are the fine granularity of events (individual packet transitions in a network) and high frequency of events that cross the boundaries of parallel simulations. These two factors severely limit parallel efficiency of the network simulation executed under the traditional protocols (Bhatt et al. 1998).

Another difficulty is the large memory size required by large-scale network simulations. With the current trend of simulating ever larger and more complicated networks, the memory size becomes a bottleneck. Centralized network configuration and routing information results in large memory requirements during construction of the simulated network. Additionally, the needed memory increases also with the intensity of traffic flows that dictate the size of the future event list. Although memory requirements can be tampered by the good design and implementation of the simulation software (Nicol 2002), we believe that to simulate truly large networks, the comprehensive, distributed memory approach is needed.

This paper describes our long-term research on developing an architecture that can efficiently simulate very large heterogeneous networks in near real time (Szymanski et al. 1999). Our approach combines simulation and modeling in a single execution. A novel coarse granularity synchronization mechanism is used to achieve better parallel efficiency. Thanks to this approach, Genesis is able to use different simulators in a single coherent network simulation. This feature motivated the name of the system: General Network Simulation Integration System, or *Genesis* in short.

Genesis addresses also the large memory requirement problem in large-scale network simulations. Many parallel simulation systems achieved speed-up in simulation time, however, they also required that every machine involved had big enough memory to hold the full network. This requirement is most easily achieved through a system with shared memory. In Genesis, in contrast, memory usage is fully distributed among participating simulators.

As discussed in (Szymanski et al. 2002), the approach underlying Genesis can also be seen as a variant of a general scheme for optimistic simulation referred to as Time-Space Mappings proposed by Chandy and Sherman in (Chandy and Sherman 1989). Although all optimistic simulations can be viewed as variants of this scheme, very few apply, as we do, iterations over the same time interval to find a solution.

2 GENESIS APPROACH

2.1 Coarse Granularity Synchronization in Genesis

Genesis uses a coarse granularity synchronization mechanism to simulate network traffics, as described below. In Genesis, a large network is decomposed into parts and each part is simulated independently and simultaneously with the others. Each part represents a subnet or a domain or even Autonomous System (AS) of the entire network. These parts are connected to each other through edges that represent communication links existing in the simulated network. In addition, we partition the total simulation time into separate simulation time intervals selected adaptively in such a way that the traffic characteristics change little during each time interval.

Each domain is simulated by a separate simulator which has a full description of the flows whose sources are within its domain. Each simulator needs to approximate flows routed to or through its domain whose sources are external to the domain. To this end, each simulator creates its *domain closure* that includes all the sources of flows that reach or pass through this domain. Since these are copies of nodes active in other domains, we call them *proxy sources*. Each proxy source uses the flow definition from the simulation configuration file.

The flow delay and the packet drop rate experienced by the flows outside the domain are simulated by the random delay and probabilistic loss applied to each packet traversing in-link proxy. These values are generated according to the average packet delay and its variance as well as observed packet loss frequency communicated to the simulator by its peers at the end of simulation of each time interval. Each simulator collects this data for all of its own out-link proxies when packets reach the destination proxy.

Every domain simulator stops its simulation at predefined checkpoints, and exchanges data with all the other domain simulators by exchanging data with others. Each domain simulator checks its convergence condition by analyzing the received data, based on some pre-defined metrics (end-to-end packet delay, packet loss rate, etc.) and parameters (e.g., precision threshold). Until the convergence condition is not satisfied, the domain simulator will be going back to the last checkpoint and re-simulating the last time interval, utilizing the data received during the latest checkpoint. When all the domain simulators converge, a global convergence is reached, and all the domain simulators go on to the next time interval. The system framework is shown in Figure 1.

Consider a flow from an external source P to the internal destination Q, passing through a sequence of external routers $r_1, \ldots r_n$ and internal routers $r_{n+1}, \ldots r_k$. The source of the flow can be represented by the sequence of pairs $(t_1, p_1), \ldots (t_m, p_m)$, where t_i denotes the time of departure of packet i and p_i denotes its size. At router i, a packet j is either dropped, or passes with the delay $d_{i,j}$. For uniformity, dropping can be represented as as delay T greater than the total simulation time. Hence, to replicate a flow with the proxy source Q' sending packets to router r_{n+1}, packet j produced by Q' at time t_j needs to be delayed by time $D_j = \sum_{i=1}^{n} d_{i,j}$. A delay at each router is the sum of constant processing, transmission and propagation delays and a variable queuing delay. If the total delay over all external routers is relatively constant in the selected time interval, a random delay with proper average and variance approximates D_j well. Thanks to the aggregated effect of many flows on queue sizes, this delay changes slower than the traffic itself, making such model precise enough for our applications.

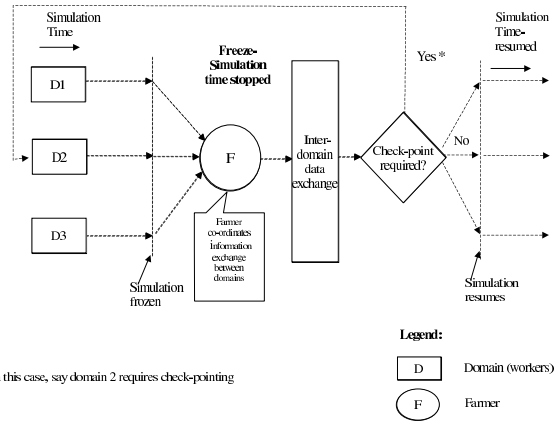

Figure 1: Progress of the Simulation Execution

2.2 Efficiency Analysis

It has been observed that the execution time of a network simulation grows faster than linearly with the size of the network (Ye et al. 2001). Theoretical analysis supports this observation because for the network size of order $O(n)$, the simulation time contains terms which are (i) of order $O(n * \log(n))$, that correspond to sorting event queue, (ii) of order $O(n^2)$, that result from packet routing, and (iii) even of order $O(n^3)$, that represent the cost of building routing tables. Therefore, it is possible to speed up the network simulation more than linearly by splitting a large simulation into smaller pieces and parallelizing the execution of these pieces.

The challenge in large scale network simulations

is that the processing of large amount of events requires enormous computational power and long execution time. Conventional packet-level parallel simulations achieved execution speedups with modest number of parallel processors, but their scalability and parallel efficiency were limited by their frequent event-level synchronization. To achieve execution speedups, fluid model network simulation took the approach of using a high level abstraction of network traffics to reduce the number of simulation events (Liu et al. 2001). But the efficiency and accuracy of simulating fluid models highly depend on the types, parameters and complexities of network models. Hence, such models can sometimes be less efficient than packet-level simulations (Liu et al. 1999), and it is difficult for them to achieve high accuracy in complex wireless network models. Genesis, in contrast, took another approach and uses a novel coarse granularity synchronization mechanism to reduce the frequency of synchronization while preserved packet-level simulation. It achieved better parallel efficiency than conventional parallel simulations with controllable accuracy. Figure 2 shows the comparison among these approaches.

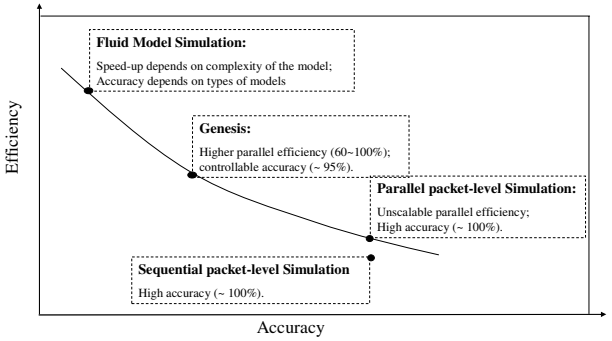

Figure 2: Simulation Efficiency vs. Accuracy

Our target application is network management based on on-line network monitoring and on-line simulation (Ye et al. 2001). The presented method fits very well such an application as it predicts changes in the network performance caused by tuning of the network parameters.

2.3 Event-level Synchronization in Genesis

The basic Genesis approach described above was designed to simulate TCP and UDP data traffics, but could not be used to simulate some other flows, for example, data flows providing information for routing protocols. This is because the traffic of a routing protocol cannot be summarized on packet delay and drop rate; instead, different content and timing of each routing packet might change the network status. Particularly, our desire to simulate BGP protocol required us to develop additional synchronization mechanism in Genesis. We developed an event-level synchronization mechanism which can work within the framework of Genesis and support the simulation of BGP.

2.4 Memory Distribution

Simulations of large-scale networks require large memory size. This requirement can become a bottleneck of scalability when the size or the complexity of the network increases. For example, ns2 uses centralized memory during simulation, which makes it susceptible to the memory size limitation. The scalability of different network simulators was studied in (Nicol 2002). This paper reports that in a simulation of a network of a dumbbell topology with large number of connections, ns2 failed to simulate more than 10000 connections. The failure was caused by ns2's attempt to use virtual memory when swapping was turned off. This particular problem can be solved by using machines with larger dedicated or shared memory. Yet, we believe that the only permanent solution to the simulation memory bottleneck is to develop the distributed memory approach.

In a typical parallel network simulation using non-distributed memory, each of the parallel simulators has to construct the full network and to store all dynamic information (e.g., routing information) for the whole network during the simulation. To avoid such replication of memory, we developed an approach that completely distributes network information. Thanks to this solution, Genesis is able to simulate large networks using a cluster of computers with smaller dedicated memory (compared to the memory size required by shared memory-based SSFNet simulating the same network), as shown in section 4.2.

3 GENESIS DESIGN OVERVIEW

3.1 Design of Domain Simulator Model

In this section, we summarize the basic design of Genesis domain simulator to support domain-based parallel simulations presented by us in (Szymanski et al. 2002).

The user is responsible only for annotating domains in the simulation configuration file. This is achieved simply by labeling each node in the configuration by the corresponding domain number. Based on these annotations, the extensions to the ns system process domain definition and its closure, collect the data for information exchange and implement the information exchange, as well as monitor convergence. A sample domain and its closure is presented in Figure 3 and discussed below.

Support for domain definition in Genesis, i.e., identifying which nodes belong to a particular domain, is

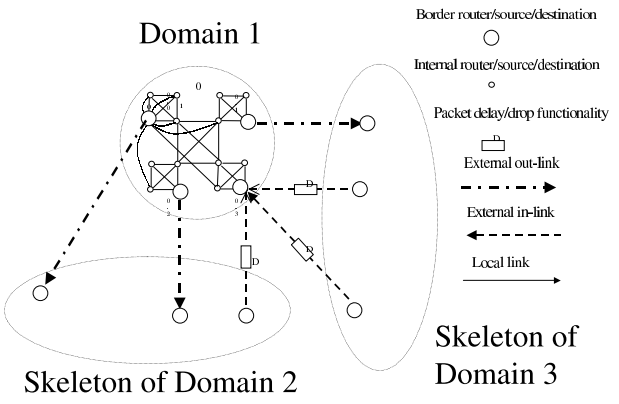

Figure 3: Domain Simulator Structure

the first step towards creating the domain closure. By definition, in the domain closure, each external proxy source is directly connected to the destination domain of its flow. We refer to such replicated source as an *proxy source* and we call the link that connects it to the domain border router an *in-link proxy*.

The design supports the selective activation and deactivation of domains. The purpose is to process entire simulation configuration on each participating processor, but then, during simulation, to keep active only one domain closure while maintaining the routing information for the entire simulation. This information is needed to identify the destination domains for all packets that leave the domain.

Consider the sample network in Figure 3. The network is split into three individual domains, numbered 1, 2 and 3. Packets that flow into the domain 1 from outside (with sources in skeletons of domains 2 and 3 in Figure 3) are produced by their proxy sources in the domain closure and delayed or dropped during transition through in-link proxies (marked by boxes in Figure 3).

Exchange of data uses the Farmer-Worker architecture, in which one processor collects the data from all the others and redirects them to all the simulators. Recording the information needed for data exchange involves calculating, for each packet leaving the domain, the time expired from the instance a packet leaves its source to the time it reaches the destination proxy. Also recorded is information about each packet source and its intended external node destination as well as whether the packet was dropped by a router inside the domain.

The following functionalities were also implemented in Genesis:

The ability to suspend the simulation to enable exchange of data on path delays using message passing between processors simulating individual domains. During the simulation freeze, each individual simulation domain exchanges information on packets generated and dropped along links leaving the domain (cf. Figure 3).

The ability to record information about the delays and drop rate experienced by the packets leaving the domain. Each delay measures the time expired from the instance a packet leaves its source to the time it reaches the domain boundary. Packet drop rates are computed for each flow separately. Also recorded is information about each packet source and its intended destination. Having this information enables us to replicate the source from the original domain to the boundary of the target domain (sources in skeletons of domains 2 and 3 in Figure 3) and postpone an arrival of each packet produced by the replicated source at the domain boundary by the delay measured in the source (and transient, if necessary) domains. Also, with probability defined by packet drop rates, packets are randomly dropped during the passage to the boundary of the destination domain (D boxes in Figure 3).

3.2 Design of Distributed Wireless Network Simulation

In this section, we summarized the additional challenges to Genesis approach arising in wireless networks and reported by us in (Mandani and Szymanski 2003). As in the Genesis interface to wired networks, domains are simulated concurrently with each other over the same time interval. The domains freeze at user-specified intervals. At the time of freeze the inter-domain data exchange takes place. In GloMoSim, a node can schedule events (transmit and receive packets) while it is mobile. The current Genesis extension to GloMoSim accounts for the "mobility-trace" defined mobility in which the user specifies the speed, start and destination locations of the nodes in a configuration file. Knowing the above parameters, before the simulation starts, Genesis computes the time and location at which the node crosses the domain boundaries. Using this information, each domain simulator knows when and where the mobile node will be active in its domain.

The introduction of domain closures creates regions in the network topology which overlap at least two domains. Thus, a node in such a region is active in both domains at the same time. The Genesis domain simulators which simulate activities of such a node must include the same events for the node. To achieve this, the inter-domain messages include information about communications (packets received and sent) by nodes lying in the domain-closure. Each domain receiving this information checks if the same communications were ex-

ecuted for its copy of the nodes in question. If not, the time interval is re-simulated with the modified list of events for the offending node.

Each domain has at most eight domains as neighbors. Thus, each domain needs to communicate information about the activity of domains lying in its closure to its neighbors only. We achieve this by establishing a peer-to-peer connection between domains. In other words, each domain receives data from at most eight of domains during the freeze event. On exchange of this information, each domain checks whether it needs to go-back and re-simulate the freeze interval (based on the information collected and its own information).

3.3 Interoperability Design

We implemented our Genesis based on ns, SSFNet and GloMoSim, and enabled the interoperability among them within the Genesis framework.

To support interoperability among different systems, we defined generic network models and common flow-based message exchange formats. Mapping files were used to convert the flow information in common formats into local network data for different systems. We also created scenarios where we had mixed-mode traffics between a wired network (modeled using SSFNet) and a wireless network (modeled using GloMoSim). The wired network (SSFNet) viewed the wireless network (GloMoSim) as a black box, and vice versa. Proxy traffic aggents were created in Genesis to represent the network in the black box. In such an approach, the implementation details of each simulator are hidden from the others.

3.4 Design of Event-level Synchronization for BGP Simulations

In the simulation systems which use only event-level synchronization based on either conservative or optimistic protocol, the correct order of event delivery is guaranteed by the protocol. The price, however is frequent synchronization.

In Genesis, we take advantage of coarse granularity synchronization for TCP and UDP traffics, and at the same time synchronize BGP update messages by doing extra rollbacks, to reflect the actual routing dynamics in the network. To simulate BGP routers separately from the Genesis domain in each parallel AS domain simulator, and to make them produce BGP update messages for its neighbor domains, we introduced proxy BGP neighbor routers. Those are routers mirroring their counterparts which are simulated by other domain simulators. The proxy BGP routers do not perform the full routing functionality of BGP. Instead, they maintain the BGP sessions and collect the BGP update messages on behalf of their counterpart routers.

At the synchronization point in Genesis, the BGP update messages collected in the proxy BGP routers, if there are any, are forwarded to the corresponding destination AS domain simulators through a component called BGP agent. These update messages are delivered to the BGP agent in the destination AS domain through a Farmer-Agent framework, and are distributed there to the BGP routers which are the destinations of these messages.

During the Genesis checkpoint after one time interval, the BGP agent in each AS domain collects BGP update messages from other BGP agents. If it receives some update messages for the previous interval, it will force the AS domain simulator to rollback to the start time of the previous interval. Then, it inserts all the received update messages into its future event list. Its domain simulator will re-simulate the time interval again, and will "receive" these update messages at the correct simulation time and will react to them correspondingly. The BGP messages produced in the current execution might be different from the once seen at previous one. Hence, the rollback process might continue in domain simulators until all of them reach a global convergence, as showed in Figure 4. High cost of checkpointing the network state makes it impractical to introduce separate rollbacks for BGP activities. Hence, the UDP/TCP traffic checkpoints are used for all rollbacks in Genesis.

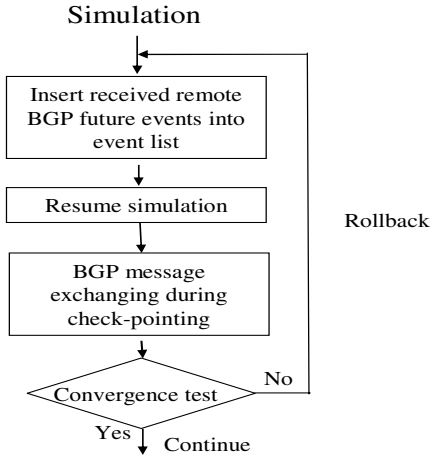

Figure 4: Synchronization for BGP Updates

3.5 Memory Distribution Design

Memory distribution is particularly challenging in Genesis, because of its special coarse granularity synchronization approach. In Genesis, within one time inter-

val, one domain simulator is working independently of others, simulating the partial traffics flowing within or through that domain. Other parts of these traffics, which are outside of that domain, are simulated by *proxy links* which compute the packet delays and losses based on flow "summaries" provided by the outside domain simulators. If the network information is completely distributed among the domain simulators, each one has information about only a part of the network. Hence, these simulators cannot simulate global traffics independently because information about some flow sources or destinations, or both will not be there. We should notice the difference here from other event-level synchronization systems. In those systems, to simulate distributed network, each individual event crossing the boundary is forwarded to remote simulators regardless of its "semantic meaning". Hence such parallel simulators do not need to simulate global flows independently, but they must synchronize their execution tightly.

In Genesis solution, each domain uses traffic proxies that work on behalf of their counterparts in the remote domains. Traffic proxies send or receive TCP or UDP data packets as well as acknowledgment packets according to the produced feedbacks. To simulate inter-domain flows, partial flows are constructed between local hosts and *proxy hosts*. Thus, in the simulation of one AS domain, the simulator just simulates one part of an inter-domain traffic by using *proxy hosts* and *proxy links*, as shown in Figure 5.

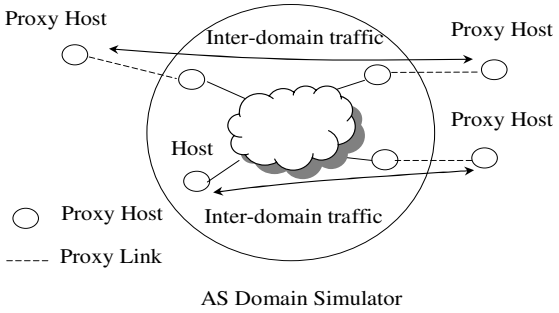

Figure 5: Proxy Hosts and Inter-domain Traffic

The actual traffic path between local hosts and remote hosts must be decided by inter-AS routing. For example, inter-AS routing changes can cause remote inbound traffic to enter the current AS domain from different entry points, thus routing the flow through a different path inside the domain. We developed a method, described below, to construct these remote traffic paths and to automatically adjust them to reflect the current inter-AS routing decision.

Global routing information consistency: To compute global routing in separate simulations, each of which has only a part of the network, IP address consistency is required to make the routers understand the routing update messages. In addition, we use BGP proxies and traffic proxies to act on behalf of their counterparts. To use routing data, these proxies need to use the IP addresses of their counterparts when they produce traffic packets. We used a global IP address scheme for the whole network, and introduced a mechanism of IP address mapping, which translates local addresses to and from global addresses used in our BGP update messages. In our global IP address scheme, domains are assigned different IP address blocks to avoid address conflicts among domain simulators. Inter-domain routing information is stored based on these global addresses. Each proxy host stores the IP address of its counterpart host which has a global IP address. When packets are sent from proxy hosts, the IP addresses in the packet headers would be replaced with corresponding global IP addresses. In this way, the addresses in these packets are consistent with the routing information and can be correctly routed to the destinations.

Remote host, traffic and link: Those definitions were added to the network definitions. *Remote host* defines the traffic host (source or sink) which is not within the current simulating domain, and specifies the global IP address for this proxy. *Remote traffic* pattern allows the definition of a traffic which will use proxy IP address instead of its own local IP address. *Remote link* is defined to connect the *remote host* to the current domain, and it is implemented as a Genesis *proxy link* which can adjust its link delay and applied packet drop rates during the simulation.

Remote traffic path construction: The difficult part of remote traffic path construction was to decide how to connect *proxy hosts* to the current AS domain. Changes in inter-AS routing decision might change the entry (exit) point of traffic packets to (from) the domain. Such a change cannot be determined during the network construction phase. We designed a structure which connected remote traffic hosts to a *proxy switch*, instead of connecting them to any entry point directly, as shown in Figure 6. When a packet sent by a *proxy host* reaches the *proxy switch*, the *proxy switch* will lookup an internal mapping from flow id to the current inter-AS routing table, and will forward this packet via the correct inbound link to one of the BGP routers on the domain boundary. If the inter-AS routing is changed by some BGP activities later, the *proxy switch* will automatically adjust its internal mapping, and the packets with the same flow id will be forwarded to a different inbound link.

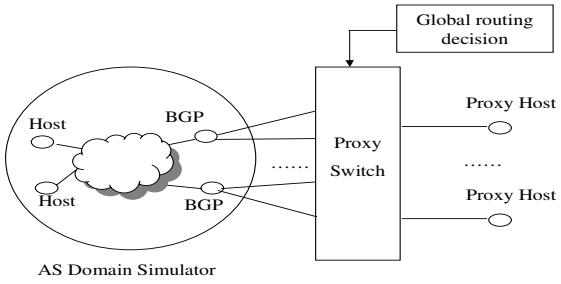

Figure 6: Remote Traffic Path Construction

4 PERFORMANCE EVALUATION

This section briefly summarize results from a series of simulations that we run on a large network model using the distributed Genesis and which were initially described by us in (Szymanski et al. 2003).

4.1 Simulation Model

To test the performance and scalability of Genesis in large-scale network simulations, we use a modified version of the baseline model defined by the DARPA NMS community (NMS Baseline Model). The topology for the model that we are using can be visualized as a ring of nodes, where each node (representing an AS domain) is connected to one node preceding it and another one succeeding it. We refer to each node or AS domain as the "campus network", as shown in Figure 7. Each of the campus networks is similar to the others and consists of four subnetworks. In addition, there are two additional routers not contained in the subnetwork, as show

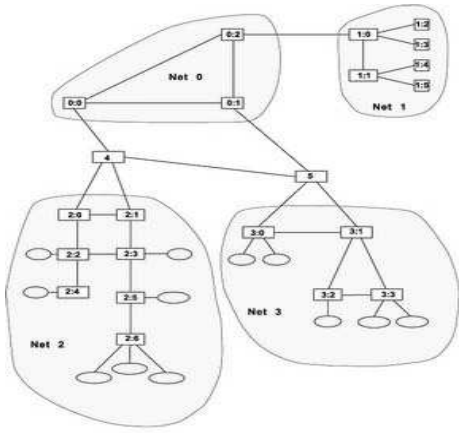

Figure 7: One campus network

The subnetwork labeled Net 0 consists of three routers in a ring, connected by links with 5 ms delay and 2 Gbps bandwidth. Router 0 in this subnetwork acts as a BGP border router and connects to other campus networks. Subnetwork 1 consists of 4 UDP servers. Subnetwork 2 contains seven routers with links to the LAN networks as shown in the diagram. Each of the LAN networks has one router and four different LAN's consisting of 42 hosts. The first three LAN's have 10 hosts each and the fourth LAN has 12 hosts. Each of the hosts is configured to run as a UDP Client. Subnetwork 3 is similar to Subnetwork 2, so internal links and LAN's have the same property as those in Subnetwork 2.

The traffic that is being exchanged in the model is generated by all the clients in one domain choosing a server randomly from the Subnetwork 1 in the domain that is a successor to the current one in the ring. We used different send-intervals of 0.1, 0.05 and 0.02 second to vary the traffic intensities, and used different numbers of nodes (AS domains) to vary the size of the network. Each simulation was run for 400 seconds of the simulated time.

All tests were run on up to 30 processors on Sun 10 Ultrasparc workstations, which were interconnected by a 100 Mbit Ethernet. One of these workstations had 1G large memory, and each of the others had at least 256M dedicated memory. In the simulations under distributed Genesis, the number of processors used was equal to the number of campus networks.

4.2 Experiment Results

Genesis distributively constructs and simulates BGP routers in AS domain simulators. To measure scalability of this solution in terms of network size, we simulated BGP networks of 10, 15, 20 and 30 AS domains, each run by a Sun 10 Ultrasparc workstation with 256 MB of memory. As shown in Figure 8, when the number of AS domains increases, unlike SSFNet, the memory usage of one Genesis AS simulator does not increase much. As a result, by utilizing more computers with smaller memories, we are able to simulate much larger networks.

Memory usage of simulation is related not only to the static network size, but also to the network dynamics. We increased the traffic intensity by reducing the traffic send-interval from 0.1 to 0.05 and 0.02 second. As shown in Figure 9, although we did not observe very big changes in memory usage in SSFNet on this campus network model, Genesis showed even smaller increase in memory size with the same changes in traffic (thanks to its smaller base memory size).

As we have shown, Genesis achieved execution

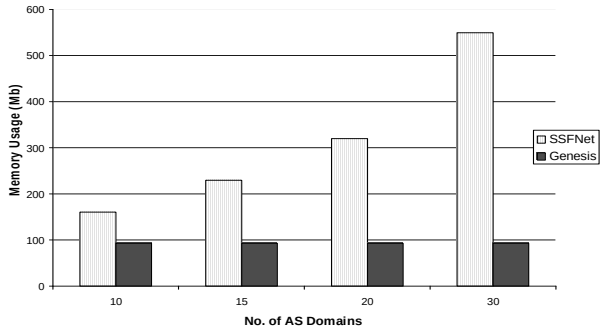

Figure 8: Memory usage of SSFNet and one Genesis domain simulator for simulations of different AS domains

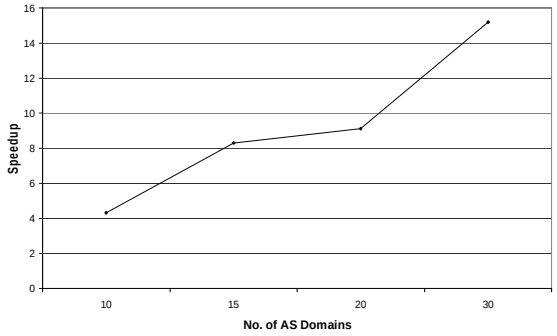

Figure 10: Speedup achieved for simulations of different BGP network sizes

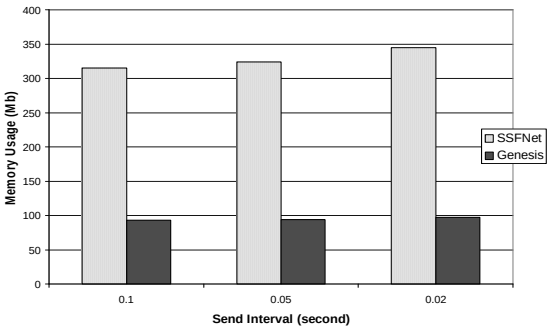

Figure 9: Memory usage of SSFNet and Genesis for 20-AS BGP network simulations with different send-rates

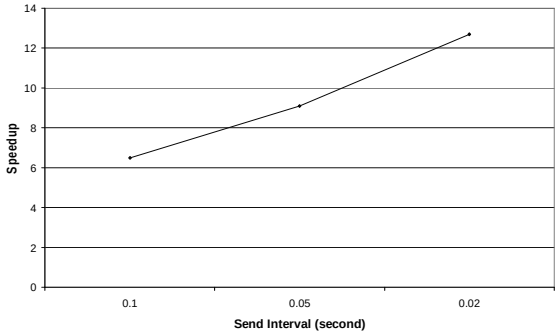

Figure 11: Speedup achieved for 20-AS BGP network simulations with different send-rates

speedups thanks to its coarse granularity synchronization mechanism. In addition, despite the extra overheads introduced by distributing the network, good speedups where achieved for 10, 15, 20 and 30 domain simulators with BGP routers. The Genesis domains were defined by the AS boundaries. Figure 10 shows the speedups of simulations for these networks.

Figure 11 shows that Genesis achieved higher speedups with higher traffic intensities. This is because with higher traffic intensity, more events need to be simulated in a fixed simulation time. Theoretical analysis tells us that sequential simulation time includes terms of order $O(n * \log(n))$, due to sorting event queues. Genesis distributes the simulation among domain simulators, which reduces the number of events needed to be simulated by one simulator, so it can achieve higher speedups when the traffic increases as well as when the network size increases.

To measure the accuracy of the simulation runs, we monitored the per flow end-to-end packet delays and packet drop rates. We compared the results from distributed Genesis with the results from sequential simulations under SSFNet, and calculated the relative errors. Our results showed that for most of the flows, the relative errors of both packet delay and drop rate were within the range from 2% to 10%, while a small number of individual flows had higher relative errors of up to 15% to 20%. Considering the fact that in a simulation with large number of flows, the network condition was mainly determined by the aggregated effects of sets of flows, we calculated the root-mean-square of the relative errors on each set of flows which went through the same domain. These root-mean-squares of relative errors were below 5%, which seems sufficiently close approximation of the sequential simulation for many applications.

Simulation results showed that by fully distributing the simulation in Genesis, we gained the scalability of memory size. In addition, the parallel simulation in Genesis still achieved performance improvement in this distributed framework, compared to sequential simulations.

5 COMMUNICATION FRAMEWORK

The current Farmer-Worker framework used in Genesis is a simple centralized client/server system. This framework worked efficiently with modest number of domain simulators. In order to maximize the efficiency of Genesis with huge number of domain simulators, we are currently working on a new framework with the following new features: peer-to-peer traffic data communications among domains; hierarchical communication structures for global synchronization. New framework is expected to provide even better scalability for large scale network simulations.

6 CONCLUSIONS

The need for scalable and efficient network simulators increases with the rapidly growing complexity and dynamics of the Internet. In this paper we introduced a novel scheme, implemented in Genesis, to support scalable, efficient parallel network simulation. Our results indicate that the superlinear speedup for the single iteration step is possible and is the result of the non-linear complexity of the network simulation. Our approach achieved significant speedup in the simulations of different network scenarios.

We also demonstrated that our system can work efficiently with fully distributed network memory. This design reduces and makes scalable the memory size requirement for large-scale network simulations, especially large BGP network simulations which require very large memory size. As a result, Genesis is able to simulate huge networks using limited computer resources.

ACKNOWLEDGMENTS

This work was partially supported by the DARPA Contract F30602-00-2-0537 and an URP Grant from CISCO Systems Inc. The content of this paper does not necessarily reflect the position or policy of the U.S. Government or CISCO Systems—no official endorsement should be inferred or implied.

REFERENCES

Bhatt, S., R. Fujimoto, A. Ogielski, and K. Perumalla, Parallel Simulation Techniques for Large-Scale Networks. *IEEE Communications Magazine*, 36, 1998.

Chandy, K.M., and R. Sherman. Space-time and simulation. *Proc. of Distributed Simulation*, Society for Computer Simulation, 53–57, 1989.

Law, L.A., and M. G. McComas. Simulation software for communication networks: the state of the art. *IEEE Comm. Magazine*, 32:44–50, 1994.

Liu, B., D. R. Figueirido, Y. Guo, J. Kurose, and D. Towsley. A study of networks simulation efficiency: Fluid simulation vs. packet-level simulation. *Proceedings of IEEE Infocom 2001*, April 2001.

Liu, B., Y. Guo, J. Kurose, D. Towsley, and W. Gong. Fluid simulation of large scale networks: issues and tradeoff. *PDPTA'99*, Las Vegas, NV, June 1999, 2136–2142.

K. Mandani and B.K. Szymanski. Integrating Distributed Wireless Simulation Into Genesis Framework. *Summer Computer Simulation Conference*, Montreal, Canada, July 2003, to appear.

Nicol, D. Comparison of network simulators revisited. Available at `<http://www.ssfnet.org/Exchange/gallery/dumbbell/dumbbell-performance-May02.pdf>`, May 2002.

NMS (*Network Modeling and Simulation DARPA Program*) baseline model. See web site at `<http://www.cs.dartmouth.edu/~nicol/NMS/baseline/>`.

B.K. Szymanski, A. Saifee, A. Sastry, Y. Liu and K. Madnani. Genesis: A System for Large-scale Parallel Network Simulation. Proc. 16th Workshop on Parallel and Distributed Simulation, Washington, DC, May 2002, pp. 89-96.

B.K. Szymanski, Y. Liu, and R. Gupta. Parallel Network Simulation under Distributed Genesis. Proc. 17th Workshop on Parallel and Distributed Simulation, San Diego, CA, June 2003, pp. 61-68.

Szymanski, B., J.-F. Zhang, and J. Jiang. A Distributed Simulator for Large-Scale Networks with On-Line Collaborative Simulators. *Proc. European Multisimulation Conference - ESM99*, Warsaw, Poland, SCS Press, II:146–150, June, 1999.

Ye, T., D. Harrison, B. Mo, S. Kalyanaraman, B. Szymanski, K. Vastola, B. Sikdar, and H. Kaur. Traffic management and network control using collaborative on-line simulation. *Proc. International Conference on Communication, ICC2001*, 2001.

AUTHOR BIOGRAPHIES

BOLESLAW K. SZYMANSKI, is a Professor of Computer Science at Rensselaer Polytechnic Institute. His research interests include simulation methodology, networking and parallel and distributed computing.

YU LIU is a Ph.D. candidate in Computer Science department at Rensselaer Polytechnic Institute. His research interests include distributed network simulation and network managements.