Proc. 17th Workshop on Parallel and Distributed Simulation, San Diego, CA, June 2003, pp. 61-68

Parallel Network Simulation under Distributed Genesis *

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

Abstract

We describe two major developments in the General Network Simulation Integration System (Genesis): the support for BGP protocol in large network simulations and distribution of the simulation memory among Genesis component simulations.

Genesis uses a high granularity synchronization mechanism between parallel simulations simulating parts of a network. This mechanism uses checkpointed simulation state to iterate over the same time interval until convergence. It also replaces individual packet data for flows crossing the network partitions with statistical characterization of such flows over the synchronization time interval. We had achieved significant performance improvement over the sequential simulation for simulations with TCP and UDP traffic. However, this approach can not be used directly to simulate dynamic routing protocols that use underlying network for exchanging protocol information, as no packets are exchanged in Genesis between simulated network parts. We have developed a new mechanism to exchange and synchronize BGP routing data among distributed Genesis simulators. The extended Genesis allows simulations of more realistic network scenarios, including routing flows, in addition to TCP or UDP data traffic.

Large memory size required by simulation software hinders the simulation of large-scale networks. Based on our new support of distributed BGP simulation, we developed an approach to construct and simulate networks on distributed memory using Genesis simulators in such a way that each participating processor possesses only data related to the part of the network it simulates. This solution supports simulations of large-scale networks on machines with modest memory size.

*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.

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 [3]. 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 in a network simulation, large memory is needed to construct the simulated network. Moreover, memory size used by the simulation increases also with the intensity of traffic loads that impact the average size of the future event list. Although memory requirements can be tampered by the good design and implementation of the simulation software [4], 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 [7, 8]. 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 un-

til they reach a global convergence with controllable precision. An execution scheme is shown in Figure 1 that illustrates also synchronization between the repeated iterations over the same time interval and use of checkpoints for the domain simulators [8].

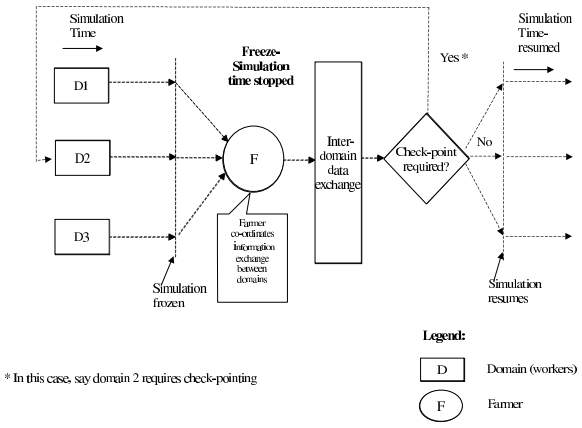

Figure 1. Simulation Execution Scheme in Genesis

Genesis achieved performance improvement thanks to simulating smaller number of events and its infrequent, high-granularity synchronization mechanism. No individual packet was synchronized between two parallel simulations; instead, packets were "summarized" on some metrics (delay, drop rate, etc.) and only these data were exchanged between domains at the end of each time interval. This approach 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; 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 new synchronization mechanism in Genesis.

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, the requirement most easily achievable through the systems with shared memory. In this paper, we describe the newly developed version of Genesis that fully distributes memory usage in Genesis. This version overcomes the memory size limitation.

2 Synchronization Mechanisms in Genesis

Genesis uses a high granularity synchronization mechanism to simulate network traffics, 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 predefined 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 [7]. Our primary application was the use of the on-line simulation for network management [10].

The Genesis approach was designed for network data traffics, e.g., TCP or UDP traffics. In many network simulation scenarios, the real data of the traffics packets are not important to the simulation result. However, for some other traffic, especially related to network routing protocols, e.g., routing protocols, summaries of packet flows are not enough. The update information stored in a packet of such a protocol need to be faithfully delivered to its destination. In addition, these packets need to be delivered in the correct time-stamp order. These two requirements are necessary to preserve network dynamics in the simulation. Hence, new synchronization mechanism is needed in Genesis to satisfy these two requirements.

We developed an event-level synchronization mechanism which can work within the framework of Genesis. This work was done using our previous development of Genesis based on SSFNet [6], which was reported in [9]. Particularly, we modified Genesis to support the simulation of BGP protocol, in addition of TCP and UDP protocols. In Genesis, when we simulate a network running BGP protocol for inter-AS (Autonomous System) routing, with background TCP or UDP traffics, we decompose the network 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 new synchronization mechanism in Genesis guarantees that these messages will be delivered in the correct time-stamp order.

3 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 traffics, 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 a farmer/agent framework, and are distributed there to the BGP routers which are the destinations of these messages. The proxy BGP router and BGP agent framework are shown in Figure 2.

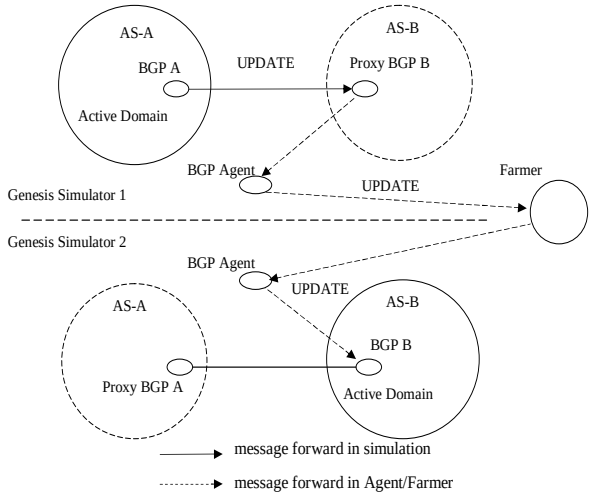

Figure 2. Proxy BGP Routers and BGP Agents

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.

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 the once seen at 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 3 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.

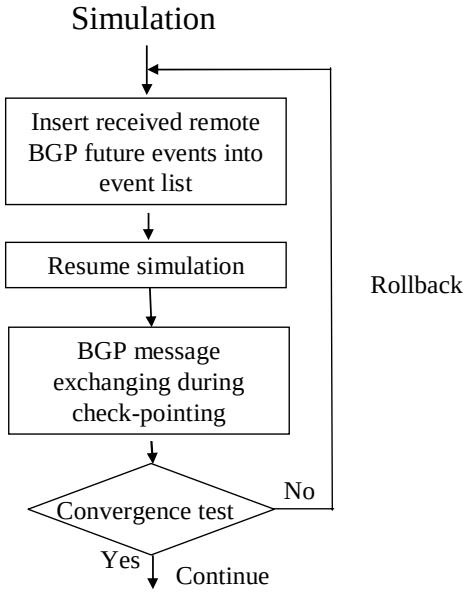

Figure 3. Synchronization for BGP Update Messages

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. The scalability of different network simulators was studied in [4]. 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 the version reported in [7], Genesis did not distribute network information among domain simulators. Each of them had to construct the full network and to store all the dynamic information (e.g., routing information) for the whole network during the simulation. To avoid such replication of memory, we developed a new version of Genesis which 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 SSFNet simulating the same network), as shown in section 6.3.

Memory distribution is particularly challenging in Genesis, because of its special high granularity synchronization approach. In Genesis, within one time interval, 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 [7]. 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 a flow source or its destination, 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", and the parallel simulators do not need to simulate global flows independently.

As a solution, in each domain we introduced 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 4.

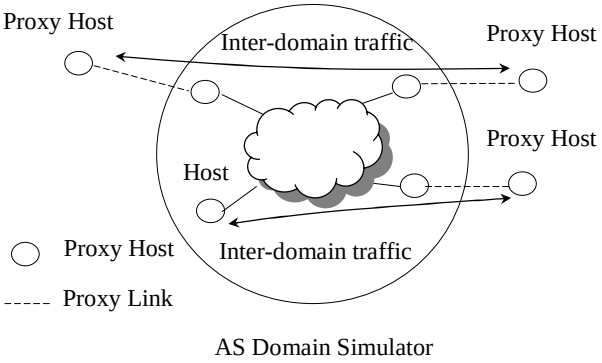

Figure 4. Proxy Hosts and Inter-domain Traffic

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.

5 Distributed Memory Simulation Design

To support distributed memory simulation in Genesis, changes were made to both DML definition and SSFNet based implementation.

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/Interface address mapping, which translates local addresses to and from global addresses used in our BGP update messages.

***Remote host, traffic* and *link*:** Those definitions were added to the current DML definitions for SSFNet [6]. *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 extends SSFNet to allow 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.

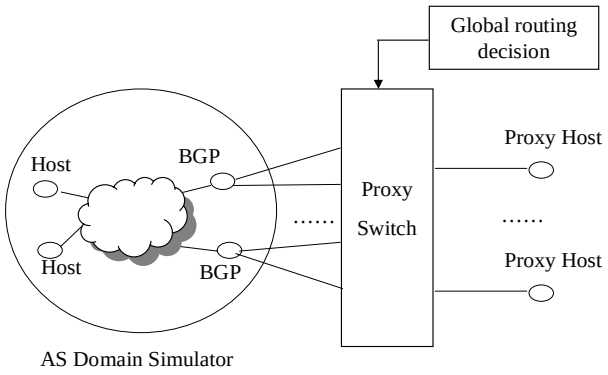

Figure 5. Remote Traffic Path Construction with Proxy Switch

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 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.

6. Performance Evaluation

6.1. Simulation Model

To test the performance and scalability of the Genesis extension described here 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 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 6. 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.

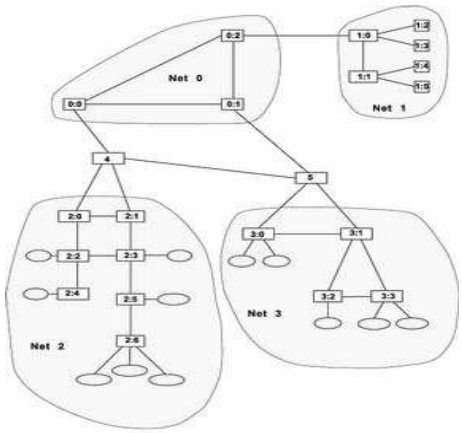

Figure 6. One campus network

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. 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 100Mbit Ethernet. In the simulations under distributed Genesis, the number of processors used was equal to the number of campus networks.

6.2. Synchronization Convergence on BGP Bursts

In BGP network simulations, the first BGP update message burst happens when AS domains need to exchange BGP information to set up the global inter-AS routing. In Genesis, AS domains are constructed distributively, and BGP update messages are synchronized by re-iterating over one time interval until the BGP messages exchanged among domains converge (no more changes) on that interval, as we showed in Figure 3. We measured the convergence of the synchronization in Genesis for BGP networks with 10, 15 and 20 AS domains. Table 1 shows the number of re-iterations needed in Genesis to converge during the BGP message burst, and the number of BGP messages exchanged by each AS simulator. It should be noted that the number of needed iterations grows sublinearly with the number of domains, so slower then the number of messages, which are proportional to the number of domains.

No. of AS	Iterations	Messages
10	7	22
15	8	36
20	12	44

Table 1. BGP synchronization convergence

The results show that the number of re-iterations is related to the size of the network, and is proportional to the longest distance (measured by number of intermediate AS domains) between two AS domains. This distance decides how many re-iterations are needed for BGP decisions from one AS domain to reach all the other AS domains in the network. In our ring-based topology, this distance is related to the ring size.

Although larger BGP network will require more re-iterations on BGP bursts in Genesis, our approach supports scalability in terms of required memory size. In networks in which BGP bursts are not frequent compared to the changes in the background traffic, this approach also assures good runtime performance thanks to high granularity synchronization of the background traffic.

6.3. Simulation Performance

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 256Mb of memory. As shown in Figure 7, when the number of AS domains increases, unlike SSFNet, the memory usage of one Genesis AS simulator does not increase. As a result, by utilizing more computers with smaller memories, we are able to simulate much larger networks.

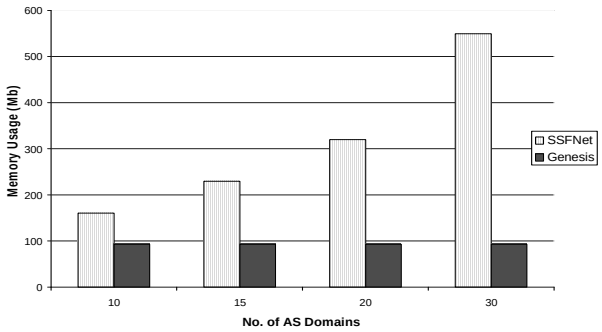

Figure 7. Memory usage of SSFNet and Genesis for simulating different sizes of BGP networks. Memory sizes of Genesis shown above are the requirements for single AS simulator

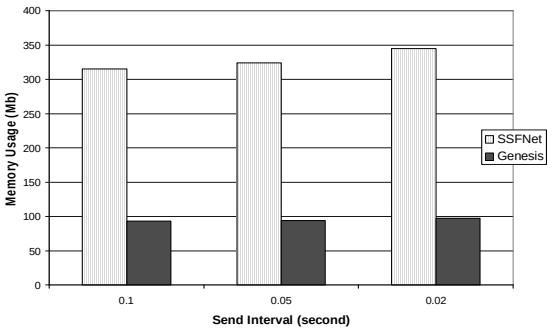

Figure 8. Memory usage of SSFNet and Genesis for 20-AS BGP network simulations with different send-rates

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 8, 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 in [7, 8], Genesis achieved execution speedups thanks to its high granularity synchronization mechanism. In the described new version of Genesis, 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 9 shows the speedups of simulations for these networks.

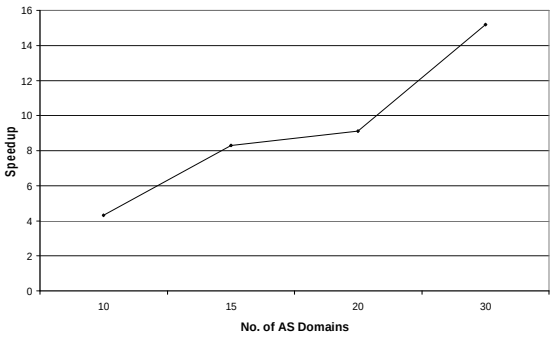

Figure 9. Speedup achieved for simulations of different BGP network sizes

Figure 10 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 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.

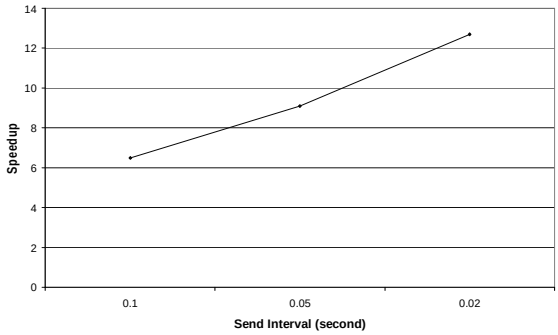

Figure 10. Speedup achieved for 20-AS BGP network simulations with different send-rates

6.4. Impacts of On-going BGP Activities

We have shown the synchronization convergence on BGP bursts under Genesis in section 6.2. When a BGP update message propagation happens, Genesis re-iterates over the same time interval until the propagation converges. Each re-iteration consumes simulation run time. When BGP update message propagations happen periodically during the simulation, the additional run time required by these re-iterations will decrease the speedups achieved by utilizing parallel simulation. An interesting question is how such on-going BGP activities would affect the simulation performance.

To investigate this question, we introduced BGP session crashes into our experiments in which the simulation time was fixed at 400 seconds for 20 AS's and, correspondingly 20 Sun 10 Ultrasparc workstations. The BGP session between two neighboring AS domains, campus network 3 and 4, crashed every D seconds, each time staying down for $D/2$ seconds, and then coming back and staying alive for another $D/2$ seconds. We varied the value of D from 80 to 60, 40 and then 20 seconds. We also used different lengths of iteration time intervals for Genesis check-pointing, denoted as T, which was set to 20, 10 and 5 seconds. Figure 11 shows the speedups achieved in these experiments.

As expected, in the simulation time intervals in which the specified BGP session went down, BGP update messages were propagated causing the broken routes to be withdrawn and back up routes being set up. Accordingly, data packet flows also changed and used the new routes. When that BGP session came back again, BGP update messages propagated again and re-established the broken routes. In either case, the relevant time interval had to be re-iterated again and again until it converged. So these intervals were "slow-down" periods. The time intervals with no BGP propagations were "speed-up" periods thanks to parallel simulation mechanism used by Genesis, as discussed in the previous

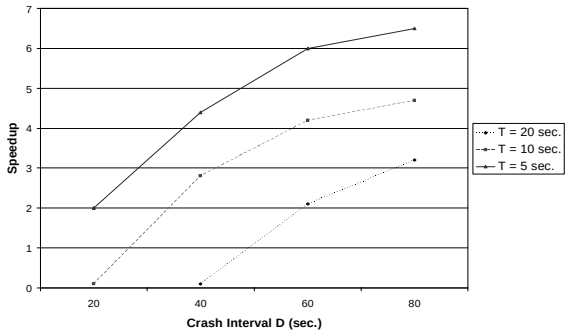

Figure 11. Speedup achieved with BGP crashes run on 20 processors. D denotes crash interval length. T stands for checkpoint interval length

section. As a result, the proportion of the "slow-down" time in the whole simulation time affects the overall speedup: the less frequently the crashes happen, the greater the speedup that can be achieved. On the other hand, the length of the time interval also affects the total re-iteration time. Ideally, the smaller the length of the time interval, the shorter the total re-iteration time. But there are two other factors which benefit from longer intervals. First, small interval length increases the synchronization and check-pointing overheads between intervals and can overwhelm the simulation speedup. Second, if the interval length is too small to cover the BGP propagation period, then the next time interval will need to be re-iterated in addition to the current one.

In our experiments, we were able to reduce the iteration time interval length to 5 seconds, as we observed that the BGP propagation in our experiment scenario was around 3 seconds. As shown in Figure 11, we achieved significant speedups for crash intervals greater than 40 seconds. Besides the crash frequency, the iteration length also played an important role in the performance. When using big iteration interval length of 20 seconds, Genesis failed to produce any speedup with short crash interval of 40 seconds. These results indicate that a method to automatically decide the optimal iteration length for a given simulation scenario could be a valuable future extension that can improve the overall performance of the system.

7. Future Work

In the paper, we demonstrated that the described here new Genesis version can work efficiently with fully distributed network memory. This design reduces and makes scalable the memory size requirement for large-scale network simulations. As a result, Genesis is able to simulate huge networks using limited computer resources. One direction of the future work in this ares is to apply Genesis to more real-world applications, to construct more practical network models and to simulate and analyze them in Genesis.

With the new support of BGP simulation in Genesis, study of the performance and stability of BGP can be another direction of future research. The memory size required by BGP network simulation increases very fast when the number of BGP routers and AS domains increases. As a result, the simulation of large BGP networks was hindered by the memory size limitation. Genesis offers a new approach to simulating BGP on distributed memory that is scalable both in terms of simulation time and the required memory.

References

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

[2] Fujimoto, R.M. Parallel discrete event simulation. *Comm. of the ACM*, 33:31–53, Oct. 1990.

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

[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] 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.

[8] 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.

[9] 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.

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