Distributed Simulation with MPI in *ns-3*

Joshua Pelkey and George Riley
Department of Electrical and Computer Engineering
Georgia Institute of Technology, Atlanta, GA. USA
{jpelkey,riley}@gatech.edu

ABSTRACT

While small topology simulations are important for validation and educational purposes, large-scale network simulations are a fundamental part of active networking research. Therefore, it is important for a network simulator to provide scalable and efficient solutions to execute these types of scenarios. Using standard sequential simulation techniques, large-scale topologies with substantial network traffic will require lengthy simulation execution times and a considerable amount of computer memory. Often, these execution times are too long for a networking researcher to run multiple simulations and collect significant data. Parallel and distributed simulation is one method that allows researchers to efficiently simulate these large topologies by distributing a single simulation program over multiple interconnected processors. To enable this scalable simulation methodology in *ns*-3, we formally present our distributed simulator, introduced in *ns*-3.8. This simulator uses the Message Passing Interface (MPI) standard and a conservative lookahead mechanism. Using the distributed simulator, we conducted a performance study using a large-scale point-to-point campus network scenario with a variable number of nodes distributed across several interconnected processors within a computer cluster. We show near-optimal improvements in simulation execution time are possible using the distributed simulator in *ns*-3.[1]

Categories and Subject Descriptors

I.6.8 [**Simulation and Modeling**]: Types of Simulation – discrete event, distributed, parallel

General Terms

Performance, Design, Experimentation

[1]This work supported in part by the U. S. National Science Foundation grant number 0958015

WNS-3 2011, March 21, Barcelona, Spain

DOI 10.4108/icst.simutools.2011.245585

Keywords

Parallel and distributed discrete event network simulation, scalability

1. INTRODUCTION

An important part of network simulation research includes the simulation of large-scale networks coupled with substantial network traffic. With standard sequential simulation techniques, these simulations can require lengthy execution times or may exceed available computer memory and fail to run completely. In either case, simulation is rendered ineffective. To run these large-scale simulations efficiently, parallel and distributed simulation techniques can be used. Earlier discrete event network simulators, such as Parallel/Distributed *ns* (pdns) and the Georgia Tech Network Simulator (GTNetS) [2, 7], have implemented distributed simulation. By dividing a single simulation topology into distinct pieces and distributing these pieces across multiple interconnected processors, the simulator can execute different parts of the network in parallel during runtime. Along with a speedup in simulation execution time through parallel simulation, distributing a simulation across multiple nodes provides an increase in computer resources, specifically main memory, and enables larger topologies to be simulated.

It is mandatory that simulations using distributed techniques produce the same results as identical sequential simulations. To ensure proper distributed simulation execution in *ns*-3, we have implemented an *ns*-3 interface with the MPI standard for message passing and time synchronization using a conservative lookahead algorithm. Our MPI interface in *ns*-3 allows time-stamped messages to be sent and received asynchronously between processes. The implementation of a lookahead algorithm ensures that simulation events are processed in time-stamped order. Failure to require sequential event ordering may lead to causality errors and, ultimately, incorrect simulation results.

To demonstrate distributed simulation in *ns*-3, we have conducted a performance study using a large point-to-point campus network topology that allows an arbitrary number of nodes and applications to be created and simulated by connecting together multiple campus networks [1]. By varying the number of nodes and applications and distributing this topology across a variable number of processors, we have measured significant improvements in simulation execution times using the distributed simulator.

In section 2, we will describe the *ns*-3 distributed simulation implementation. The MPI interface and the lookahead algorithm will be presented, along with a brief description of

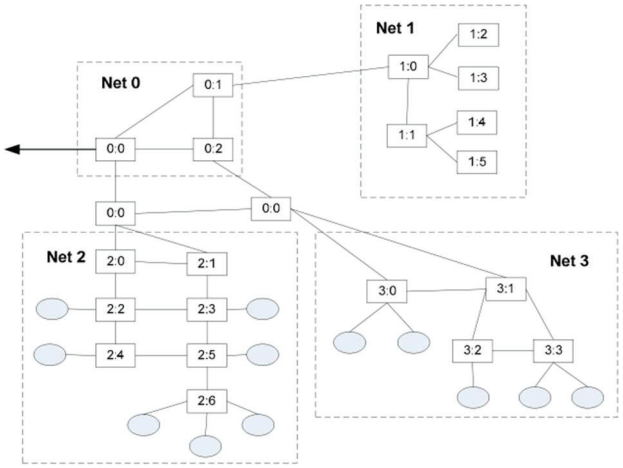

Figure 1: A campus network topology block. Multiple blocks are connected together to create large-scale point-to-point topologies in *ns*-3.

common terminology for parallel and distributed simulation systems. Section 3 will present the campus network simulation used to test the distributed simulator, along with the results from our performance study. Overall performance will be analyzed for distributed simulation in *ns*-3. Finally, section 4 will present our conclusions and future work.

2. IMPLEMENTATION

Parallel and distributed discrete event network simulation can be implemented as a set of sequential simulations, each executing a different part of the simulated network [3]. An individual sequential simulation is referred to as a logical process (LP), and the unique number for each LP is referred to as the *rank*. Potentially, each LP can be placed on a different processor for execution. During the course of a distributed simulation, LPs may need to be notified of events occurring in different LPs. For networking simulation this occurs often, as packets in one LP are destined for a different LP. Each time a packet needs to change LPs, or cross simulator boundaries, a time-stamped message must be sent between LPs.

For distributed simulation in *ns*-3, we use MPI and a remote point-to-point channel to handle simulator boundaries and message passing. During the setup of a network topology, individual nodes are assigned numbers to indicate their LP. As point-to-point links are created, if a link between two nodes with differing LPs is encountered, a remote point-to-point channel is created between them. The transmit method for a remote point-to-point channel calls the MPI send operation, using our MPI interface in *ns*-3. Thus, any packet destined for a different LP will cross a remote point-to-point channel via a time-stamped MPI message, and the receiving LP can unwrap this message to obtain the packet and proceed with its simulation.

Along with message passing, ensuring events are processed in time-stamped order is a fundamental part of distributed simulation. Without strict sequential event ordering at all LPs, distributed simulation will fail to produce identical results to sequential simulation. To ensure sequential ordering, several synchronization algorithms exist for distributed simulation [5]. We have chosen a conservative synchronization algorithm in *ns*-3. Conservative synchronization guarantees that events will be executed at an LP only if it is certain that no new events with a smaller timestamp will be received at that LP. To accomplish this, conservative synchronization is dependent upon predetermining a value known as *lookahead*. In the simulation model, the lookahead value is the minimum amount of time that must elapse before any action at LP-A can possibly affect anything at LP-B. In the case of network simulations, a packet transmission at router A cannot possibly be known to router B until at least the speed-of-light delay on the wire connecting routers A and B. The lookahead value is used by each LP as the longest time-window allowed for local event processing before synchronizing with the remaining LPs. Events outside of this lookahead window are blocked until synchronization occurs. Subsequently, the lookahead window advances, and LPs are free to execute events within this new window. By carefully choosing a lookahead value, conservative synchronization can guarantee strict sequential event ordering. In *ns*-3, after topology creation, each remote point-to-point link is visited and the time delay for that link is noted. The smallest time delay value for all remote point-to-point links is used as our lookahead value.

It is important to note a few limitations of the current implementation of distributed simulation in *ns*-3. First, pure distributed wireless simulation is not currently possible. Simulator boundaries must be created along point-to-point links; therefore, a distributed simulation in *ns*-3 requires at least one point-to point link within the topology. Second, distributing the topology must be done manually by the user. Individual nodes must be assigned rank during the topology setup, and the user must determine which nodes should be placed on which LPs for efficient distributed simulation. Finally, all nodes in the topology are created on each LP, regardless of their rank. Distributing the simulation is instead handled at the application level. The user should only install applications on nodes when their rank matches the given LP.

3. CAMPUS NETWORK SIMULATIONS

A performance study was conducted to demonstrate distributed simulation in *ns*-3, using the *ns*-3.9 release and slightly modified versions of the nms-p2p-nix-distributed example provided in that release. The modification was simply a change in the runtime of the simulation to decrease the amount of wall-clock time needed per simulation. The point-to-point campus network "building block", shown in figure 1, was used to create large-scale topologies by connecting together multiple campus networks. Each campus network block consists of 4 servers in net 1, as well as 30 routers and a variable number of LAN node clients on nets 2 and 3. The number of clients was varied between 2 and 40 nodes per LAN; therefore, a 2 campus network topology with 2 nodes per LAN contains a total of 116 nodes, while a 10 campus network topology with 40 nodes per LAN contains a total of 5140 nodes. LAN nodes are connected to their respective LAN router using point-to-point links with a bandwidth of 100 Mb/s and a 1 ms delay. Campus network blocks are connected together with a 2 Gb/s point-to-point link and a variable delay. The delay was varied from 200 ms to 10 μs to change the lookahead during our study. Multi-

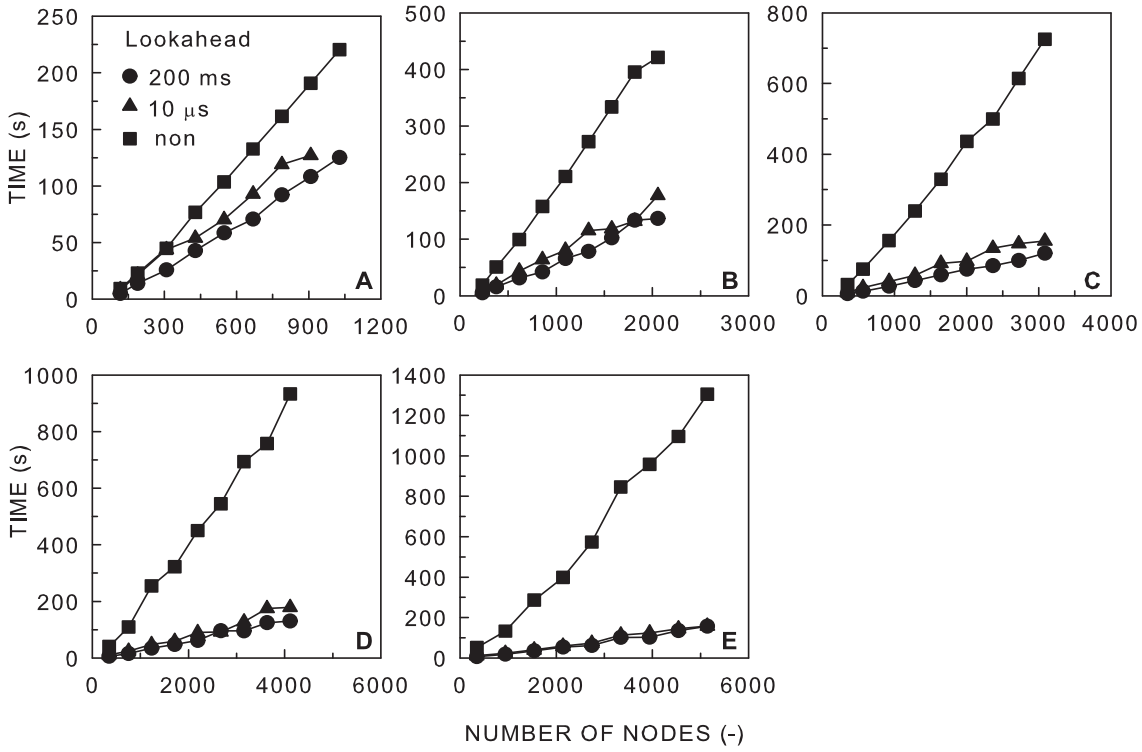

Figure 2: Campus network (CN) simulations on the Georgia Tech Hogwarts cluster with (A) 2 CNs (B) 4 CNs (C) 6 CNs (D) 8 CNs (E) 10 CNs.

ple on-off applications were placed on the servers in net 2 of each campus network. These applications sent traffic to every LAN node in nets 1 and 3 of a neighboring campus network. To distribute the topology, individual campus networks were placed on separate LPs. This topology provided a large number of nodes and substantial traffic flow, while ensuring packets also crossed simulator boundaries.

Two different clusters were used for this simulation study, the Georgia Tech Ferrari and Hogwarts clusters. The Hogwarts cluster consists of 6 nodes, each with 2 quad-core processors and 48GB of RAM. The Ferrari cluster consists of a mix of machines including 3 quad-core nodes and 8 dual-core nodes. For distributed simulation each campus network was placed on a different LP and executed by a different processor. For sequential simulations, the same clusters were used; however, only one processor was used to run the simulation. For this performance study, the number of campus networks was varied between 2, 4, 6, 8, and 10. Finally, every simulation was compared using distributed and non-distributed techniques, and each simulation run was completed three times in order to take an average for the total execution time.

3.1 Distributed vs. Non-distributed

Distributed and non-distributed performances of the campus network simulations on the Hogwarts and Ferrari clusters were investigated (figures 2 and 3). To compare distributed simulations against sequential simulations, a lookahead value of 200 ms was chosen. Similar lookahead values were tested, including 100 ms, 50 ms, and 10 ms, with similar results (not shown).

Distributed simulations run on the Hogwarts cluster were significantly faster than identical sequential simulations. For a 2 campus network simulation with 1028 nodes, execution times were 125 and 220 seconds for distributed (200 ms lookahead) and non-distributed runs, respectively (figure 2A). The maximum number of nodes for a 10 campus network simulation was 5140 with execution times of 156 and 1304 seconds for distributed (200 ms lookahead) and non-distributed runs, respectively (figure 2E). In addition, nearly optimum linear speedup was seen across all campus network simulations on Hogwarts. The 2, 4, 6, 8, and 10 campus network simulations ran on average 1.8, 3.3, 5.8, 6.9, and 8.3 times faster than the non-distributed simulations, respectively (figure 4A).

Using the Ferrari clusters, the distributed simulations were faster than identical sequential simulations[2]. For a 2 campus network simulation with 1028 nodes, execution times were 50 and 94 seconds for distributed (200 ms lookahead) and non-distributed runs, respectively (figure 3A). The maximum number of nodes for a 10 campus network simulation was 5140 with execution times of 322 and 712 seconds for distributed (200 ms lookahead) and non-distributed runs, respectively (figure 3E). Linear speedup was only observed for the 2 campus network simulation runs. The average speedup values for 2, 4, 6, 8, and 10 campus network simulations were 1.9, 1.6, 2.0, 2.3, and 2.4 times faster than the non-distributed simulations, respectively (figure 4B).

[2]It is important to note that the simulation scripts used on Ferrari and Hogwarts differed slightly; therefore, the execution times on the two clusters can not be compared directly.

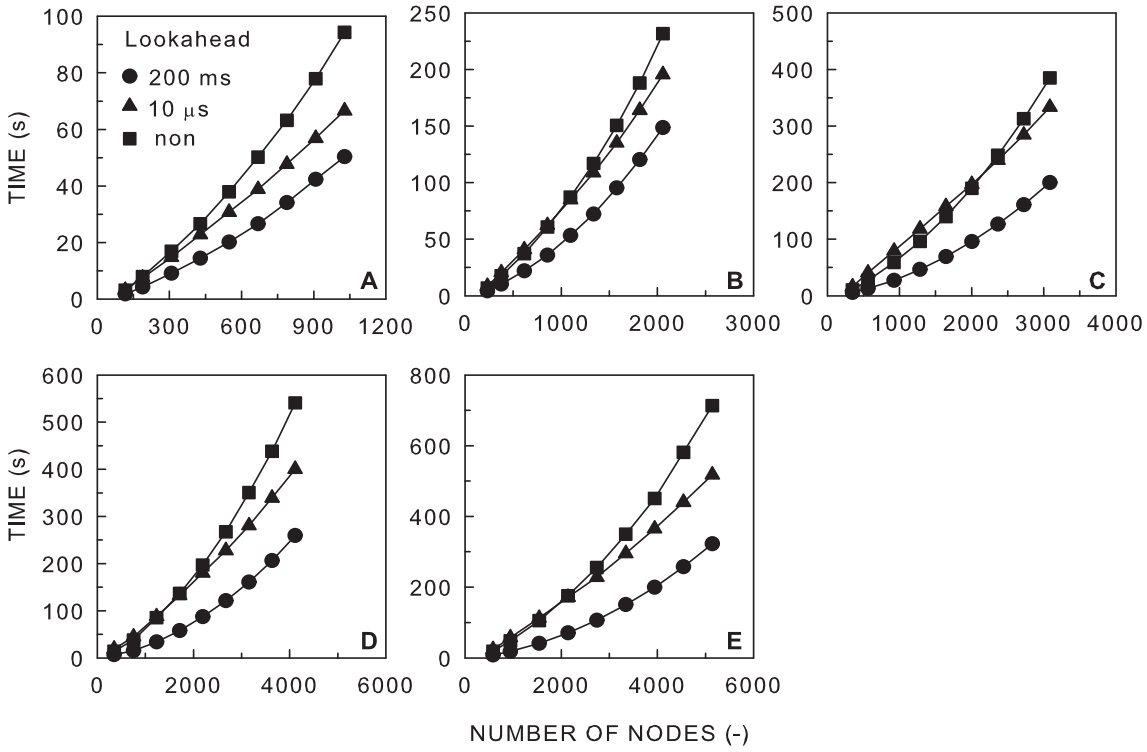

Figure 3: Campus network (CN) simulations on the Georgia Tech Ferrari cluster with (A) 2 CNs (B) 4 CNs (C) 6 CNs (D) 8 CNs (E) 10 CNs.

Although the distributed simulations on Ferrari were faster than the non-distributed runs, the overall speedup on Ferrari was far less effective than on Hogwarts across all campus networks. After noting this trend, we discovered the Ferrari system consisted of a mix of machines, with the first three nodes containing significantly faster hardware. Thus, upon reaching four or more campus networks, the slower machines were introduced into the simulation. Ultimately, we could have altered the Ferrari runs to use only the three fastest nodes, and it is expected that this would result in near-linear speedup across all campus network simulations.

3.2 Changing the Lookahead

Along with comparing simulation execution times against sequential and distributed simulations, we also compared purely distributed simulation techniques with varying lookahead values. When changing the lookahead value from 200 ms to 10 μs, the simulation time increased when using both Hogwarts and Ferrari. For the Hogwarts simulations, the 10 μs lookahead runs executed an average of 25% slower compared to the 200 ms lookahead simulations (figure 2). For the Ferrari simulations, a 47% decrease was observed for the 10 μs runs when compared to the 200 ms distributed simulations (figure 3). With a smaller lookahead time, the simulators must synchronize with greater frequency. In fact, for the 10 μs lookahead simulations, the simulators must synchronize a factor of 20,000 times more often than the simulations with a lookahead of 200 ms.

Such low lookahead values are common in wireless networking simulations due to the small distances between nodes and short propagation delays; thus, distributed simulation for wireless topologies presents a challenging problem. Although distributing the wireless topology continues to offer gains in computer memory, the significant decrease in lookahead for standard conservative distributed techniques greatly decreases the achievable speedup. In order to efficiently simulate large-scale wireless topologies, other methods such as event reduction and event bundling [4, 6] should be implemented to augment the performance of wireless distributed simulation. Without implementing such additional techniques, larger lookahead values are crucial for efficient distributed network simulation in *ns*-3.

4. CONCLUSIONS AND FUTURE WORK

In this paper, we have formally presented our distributed simulator, released in *ns*-3.8, and described its implementation details. Specifically, the MPI interface, remote point-to-point channel, and the conservative synchronization algorithm were discussed. Using this distributed simulator, a performance study using a large-scale point-to-point campus network topology was presented. The study showed nearly optimal linear speedup is achievable using the distributed simulator in *ns*-3. Furthermore, the study confirmed that speedup was greatly affected by the lookahead value used, as expected for any distributed simulation using the conservative synchronization technique.

For future work, several additions and improvements will be examined. First, an MPI helper class could be designed to facilitate creating distributed topologies. Rather than a user assigning rank to each node individually, nodes could be assigned rank automatically by examining the topology

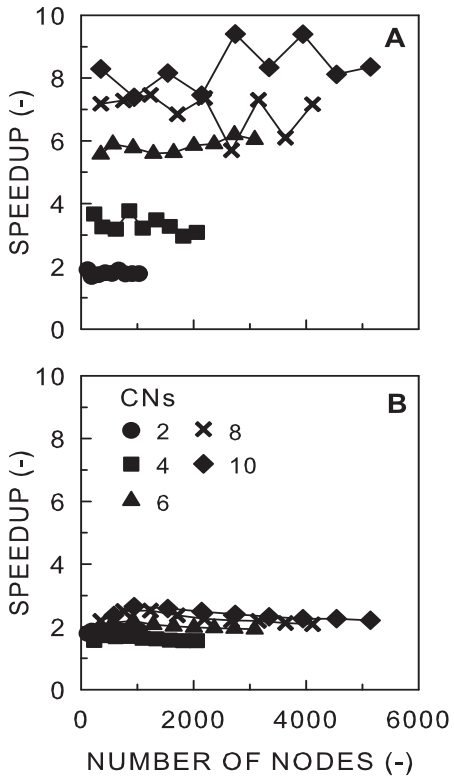

Figure 4: Speedup using distributed simulation for campus network topologies on the (A) Hogwarts cluster (B) Ferrari cluster.

and determining the optimal simulator boundary positions. With this addition, existing simulation scripts could be distributed with relative ease. Second, distributing the topology could occur at the node level, rather than the application level. Distributing at the node level would greatly reduce memory consumption and lead to a speedup in topology generation. Finally, pure distributed wireless is a commonly requested feature and will be added to the distributed simulator in *ns*-3.

5. REFERENCES

[1] Standard baseline NMS challenge topology. `http://www.ssfnet.org/Exchange/gallery/baseline`, July 2002.

[2] PDNS - Parallel/Distributed *ns*. `http://www.cc.gatech.edu/computing/compass/pdns`, March 2004.

[3] R. M. Fujimoto. Parallel simulation: parallel and distributed simulation systems. In *Proceedings of the 33nd conference on Winter simulation*, WSC '01, pages 147–157, Washington, DC, USA, 2001. IEEE Computer Society.

[4] Z. Ji, J. Zhou, M. Takai, J. Martin, and R. Bagrodia. Optimizing parallel execution of detailed wireless network simulation. In *Proceedings of the eighteenth workshop on Parallel and distributed simulation*, PADS '04, pages 162–169, New York, NY, USA, 2004. ACM.

[5] K. S. Perumalla. Parallel and distributed simulation: traditional techniques and recent advances. In *Proceedings of the 38th conference on Winter simulation*, WSC '06, pages 84–95. Winter Simulation Conference, 2006.

[6] P. Peschlow, A. Voss, and P. Martini. Good news for parallel wireless network simulations. In *Proceedings of the 12th ACM international conference on Modeling, analysis and simulation of wireless and mobile systems*, MSWiM '09, pages 134–142, New York, NY, USA, 2009. ACM.

[7] G. F. Riley. The Georgia Tech Network Simulator. In *Proceedings of the ACM SIGCOMM workshop on Models, methods and tools for reproducible network research*, MoMeTools '03, pages 5–12, New York, NY, USA, 2003. ACM.