Proc. Summer Computer Simulation Conference SCSC04, San Jose, CA, July 2004,
SCS Press, San Diego, CA, pp. 271-278

Distributed Packet-Level Simulation for BGP Networks under Genesis *

Yu Liu and Boleslaw K. Szymanski
Department of Computer Science, RPI, Troy, NY 12180, USA
{*liuy6, szymansk*}*@cs.rpi.edu*

Abstract

The complexity and dynamics of the Internet is driving the demand for scalable and efficient network simulation. Parallel and distributed network simulation techniques make it possible to utilize the power of multiple processors or clusters of workstations to simulate large-scale networks efficiently. However, synchronization overheads inherent in parallel and distributed simulations limit the efficiency and scalability of these simulations.

We developed a novel distributed network simulation framework and synchronization approach which achieved better efficiency than conventional approaches. In this framework, BGP networks are partitioned into domains of Autonomous Systems (ASes), and simulation time is divided into intervals. Each domain is simulated independently of and concurrently with the others over the same time interval. At the end of each interval, packet delays and drop rates for each inter-domain flow are exchanged between domain simulators. The simulators iterate over the same time interval until the exchanged information converges to the value within a prescribed precision before progress to the next time interval. This approach allows the parallelization with infrequent synchronization, and achieves significant simulation speedups. In this paper, we focus on the design of distributed BGP network simulation in Genesis in which many BGP ASes can be assigned to a single Genesis domain. We also report our experimental results that measure Genesis distributed efficiency in large scale BGP network simulations.

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 [2]. Conventional simulation techniques require tight synchronization for each individual event that crosses the processor boundary [1]. The inherent characteristics of network simulations are the small 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 execution under the traditional protocols [1].

Another difficulty in network simulation is the large memory size required by large-scale network simulations. With the emerging requirements of simulating larger and more complicated networks, the memory size becomes a bottleneck. When network configuration and routing information is centralized, large memory is needed to construct the simulated network. As a result, simulations of large scale BGP networks were hindered by the large memory size required to construct and store inter-domain routing information. Hence, we believe that to simulate truly large networks, the comprehensive, distributed memory approach needs to be developed.

We have described the details of our novel approach to scalability and efficiency of parallel network simulation in Genesis in [8, 9]. Genesis combines simulation and modeling in a single execution. It decomposes a network into parts, called domains, and simulation time into intervals, and simulates each network partition independently and concurrently over one interval. After each time interval, flow delays and packet drop rates observed by each domain simulator are exchanged with others. Each domain simulator will model the traffic external to its own domain based on the flow data received from other domains. *Link proxies* are used in Genesis to represent the external traffic paths. Domain simulators iterate over the same time interval until they reach a global convergence with controllable precision. An execution scheme is shown in Figure 1 that illustrates also synchronization between the repeated itera-

*This work was partially supported by the DARPA Contract F30602-00-2-0537 and by the grant from the University Research Program of 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.

tions over the same time interval and use of checkpoints for the domain simulators [9].

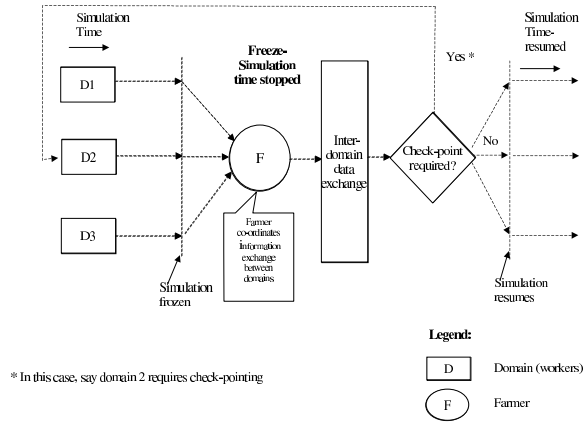

Figure 1: Genesis Execution Scheme

In this paper, we focus on the design of distributed BGP network simulation in Genesis. In particular, we describe the system extension that allows the user to assign many BGP ASes to a single Genesis domain. To measure the system efficiency, we partitioned large scale BGP networks into simulation domains and distributively constructed them in cluster of simulators. Our experimental results showed that Genesis achieved higher distributed efficiency in large scale BGP network simulations than SSFNet.

2 BGP SYNCHRONIZATION MECHANISMS IN GENESIS

Genesis uses coarse granularity synchronization to simulate network traffic, e.g., TCP or UDP flows. This is achieved by having parallel simulators loosely cooperating with each other. They simulate partitioned network concurrently with and independently of each other in one iteration. They exchange data only during the checkpoints executed between iterations. In addition, individual packets are not stored or exchanged among parallel simulators. Instead, each data flow is summarized based on some pre-defined metrics, and only the summarized traffic information is exchanged among parallel simulators. This approach avoids frequent synchronization of parallel simulators. We have shown that it achieved significant speed-up for TCP or UDP traffic simulations. The total execution time could be decreased by an order of magnitude or more [8]. Our primary application was the use of the on-line simulation for network management [12].

For distributed simulation of BGP protocol, Genesis embeds an event-level synchronization mechanism into its coarse granularity synchronization framework. This work was done using our previous development of Genesis based on SSFNet [6], which was reported in [11]. In Genesis, to simulate a network running BGP protocol for inter-AS (Autonomous System) routing, with background TCP or UDP traffic, the network is decomposed along the boundaries of AS domains. Each parallel simulator simulates one AS domain, and loosely cooperates with other simulators. When there are BGP update messages that need to be delivered to neighbor AS domains, the BGP event-level synchronization mechanism in Genesis guarantees that these messages will be delivered in the correct time-stamp order.

In addition, to support more flexible network partitioning in BGP networks, we extended our system to support BGP AS grouping. By providing a mapping scheme to Genesis, multiple BGP ASes can be grouped into one Genesis domain and assigned to one simulator. This made it possible to apply different partitioning schemes to the same BGP network model.

3 GENESIS BGP SIMULATION DESIGN OVERVIEW

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 high granularity synchronization for TCP and UDP traffic, and at the same time synchronize BGP update messages by doing extra rollbacks, to reflect the actual routing dynamics in the network.

In Genesis, simulators are running independently of each other within one iteration. 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 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 the connections among BGP agents, and are

distributed there to the BGP routers which are the destinations of these messages.

This framework enables the system to exchange real BGP message data among Genesis simulators. But this is not a full solution yet. Within the independent simulation of one iteration in Genesis, the BGP routers produce update messages for their neighbors, but do not receive update messages from their neighbors in other AS domains. Had they received these update messages, as it happens in an event-level synchronization simulation system, they would have probably produced different update messages. In addition, the routing might also have been changed. To simulate BGP protocol correctly, these BGP updates need to be executed in their correct time-stamp order in each BGP router. Genesis achieved this event-level synchronization for BGP updates by doing extra rollbacks.

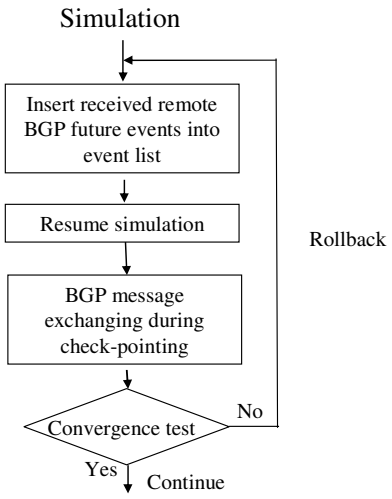

Figure 2: Synchronization for BGP Update Messages

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 reiterate 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 reiteration might be different from those seen during a previous iteration. Hence, the rollback process might continue in domain simulators until all of them reach a global convergence (the update messages in subsequent rollback iterations are the same for each domain). Figure 2 shows the flowchart of rollback in the BGP agent. 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.

3.1 Grouping Multiple ASes in One Genesis Domain

The initial design of distributed BGP simulation in Genesis supported partitioning the network on the boundary of AS domains. As a result, each distributed Genesis simulator simulated one AS domain. The BGP AS domain IDs could be used as the identifier of Genesis simulators and used to route BGP messages to the corresponding destination simulator.

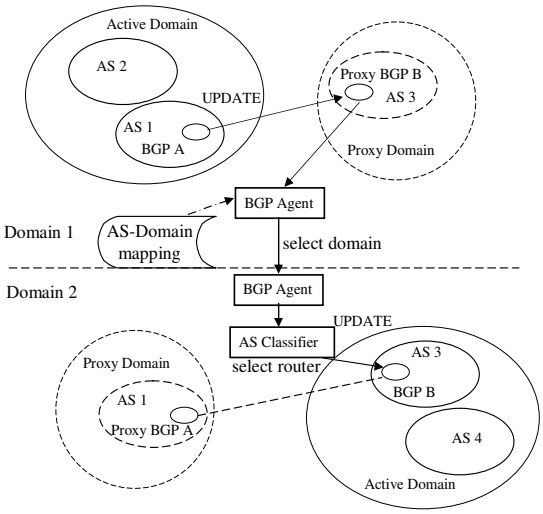

Figure 3: BGP Message Forwarding with AS Grouping

To support more flexible network partitioning and study the effects of different partitioning schemes, Genesis was extended to support AS domain grouping in distributed simulators. A BGP update message sent to a BGP peer router is routed through two layers to reach the destination. The first layer is a mapping between BGP ASes and Genesis domains. This can be done by specifying a grouping factor (e.g., N BGP ASes per domain) when each group has the same number of ASes, or by providing a mapping file when the grouping is not even. This grouping information is shared by all domain simulators and is used to forward any BGP update messages to the domain which contains the corresponding destination BGP router. The second layer contains a message classifier in each domain. It holds a list of all the BGP routers in the ASes in that domain. The classifier first identifies the destination BGP router address of

each received BGP messages, and then it forwards the messages to its destination. Figure 3 shows the BGP message forwarding in the AS grouping scenario.

4 SIMULATION WITH DISTRIBUTED MEMORY

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. For example, [4] 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.

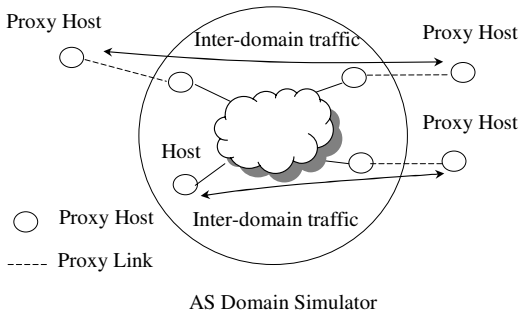

Figure 4: Proxy Hosts and Inter-domain Traffic

To fully distribute the simulation of BGP network, traffic proxies are introduced in each domain. They work on behalf of their counterparts in the remote domains by sending or receiving 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 4.

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.

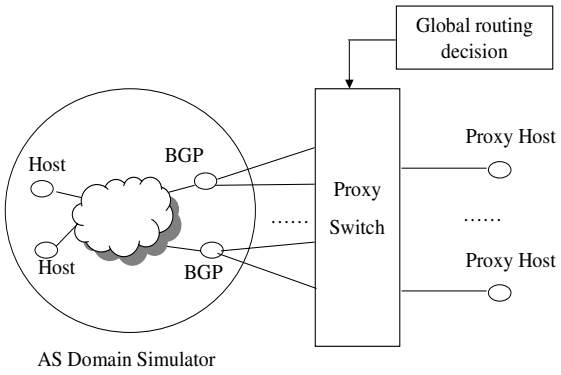

Figure 5: Remote Traffic Path Construction with Proxy Switch

The difficult part of remote traffic path construction was to decide how to connect *proxy hosts* to the current AS domain. 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 5. 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.

In a fully distributed simulation (the network model is constructed distributively) which supports dynamic routing, global routing decisions need to be computed distributively as well. The support of distributed BGP simulation made this distributed computation possible. On the other hand, fully distributed network construction significantly reduced the memory requirement on each individual computer, which facilitated the simulation of very large BGP network models.

5 PERFORMANCE EVALUATION

5.1 Campus Network Simulation Model

To test the performance and scalability of the Genesis and to compare them to those of SSFNet, we use a modified version of the baseline model defined by the DARPA NMS community [5]. The topology of this model can be visualized as a ring of nodes, where each node (representing an AS domain shown in Figure 6) is connected to one node preceding it and another one succeeding it. We refer to each node or AS domain as the "campus network". 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 shown in the diagram.

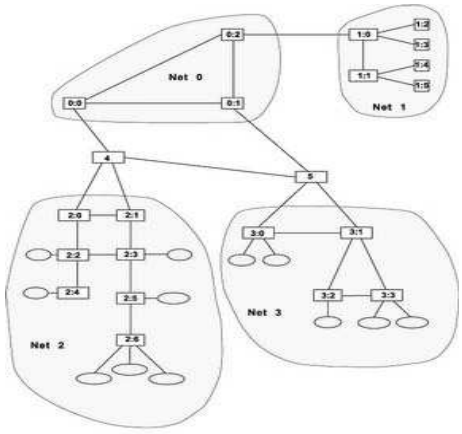

Figure 6: One campus network

The subnetwork labeled Net 0 consists of three routers in a ring, connected by links with 5ms delay and 2Gbps 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. Internal links and LAN's have the same property as 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.

This individual campus network model was the same as the one we used in [10]. However, in the experiments in this paper, we set the traffic send-rate at the higher rate of 0.02 second per packet. In addition, we reduced the total simulation time from 400 seconds to 100 seconds, while keeping the traffic time as 60 seconds. As a result, the network was loaded with higher traffic intensity throughout the simulation and the communication overheads during each checkpoint became comparatively smaller. These settings were used to facilitate our study of the performance impacts of different types of traffic, local or remote, on the simulation.

In our network model, we connected 24 such campus networks in a ring as described above. We grouped every N adjacent campus networks into one Genesis domain and assigned it to one distributed simulator, as shown in Figure 7. N was assigned the following sequence of values: 1, 2, 3, 6, 12 and 24, thus the number of domains was, correspondingly 24, 12, 8, 4, 2 and 1.

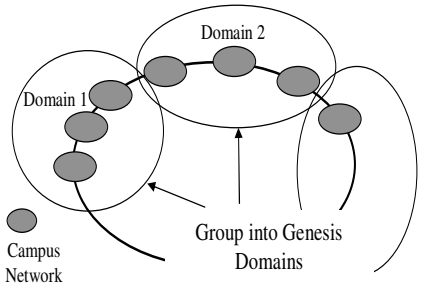

Figure 7: Grouping adjacent campus networks

Our experiments of Genesis were run on Sun UltraSPARC-III, 750MHz dual-cpu machines, which were interconnected by a 100Mbit Ethernet. Each Genesis domain simulator was assigned to one processor.

5.2 Campus Network Experimental Results

In order to compare the simulation performance of Genesis and SSFNet, we first did some experiments under SSFNet on the same campus network model. We ran a set of simulations on a Sun UltraSPARC-II 4-CPU machine, and varied the number of processors in these SSFNet parallel simulations setting it to 1, 2, 3 and 4. Ring size of 6 campus networks was used for the 1, 2 and 3 processors simulations, while ring size of 4 campus network was used for 1 and 4 processors simulations. We always assigned adjacent campus networks to the same processor. Thus, by varying the number of processors, we varied the percentage of remote traffic (in this case, traffic exchanged between parallel processors) as well.

Table 1: SSFNet Parallel Efficiency A

Network Size(CN)	Partitions	Remote Traffic	Speedup	Distributed Efficiency
6	1	0%	N/A	N/A
6	2	50%	1.78	89%
6	3	75%	1.68	56%
4	4	100%	2.1	53%

Table 1 shows the experimental results that demonstrate that when the percentage of remote traffic increased, the parallel efficiency of ssfnet decreased. The parallel efficiency dropped to just above 50% when the

percentages of remote traffic was close to 100%. Because all links between two campus networks were the same, the lookahead was the same when the partitioning were different in these simulations. However, larger amount of events (packets) needed to be exchanged and synchronized among processors introduced high overheads in the parallel simulation.

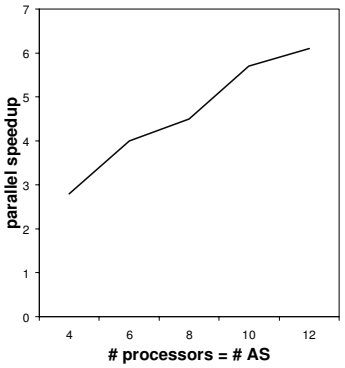

Figure 8: SSFNet Parallel Simulation Speedups

The parallel simulation performance of SSFNet was also reported in [7], where experiments were done on Sun enterprise server with up to 12 processors. A similar ring of campus network model was used while the size of an individual campus network was smaller. We included the reported results for the simulations in which each campus network exchanged traffic only with its neighboring network in Figure 8. We computed the parallel efficiencies of those experiments based on these results and showed them in Table 2.

Table 2: SSFNet Parallel Efficiency B

Network Size(CN)	Partitions	Remote Traffic	Speedup	Distributed Efficiency
4	4	100%	2.7	67.4%
6	6	100%	4.0	66.7%
8	8	100%	4.5	56.2%
10	10	100%	5.7	57.0%
12	12	100%	6.1	50.8%

The results in Table 2 shows that when the number of processors increased, the parallel efficiencies dropped not very significantly, from above 60% to about 50%. Because in these simulation, each campus network was simulated by one processor and only exchanged traffic with its neighboring network, the remote traffic percentages were 100%. The amount of remote traffic and the overheads for each processor to handle remote events from other processors were about the same. Also the lookahead was the same because the link delays between campus network did not change. However, with more parallel processors, the overheads of global synchronization increased. From these results, we observed that in the simulations with high percentage of remote traffic, the parallel efficiency of SSFNet was only about 50% to 60%.

Table 3: Distributed Efficiency in Genesis Distributed Simulation

Network Size(CN) /Domains	Remote Traffic	Packet-hop Difference	Speedup	Distributed Effic.
24/ 1	0%	N/A	N/A	N/A
24/ 4	28.6%	3.8%	4.15	104%
24/ 8	50.0%	3.2%	7.73	96.7%
24/12	66.7%	4.0%	11.0	91.7%
24/24	100%	5.0%	19.8	82.6%

Another set of experiments were done under Genesis. 24 campus networks (CN) were connected in a ring, as described in the previous section. We grouped N adjacent CNs into one Genesis domain, and constructed each domain distributively in Genesis domain simulators. N was varied as we set it to 24, 6, 3, 2 and 1. Because each campus network exchanged traffic only with its adjacent networks, when we changed the grouping scheme, the percentage of remote traffic (traffic exchanged between distributed domains) in the simulation also changed. We compared the results of different grouping schemes with the non-distributed simulation (24 CNs simulated in one simulator) and computed the efficiencies.

Table 3 shows the experimental results. From these results, we observed that when the percentage of remote traffic increased from about 28% to 100%, the distributed efficiency dropped slightly. Even with 100% remote traffic, the distributed efficiency was still above 80%. In addition, the error of the total packet hops was within 5% compared to non-distributed, accurate simulation. Genesis showed better performance in these experiments than SSFNet.

We attributed this advantage of Genesis to its coarse granularity synchronization approach. Unlike conventional event-level synchronizations in which each individual remote packet (event) need to be exchanged and synchronized, Genesis aggregated these simulation data and reduced both of the amount of data exchange and frequency of synchronization. As the result, higher percentage of remote traffic did not introduce significant

synchronization overheads in Genesis, as they usually did in other conventional simulation systems.

5.3 Distributed UDP Flooding in U.S. Backbone Network Model

In order to study the performance of Genesis with large scale, real-world network models, we selected the U.S. backbone network model introduced in [3]. This model was a large scale BGP network topology consisting of 8 national-level ISP networks. The full topology included 9828 backbone routers and 787 BGP speakers.

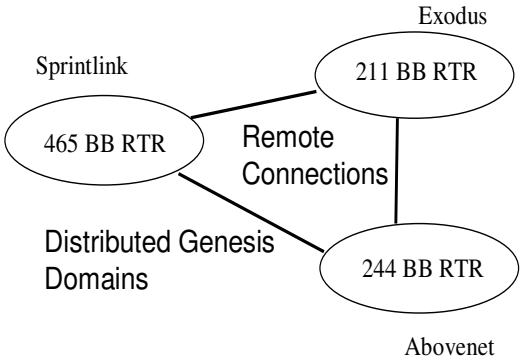

Figure 9: Network Model of Three ISP's

Due to our hardware limitation, we selected three ISPs to construct a subset of this U.S. backbone network topology for our experiment. The selected ISP's were Sprintlink with 465 backbone routers and 56 BGP speakers, Exodus with 211 backbone routers and 32 BGP speakers and Abovenet with 244 backbone routers and 39 BGP speakers. We connected them as a clique, as shown in Figure 9. We distributively constructed each of these ISP BGP networks into one Genesis domain and assigned each of them to one distributed simulator.

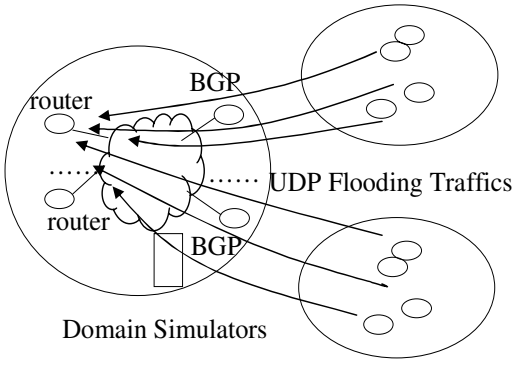

Figure 10: Distributed UDP Flooding Simulation

We simulated distributed UDP flooding in this network model under Genesis. Distributed UDP flooding simulation was used in network security research to study of the effectiveness of different protection techniques. The "distributed" here means that the UDP flood is generated from distributed nodes in the network being simulated, and it can be simulated by sequential simulation systems. However, to simulate it in large-scale network models, e.g. the Internet, parallel and distributed simulation provides a more scalable solution. Yet, thanks to its distributed traffic pattern and high traffic intensity, it usually introduces high synchronization overhead in parallel and distributed simulations.

We designed our experiment to demonstrate the performance of Genesis in a simulation with burst-out distributed UDP flooding. The network model introduced in [3] is a "bare-boned" network which consists of only routers. We introduced UDP clients and servers into the network and attached them to different routers to generate UDP traffic. At the first stage of the simulation, each of the three ISP networks had only background traffic within each ISP network. At simulation time 40 second, we randomly selected 20 distributed hosts in each ISP network to generate high intensity UDP traffic (15Mb/sec) that flooded the other two ISP networks, and selected one host in each ISP networks as the victims of these attacks. These burst-out flooding traffic increased the percentage of remote traffic in the network to over 90%. We monitored the packet-hop rate changes during the simulation. Higher packet-hop rate represents higher simulation efficiency, and lower packet-hop rate represents higher synchronization overheads in the simulation.

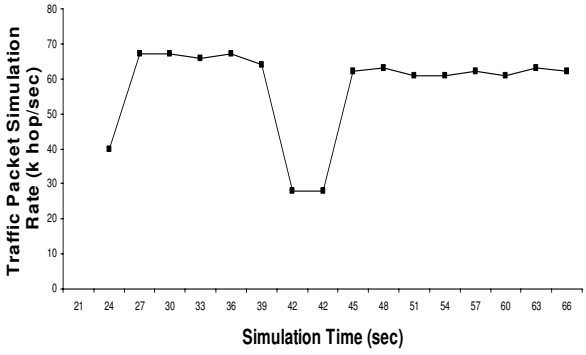

Figure 11: Simulation Rate

Figure 11 shows the experimental results. We observed that when the distributed UDP flooding burst out, the packet-hop rate dropped significantly. This

was because initially, during the synchronization, Genesis rolled back the simulation for the iteration in which the burst happened, to adjust each distributed simulator for this network traffic change [9]. After the Genesis simulation converged on this change in the network, it proceeded without this rollback. In this stage, because of the distributed UDP flooding traffic flowing from one distributed simulator to another, the percentage of remote traffic in the simulation increased from 0% to over 90%. Usually this change would introduce significant synchronization overhead, as we showed above in SSFNet simulations. One way to measure the network simulation efficiency and overhead is to measure the packet-hop rate in the simulation. The higher the packet-hop rate, the less synchronization overhead in the simulation. We compared the packet-hop rates in Genesis before and after we introduced distributed UDP flooding into the simulation, and observed that after the rollback, the packet-hop rate raised back to above 90% of the rate before the flooding. Genesis was able to simulate very high percentage of remote traffic with very little extra synchronization overhead.

6 CONCLUSIONS

In this paper, we demonstrated that Genesis worked efficiently in distributed network simulations. Our experimental results showed that in SSFNet parallel simulations, when each single campus network was assigned to one processor, the parallel efficiency was limited in the range from 50% to 60% when up to 12 processors were used. The synchronization overheads increased significantly when the degree of parallelism increased. In contrast, Genesis achieved better distributed efficiency at about 80% when each single campus network was assigned to one distributed simulator and up to 24 processors were used, thanks to its coarse granularity synchronization approach which did not require exchanging and synchronizing each individual remote packet. Genesis significantly reduced the synchronization overheads and was more scalable in distributed simulations.

In addition, because of its aggregation of the simulation information being exchanged, Genesis reduced both the amount and frequency of data exchanges among distributed simulators. Our experimental results showed that when the network partitioning scheme was changed and the percentage of remote events in the distributed simulation was increased, Genesis consistently achieved high distributed efficiency at above 80%. It does not require changes of synchronization algorithm for different network topologies, network channel latencies or network partition schemes to achieve good performance. Thanks to this approach, Genesis became a very general system for distributed network simulation.

With the support of flexible partitioning of BGP networks, and distributed simulation of large scale, real world network models, study of the performance and stability of BGP and defensive techniques against flooding and worm attacks will be directions of our future research.

References

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

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

[3] Liljenstam, M., J. Liu and D.M. Nicol. Development of an internet backbone topology for large-scale network simulations. *Proc. of the 2003 Winter Simulation Conference*, December 2003.

[4] Nicol, D. Comparison of network simulators revisited. *Available at http://www.ssfnet.org/Exchange/gallery/dumbbell/dumbbell-performance-May02.pdf*, May 2002.

[5] NMS (*Network Modeling and Simulation DARPA Program*) baseline model. *See web site at* http://www.cs.dartmouth.edu/ nicol/NMS/baseline/.

[6] SSFNet(*Scalable Simulation Framework Network Models*). *See web site at* http://www.ssfnet.org/homePage.html.

[7] Java SSFNet parallel performance. *See web site at* http://www.ssfnet.org/parallelSolaris.pdf.

[8] Szymanski, B., 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*, pages 89–96, May 2002.

[9] Szymanski, B., Y. Liu, A. Sastry, and K. Madnani. Real-time on-line network simulation. *Proc. 5th IEEE Int. Workshop on Distributed Simulation and Real-Time Applications, DS-RT 2001*, August 13–15, 2001, IEEE Computer Society Press, Los Alamitos, CA, 2001, pages 22–29.

[10] Szymanski, B., Y. Liu and R. Gupta. Parallel network simulation under distributed Genesis. *Proc. 17th Workshop on Parallel and Distributed Simulation*, June 2003.

[11] Szymanski, B., Q. Gu, and Y. Liu. Time-network partitioning for large-scale parallel network simulation under ssfnet. *Proc. Applied Telecommunication Symposium, ATS2002*, B. Bodnar (edt), San Diego, CA, April 14–17, SCS Press, pages 90–95, 2002.

[12] 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.