COST AND ACCURACY OF PACKET-LEVEL VS. ANALYTICAL NETWORK SIMULATIONS: AN EMPIRICAL STUDY

A THESIS SUBMITTED TO THE GRADUATE DIVISION OF THE UNIVERSITY OF HAWAI'I IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE IN COMPUTER SCIENCE

MAY 2007

By
Kayo Fujiwara

Thesis Committee:
Henri Casanova, Chairperson
Kim Binsted
Philip Johnson
We certify that we have read this thesis and that, in our opinion, it is satisfactory in scope and quality as a thesis for the degree of Master of Science in Computer Science.

THESIS COMMITTEE

[Signatures and names]

Chairperson

KIM BINSTED

Philip Johnson
Copyright 2007

by

Kayo Fujiwara

iii
Acknowledgements

I am grateful to my advisor, Professor Henri Casanova, for his guidance and support. This work would not be possible without his help. I would also like to thank my committee members, Professors Kim Binsted and Philip Johnson, for their suggestions and advise. Discussions with Dr. Arnaud Legrand and other SIMGRID users/developers were very helpful. Dr. Legrand's help with SimGRID code is also greatly appreciated.
ABSTRACT

Large-scale distributed computing platforms have become widely used due to improvements in hardware and software. To evaluate algorithms for those platforms, one usually conducts experiments in simulations. While simulation has been employed in various areas such as microprocessor design and network protocol design, simulation techniques for large-scale distributed applications are still in their infancy.

In this thesis, we focus on network simulation, which is one of the most important components of simulation for large-scale distributed applications. We first investigate the trade-offs between simulation cost and accuracy for two commonly used network simulation methods, packet-level and analytical simulations. We do this by comparing three existing packet-level network simulation tools, GTNetS, SSFNet, and ns-2, with the analytical simulation models implemented in SIMGRID, a toolkit for simulating Grid applications. We find that the following three factors affect the accuracy of SIMGRID when compared to packet-level simulation: (1) the size of the data transferred by TCP flows, (2) the complexity of the network topology, and (3) the network contention. Our results provide a new understanding of the relative merit of analytical network simulation versus packet-level simulation.

We then integrate GTNetS into SIMGRID, in order to enable the SIMGRID users to switch between packet-level and analytical network simulations and make their own decision regarding the cost/accuracy trade-offs. The two main challenges are the translation between simulation objects in the two simulators, and the addition of suspendable/resumable simulation and of speculative simulation capabilities to GTNetS. We address the latter by modifying the GTNetS source code, and address the former by implementing an API on top of the modified GTNetS. The end result for SIMGRID users is a new ability to switch easily (i.e., with no user code modification) between analytical and packet-level simulations.
Contents

Acknowledgements iv
Abstract v
List of Tables viii
List of Figures ix
1 Introduction 1

2 Related Work 5
  2.1 Simulators ................................ 5
     2.1.1 SIMGRID ................................ 5
     2.1.2 Bricks ................................ 6
     2.1.3 GridSim ................................ 6
     2.1.4 MicroGrid ............................... 6
  2.2 Comparison between pacclret-Ievel and analytical simulations ........ 7

3 Network simulation 8
  3.1 Packet-level simulation .................................. 8
     3.1.1 Popular packet-level simulators ................ 8
  3.2 Analytical modeling ...................................... 9
     3.2.1 TCP Slow start and congestion avoidance .......... 10
     3.2.2 Modeling TCP throughput ........................ 10
     3.2.3 Bandwidth sharing ................................ 11
  3.3 Network simulation in SIMGRID ............................ 13
     3.3.1 Modeling a single flow ............................ 13
     3.3.2 Bandwidth-Sharing Model .......................... 14
     3.3.3 Limitations of the network model in SIMGRID .... 14

4 Experimental Evaluation 16
  4.1 A single network link .................................. 16
     4.1.1 Experimental design ............................. 16
     4.1.2 Results ..................................... 17
     4.1.3 Single network link, multiple flows ............... 22
  4.2 Dogbone topology ..................................... 22
     4.2.1 Experimental design ............................. 22
4.2.2 Results ........................................... 23
4.3 Random topology ..................................... 26
  4.3.1 Experimental design .......................... 26
  4.3.2 Results for $S = 10^5$ KB .................. 29
  4.3.3 Results for $S = 100$ KB ................... 35
  4.3.4 Cost ........................................... 35
4.4 Discussion ........................................... 38

5 Integration of SimGrid and GTNetS .............. 40
  5.1 Introduction and Motivation ..................... 40
  5.2 Overview of GTNetS and SimGrid ............... 40
    5.2.1 SimGrid .................................... 40
    5.2.2 GTNetS .................................... 42
  5.3 Design and implementation ........................ 43
    5.3.1 Object translation ........................ 43
    5.3.2 Interrupting and resuming GTNetS .......... 44
    5.3.3 Speculative execution of GTNetS .......... 45
  5.4 Interfaces between SimGrid and GTNetS .......... 45
  5.5 Conclusion ..................................... 46

6 Conclusion ........................................... 47
  6.1 Summary of contributions and impacts .......... 47
  6.2 Future work ..................................... 48

A Single network link ................................... 49

B Single link – different window sizes ............... 55

C Multiple flows on a single link ................... 59

D Dogbone topology .................................... 63

E Random topology .................................... 68

Bibliography ........................................... 71
<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Simulated time and simulation time observed with a packet-level simulator (GTNetS) and an analytical model (SimGrid). The time to transfer 100 MB data is measured for a randomly generated topology with 200 nodes and various numbers of flows. Ratio ( \equiv \frac{\text{simulation time}}{\text{simulated time}} ). The unit of time is seconds.</td>
<td>3</td>
</tr>
<tr>
<td>4.1</td>
<td>Simulated time and simulation time observed with a packet-level simulator (GTNetS) and an analytical model (SimGrid). The time to transfer 10^5 KB data is measured for a randomly generated topology with 200 nodes and various numbers of flows. Ratio ( \equiv \frac{\text{simulation time}}{\text{simulated time}} ). The unit of time is seconds.</td>
<td>37</td>
</tr>
<tr>
<td>5.1</td>
<td>GTNetS layers</td>
<td>42</td>
</tr>
<tr>
<td>5.2</td>
<td>Object translation</td>
<td>44</td>
</tr>
<tr>
<td>5.3</td>
<td>TCP parameters</td>
<td>45</td>
</tr>
</tbody>
</table>
List of Figures

3.1 Linear network with identical links of capacity $C$. ........................................ 11

4.1 Topology for a single flow over a single link. ............................................. 17

4.2 Throughput for ns-2, SSFNet, and GTNetS as a function of bandwidth and latency. The upper panel shows the throughput for data size = $10^8$ KB and the lower panel for data size = 100 KB. For the lower panel, to show the data points where the throughputs are smaller than the mesh, the $z$-axis is reversed. Squares (□) show the throughput with GTNetS; Circles (○) with ns-2; Crosses (×) with SSFNet ($D = 0.2$); and Pluses (+) with SSFNet ($D = 0.01$). ........................................... 18

4.3 The throughput (upper panel) and the relative error in throughput (lower panel) for ns-2, SSFNet ($D = 0.2, 0.01$), GTNetS, and SIMGRID as a function of data size. $\alpha = 10$ ms and $\beta = 10^3$ KB/sec. ........................................... 20

4.4 The throughput (upper panel) and the relative error in throughput (lower panel) for ns-2, SSFNet ($D = 0.2, 0.01$), GTNetS, and SIMGRID as a function of data size. $\alpha = 10$ ms and $\beta = 100$ KB/sec. ........................................... 21

4.5 Dogbone topology. ........................................... 23

4.6 Throughput for SIMGRID and GTNetS as a function of latency. $\beta = 10^8$ KB/sec, $S = 10^5$ KB. The upper panel shows the throughput for flow A and the lower panel for flow B. ........................................... 25

4.7 Relative error in throughput of SIMGRID to GTNetS as a function of latency for $\beta = 10^8$ KB/sec. ........................................... 26

4.8 Throughput for SIMGRID and GTNetS as a function of latency. $\beta = 100$ KB/sec, $S = 10^5$ KB. The upper panel shows the throughput for flow A and the lower panel for flow B. ........................................... 27

4.9 Throughput error of SIMGRID as a function of latency. ........................................... 28

4.10 Relative error in throughput of SIMGRID for each flow. The upper panel shows the error for one of the experiments in case 1, the middle panel for case 2, and the lower panel for case 3. The red dashed lines show $E = \pm 0.1$, and the green lines show $E = \pm 0.5$. ........................................... 30

4.11 Distribution of error as a function of $N_f$, for case 1 (upper), case 2 (middle), and case 3 (lower) for $\gamma = 0.1$. Crosses (×) show the percentages of flows such that $E < -0.01$; Circles (○), $-0.01 \leq E \leq 0.01$; Squares (□), $E > 0.01$. ........................................... 32
4.12 Distribution of error as a function of $N_f$ for case 1 (upper), case 2 (middle), and case 3 (lower) for $\gamma = 0.1$. Crosses $(\times)$ show the percentages of flows such that $E < -0.1$; Circles $(\bigcirc)$, $-0.1 \leq E \leq 0.1$; Squares $(\square)$, $E > 0.1$. 33

4.13 Distribution of error as a function of $\eta$, physical bandwidth divided by the number of flows that share the link on the network, for $\gamma = 0.01$ (upper left), $\gamma = 0.1$ (upper right), and $\gamma = 0.2$ (lower left), and $\gamma = 0.5$ (lower right). 34

4.14 Relative error in throughput of SimGRID for each flow. $S = 100$ KB. The upper panel shows the error for one of the experiments in case 1, the middle panel for case 2, and the lower panel for case 3. 36

5.1 The SimGRID components 41

A.1 Throughput (left panels) and relative error in throughput (right panels) for ns-2, SSFNet ($D = 0.2, 0.01$), GTNetS, and SimGRID as a function of data size for $\beta = 100$ KB/sec. The upper panels show the results for $\alpha = 0$ ms, the middle panels for $\alpha = 1$ ms, and the lower panels for $\alpha = 100$ ms. 50

A.2 Throughput (left panels) and relative error in throughput (right panels) for ns-2, SSFNet ($D = 0.2, 0.01$), GTNetS, and SimGRID as a function of data size for $\beta = 10^3$ KB/sec. The top panels show the results for $\alpha = 0$ ms, the upper middle panels for $\alpha = 1$ ms, the lower middle panels for $\alpha = 10$ ms, and the bottom panels for $\alpha = 100$ ms. 51

A.3 Throughput (left panels) and relative error in throughput (right panels) for ns-2, SSFNet ($D = 0.2, 0.01$), GTNetS, and SimGRID as a function of data size for $\beta = 10^4$ KB/sec. The top panels show the results for $\alpha = 0$ ms, the upper middle panels for $\alpha = 1$ ms, the lower middle panels for $\alpha = 10$ ms, and the bottom panels for $\alpha = 100$ ms. 52

A.4 Throughput (left panels) and relative error in throughput (right panels) for ns-2, SSFNet ($D = 0.2, 0.01$), GTNetS, and SimGRID as a function of data size for $\beta = 10^5$ KB/sec. The upper panels show the results for $\alpha = 0$ ms, the middle panels for $\alpha = 1$ ms, and the lower panels for $\alpha = 100$ ms. 53

A.5 Throughput (left panels) and relative error in throughput (right panels) for ns-2, SSFNet ($D = 0.2, 0.01$), GTNetS, and SimGRID as a function of data size for $\beta = 10^6$ KB/sec. The top panels show the results for $\alpha = 0$ ms, the upper middle panels for $\alpha = 1$ ms, the lower middle panels for $\alpha = 10$ ms, and the bottom panels for $\alpha = 100$ ms. 54

B.1 Throughput for ns-2, SSFNet ($D = 0.2, 0.01$), GTNetS, and SimGRID as a function of data size, for $\beta = 100$ KB/sec (left) and $\beta = 10^5$ KB/sec (right). The TCP advertised window size is 10 KB. When the data size is larger than $10^4$ KB, the throughput with SimGRID approaches the throughput with ns-2 for $\beta = 100$ KB/sec. The throughput with SimGRID approaches the throughput with GTNetS and SSFNet. 55

B.2 Same as Figure B.1, but the window size is 30 KB. The trends of the throughputs are similar to those in Figure B.1. 56
B.3 Throughput with three different TCP advertised window sizes, 10 K, 20 K, and 30 K, for SimGrid (upper panels), ns-2 (middle), GTNetS (lower) as a function of data size. Left panels are for $\beta = 100$ KB and right panels for $\beta = 10^5$ KB. For all the simulators, when $\beta = 100$ KB, the throughput does not depend on the window size. When $\beta = 10^5$ KB (latency bound case), the throughput varies depending on the window size: higher throughput with larger window size. ......................... 57

B.4 Same as Figure B.3, but for SSFNet ($D = 0.2$ (upper panel), 0.01 (lower)). The trend of the throughputs are similar to those in Figure B.3. ........... 58

C.1 Throughput for ns-2, SSFNet, GTNetS, and SimGrid as a function of bandwidth, for the number of flows is 1 (upper left), 2 (upper right), 16 (lower left), and 32 (lower right). $\alpha = 0$ ms and $S = 10^5$ KB. Average throughputs are shown for the experiments with more than one flows. .......... 59

C.2 Same as Figure C.1, but for $\alpha = 1$ ms. .............................. 60

C.3 Same as Figure C.1, but for $\alpha = 10$ ms. .............................. 61

C.4 Same as Figure C.1, but for $\alpha = 100$ ms. .............................. 62

D.1 Throughput for flow A (upper), flow B (middle), and error in throughput (lower) for $\beta = 10$ KB/sec. ........................................... 64

D.2 Throughput for flow A (upper), flow B (middle), and error in throughput (lower) for $\beta = 10^3$ KB/sec. ................................. 65

D.3 Throughput for flow A (upper), flow B (middle), and error in throughput (lower) for $\beta = 10^4$ KB/sec. ................................. 66

D.4 Throughput for flow A (upper), flow B (middle), and error in throughput (lower) for $\beta = 10^5$ KB/sec. ................................. 67

E.1 Relative error in throughput of SimGrid for $S = 10^5$ KB. The left panels show the error for $N_f = 200$, and the right panels for $N_f = 100$. The upper panels show the error for case 1, the middle panels for case 2, and the lower panels for case 3. Each curve corresponds to one experiment. Each panel includes ten curves. ........................................ 69

E.2 Relative error in throughput of SimGrid for $S = 100$ KB. The left panels show the error for $N_f = 200$, and the right panels for $N_f = 100$. The upper panels show the error for case 1, the middle panels for case 2, and the lower panels for case 3. Each curve corresponds to one experiment. Each panel includes ten curves. ........................................ 70
Chapter 1

Introduction

Large-scale distributed computing platforms have become widely used due to improvements in hardware (better networks) and software (better middleware). Applications on these large-scale platforms have rapidly emerged such as distributed file sharing [17, 27], distributed volunteer computing [2, 3, 32], or distributed gaming [4, 24]. One particularly active domain among them is grid computing [13], which is the use of network-connected compute and storage resources across institutions to support large scientific user communities. In all these domains, one must develop efficient and scalable distributed algorithms, both at the user application level and at the middleware level. Examples include resource discovery algorithms, resource selection and application scheduling algorithms, resource management algorithms, overlay network protocols, data dissemination algorithms, search algorithms, and publisher/subscriber algorithms.

It is challenging to design, develop, and deploy distributed algorithms that are efficient and scalable. There are three methods to evaluate candidate algorithms: develop abstract analytical models of the whole system so that one can reason about it; conduct experiments in real-world environments; and conduct experiments in simulations.

Analytical models are perhaps the most attractive as they allow formal reasoning about performance and scalability. Unfortunately such models often make stringent assumptions that do not hold in real-world large-scale distributed systems. The second method, i.e., real-world experimentation, has its own set of difficulties. First, it is difficult to set up experimental environments that span a wide range of scenarios. One may not be able to use a large number of distinct platforms, or be allowed to set up extensive experimental scenarios on them. Second, experimental results are often non-repeatable. The availability of the resources may vary because of loads by other users or resource failures. Third, con-
ducting experiments may be very time consuming and labor intensive. Consequently, one typically cannot run sufficiently large numbers of experiments to evaluate many candidate algorithms over a wide range of platform configurations.

Because of these difficulties, one must resort to simulation to study distributed algorithms and applications. Although simulation has been employed in key areas of computer science such as computer architecture and microprocessor design, or networking protocol design, simulation techniques for large-scale distributed applications are still in their infancy. An important question for simulation is the trade-offs between simulation accuracy (i.e., how close is the simulation to the real-world) and simulation cost (i.e., how much time is needed to run the simulation). An ideal simulator would have high accuracy and low cost.

Network simulation approaches and their limitations

One of the key domains of simulation for large-scale distributed applications is network simulation. A well-known approach for network simulation is packet-level simulation (ns-2 [26], GTNetS [31], SSFNet [9]). This approach in principle achieves high accuracy as it simulates every individual packet going through IP network links and routers. The problem is that it is time-consuming. For instance, consider a realistic network topology with 200 nodes and a number of TCP connections, or "flows," between randomly picked pairs of nodes. Each flow sends 100 MB of data. The left side of Table 1.1 shows the simulated time and simulation time for various numbers of flows observed with a popular packet-level simulator (GTNetS) as well as their ratio. (See Chapter 4 for details of these experiments.) The ratio is defined as simulation time divided by simulated time, and thus, a high ratio means a high simulation cost. As seen in the table, packet-level simulation can take more time than real-world experiments. While this is perfectly acceptable when performing experiments to study network protocols for instance, it is impractical for evaluating large-scale distributed applications, which typically requires large numbers of long-running experiments.

Another approach is to develop analytical models that do not consider individual packets but instead consider each flow as a whole and reason about the bandwidth allocated to each flow. This approach may not be as accurate as packet-level simulation, but it is much faster. SIMGRID [7, 16] (a toolkit for simulating distributed applications) uses this approach for its network simulation. The right side of Table 1.1 shows results obtained
Table 1.1: Simulated time and simulation time observed with a packet-level simulator (GTNetS) and an analytical model (SIMGRID). The time to transfer 100 MB data is measured for a randomly generated topology with 200 nodes and various numbers of flows. Ratio $\equiv (\text{simulation time})/(\text{simulated time})$. The unit of time is seconds.

<table>
<thead>
<tr>
<th># of flows</th>
<th>Packet-level</th>
<th>Analytical</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>simulated</td>
<td>simulation</td>
</tr>
<tr>
<td></td>
<td>(s)</td>
<td>(s)</td>
</tr>
<tr>
<td>10</td>
<td>70.31</td>
<td>65.34</td>
</tr>
<tr>
<td>25</td>
<td>87.77</td>
<td>163.10</td>
</tr>
<tr>
<td>50</td>
<td>93.73</td>
<td>364.67</td>
</tr>
<tr>
<td>100</td>
<td>93.19</td>
<td>753.42</td>
</tr>
<tr>
<td>200</td>
<td>124.10</td>
<td>1562.90</td>
</tr>
</tbody>
</table>

with the analytical network model of SIMGRID proposed in [7]. We can see that simulation time is improved by several orders of magnitude when compared to packet-level simulation. However, one concern here is that such analytical models are not as accurate as packet-level simulation.

Contributions of this thesis

In this thesis, we investigate the trade-offs between simulation cost and accuracy for packet-level and analytical simulations by comparing three existing packet-level network simulation tools, GTNetS [31], SSFNet [9], and ns-2 [26], with the analytical simulation model implemented in SIMGRID [7, 16]. Our goal is to identify the regimes in which the analytical model, which is orders of magnitude faster, provides accuracy comparable to packet-level simulation. We find that the factors impacting the accuracy of SIMGRID when compared to packet-level simulation are:

- the size of the data transferred by TCP flows;
- the complexity of the network topology;
- the network contention (i.e., how crowded the topology is).

Our results quantify the impact and interplay of these factors. Overall, our conclusion is that the analytical network simulation of SIMGRID is a viable alternative to packet-level simulation in many relevant scenarios. We find that the network model of SIMGRID does achieve comparable accuracy to the packet-level simulators for large data (larger than 1
MB) and low network contention. For a single link topology, the throughput error is below 5% if the data size is larger than 10 MB. If the data size is 100 MB, even for a randomly generated topology with 200 nodes and a number of flows, for more than 70% of flows the error rate is under 1% if the network contention is not high. The error rate tends to become larger when the links are more congested.

Another goal in this thesis is to enable users to seamlessly switch between both simulation approaches and make their own decision regarding the trade-offs between simulation accuracy and cost. To this end we have implemented an interface between SIMGRID [16] and GTNetS [31], which allows SIMGRID users to switch between packet-level and analytical network simulations as they wish.

This thesis is organized as follows. Chapter 2 discusses related work. Chapter 3 gives an overview of packet-level network simulation, analytical network models, and the network model in SIMGRID. Chapter 4 presents experimental results on comparison between packet-level simulators and the network model in SIMGRID. Chapter 5 describes our integration of GTNetS into SIMGRID. Finally, Chapter 6 summarizes the contributions and impacts.
Chapter 2

Related Work

In this chapter, we review (1) simulation tools that could be employed to conduct simulation for large-scale distributed systems, and (2) studies on comparison between packet-level and analytical simulations.

2.1 Simulators

Several simulators for large-scale distributed systems have been proposed. We review four existing simulators, SIMGRID [16], Bricks [1, 5], GridSim [14, 35], and MicroGrid [23, 34]. We focus especially on their network models.

2.1.1 SIMGRID

SIMGRID [16] is a toolkit written in C that provides core functionalities for the simulation of distributed applications in heterogeneous distributed environments. SIMGRID performs event-driven simulation. SIMGRID considers the platform as a set of resources (network links, workstations, etc.), on which actions (flows, computations) are simulated. More details of its architecture are described in Section 5.2.1.

SIMGRID uses an analytical model for its network simulation. The details of the model are described in Section 3.3. In the first part of this thesis (Chapter 4), we evaluate the current network model of SIMGRID by comparing it with packet-level network simulators. In the second part (Chapter 5), we modify SIMGRID to integrate a packet-level simulator into it.
2.1.2 Bricks

Bricks [1, 5] is a performance evaluation system for scheduling algorithms and frameworks of high performance global computing systems. Bricks, which is written in Java, allows the simulation of various behaviors such as network and resource scheduling algorithms. In Bricks, the users are able to specify network topologies, server architectures, communication models and scheduling framework components. Bricks provides two network models. In both models, networks are modeled as queuing systems. The first model assumes that the network congestion is represented by adjusting the amount of arrival data from extraneous traffic. For this model, the users need to specify ideal bandwidth, the average of actual bandwidth, the average size of extraneous data, and their variances. The cost and the accuracy depend on the packet size: smaller size provides greater accuracy but larger computation cost. In the other network model, the variation of the network bandwidth at each time-step is determined by parameters observed in the real network environment. It achieves high accuracy. However, it requires that the users measure the parameters prior to the simulation.

2.1.3 GridSim

GridSim [14, 35] is a Java-based discrete event grid simulation package. It allows modeling of heterogeneous resources and schedulers. GridSim is implemented on top of an existing discrete event simulation engine, SimJava2. Its earlier versions did not allow for network specification, nor did they have functionality to connect resources through network links. The latest version has been extended: it allows the users to create a network topology; it packetizes data into smaller chunks for sending over a network; and it generates background traffic. Its transport protocol model is similar to UDP rather than TCP. Since it packetizes data and sends them one by one, the simulation cost can be very expensive when the data size is large.

2.1.4 MicroGrid

MicroGrid [23, 34] is a emulation tool which allows the users to run Grid applications on virtual Grid resources. It virtualizes every resource of a Grid platform (memory, CPU, and network). Just as in LAPSE (Large Application Parallel Simulation Environment) [11], this virtualization is achieved by trapping every relevant library and system call. The first
version of MicroGrid uses a modified version of ns-2 [26] for its network simulation. The latest version of MicroGrid uses the MaSSF system for the network simulation. MaSSF is based on DaSSF [10] and runs in parallel on clusters using MPI. MicroGrid is an emulator and thus can provide realistic results as it can capture low-level behaviors. However, the simulation cost tends to be large.

2.2 Comparison between packet-level and analytical simulations

Some studies have been conducted on analyzing and comparing fluid/analytical models and packet-level simulation, and are therefore particularly related to this work.

Liu et al. [18, 19] evaluated the relative performance of fluid simulation over packet-level simulation by analyzing several networking scenarios. In fluid simulation, network traffic is modeled as continuous fluid flows rather than discrete packet instances. The results showed that the simulation execution time is proportional to the simulation event rate. The fluid model usually outperforms the packet-level simulator. However, when flows compete for resources, which increases computation, it may be less efficient than packet-level simulation. Nicol et al. [25] also studied the performance/accuracy trade-offs between fluid models and packet-level simulation. The results showed that the relative error of fluid simulation is very small compared to the results of packet-level simulation. Nevertheless, the cost of simulation would be prohibitive for simulating long-running large-scale distributed applications, which is the goal of SIMGRID.

Schwefel et al. [33] compared three different analytical TCP models with results of simulations of detailed TCP behavior. The results showed that all of the analytical models have deficiencies in some scenarios for which the models' assumptions are violated. Otherwise, the results showed that the accuracy of the models are within acceptable bounds. The analytical TCP models compared in their study require some parameters such as loss rate. Such parameters were determined by the results of simulation experiments for two scenarios. A difference between this work and ours is the network models: the network model in SIMGRID is a simplified version, which does not require the user to provide those parameters. Although the accuracy may not be as high as with the models in [33], the omission of these parameters in the model makes it possible to instantiate it easily in a simulation.
Chapter 3

Network simulation

In this chapter, we review both packet-level network simulation and analytical network models.

3.1 Packet-level simulation

Packet-level simulation uses discrete event simulation techniques. A flow over network links can be represented as a sequence of events, each of which occurs at a certain discrete time. For example, the departure of a packet and its arrival are treated as two separate events. The simulator models such events and keeps a list of events to be processed. The simulator assigns a time to each event, and when the simulation clock reaches this time, the simulator executes the event and changes the statuses of other events if necessary. Since each packet is simulated, the accuracy of the simulation is high. The time needed for the simulation, however, increases roughly in proportion to the number of events [19].

3.1.1 Popular packet-level simulators

Over the years, researchers have developed many packet-level simulators. We give here an overview of three popular packet-level simulators.

ns-2

The ns-2 simulator [26] is a widely used discrete event simulator for networking research. It includes several network protocols such as TCP and UDP, queuing models such as DropTail and RED, and application models such as HTTP and FTP. It supports both wired and wireless networks. ns-2 is written in C++ and OTcl. C++ is used for implementation of protocols, queueing models, and so on. OTcl is mainly used for configuration.
One weakness of ns-2 is its low scalability: ns-2 can be comfortably used on topologies of up to 1,000 network elements, but it does not scale well beyond that size [30].

Georgia Tech Network Simulator (GTNetS)

The Georgia Tech Network Simulator (GTNetS) [31] is designed specifically to allow large-scale simulations. The design philosophy is to make the structure of the simulator closely reflect that of real network protocols. In GTNetS, protocol stack layers are separated as they are in real networks. It is implemented in C++. GTNetS can be used on topologies of up to 177,000 nodes on an inexpensive workstation with 2Gb of main memory [30].

The Scalable Simulation Framework (SSFNet)

SSFNet [9] is a Java implementation of the Scalable Simulation Framework (SSF), which is a unified, object-oriented application programming interface for a public domain standard of discrete-event simulation of large complex systems. Its original design is to be scalable and to support high performance simulation. SSFNet can be used on topologies of up to 100,000 network elements [30]. The Domain Modeling Language (DML) is used for network configuration.

3.2 Analytical modeling

A number of analytical models for TCP performance have been proposed [29]. We focus on the TCP protocol because it is typically used for large-scale distributed applications. TCP throughput is characterized by the TCP congestion control mechanism. TCP uses a window-based flow control mechanism. There are two windows: the congestion window ($cwnd$), which determines the amount of data that the sender can transfer before receiving an acknowledgment (ACK), and the advertised window ($rwnd$), which represents the receiver's resource capacity. The window size of the sender, $w$, is defined as the minimum of $cwnd$ and $rwnd$, and governs the data transmission. While $rwnd$ is advertised by the receiver at the beginning of a connection, $cwnd$ varies throughout the connection based on the level of congestion in the network. To determine appropriate $cwnd$, congestion control, slow start, and congestion avoidance mechanisms are used by TCP.
3.2.1 TCP Slow start and congestion avoidance

At the beginning of a connection, the sender does not know the congestion level in the network. To determine the available capacity efficiently, a mechanism called slow start is used: When a connection is established, \( cwnd \) is set to one packet. When the sender receives the ACK for this packet, it increments \( cwnd \) by one and sends two packets. When the sender receives the corresponding two ACKs, it increments \( cwnd \) by two and sends four packets. In this way, \( cwnd \) is doubled for each RTT. This exponential growth of \( cwnd \) continues until congestion is observed, or \( cwnd \) exceeds the slow start threshold, \( ssthresh \).

When congestion is detected, \( cwnd \) is reduced to the half of the current size. In case of a time-out event, \( cwnd \) is reduced to one, and slow start resumes. If \( cwnd \) becomes larger than \( ssthresh \), the increment rate becomes linear rather than exponential (congestion avoidance phase).

3.2.2 Modeling TCP throughput

In this section, we review an analytical TCP throughput model developed in [29]. Like other proposed such models [21, 28], the work in [29] models throughput as a function of packet loss and round trip delay. It also assumes that the flow is long enough so that packet transmission is in steady state. Consequently, it assumes that the time spent in slow start is negligible. Another assumption is that the round trip time is independent of the window size. A more complex aspect of this model, when compared to other proposal models [21, 28], is that it captures not only the behavior of TCP's fast retransmit mechanism, but also the behavior of TCP's timeout mechanism. Also, the model predicts throughput well over a wider range of loss rates than previously proposed models. The TCP throughput modeled in [29] is:

\[
B(p) = \min \left( \frac{W_{\text{max}}}{\text{RTT}}, \frac{1}{\text{RTT} \sqrt{\frac{3p}{8}} + T_0 \min \left( 1, 3 \sqrt{\frac{3p}{8}} \right) p (1 + 32p^2)} \right),
\]

where \( W_{\text{max}} \) is the receiver's advertised window, \( p \) is the loss indication rate, \( T_0 \) is TCP's average retransmission time-out value, and \( b \) is the number of packets that are acknowledged by each ACK. The first term, \( W_{\text{max}}/\text{RTT} \), corresponds to the constraint due to the advertised window size. Note that simpler models only use the first term in the denominator of the second term in the above equation.
Although it is shown in [29] that this model captures the TCP congestion control characteristics under the given assumptions, parameters $b$ and $p$ are not usually known. However, for simulation purposes, one needs to model TCP throughput using known parameters such as physical latency and bandwidth. Therefore, although this model is interesting, it is not clear that it can be instantiated effectively for simulation. It is likely that a simpler, but perhaps less accurate model would be needed.

### 3.2.3 Bandwidth sharing

While modeling the throughput of a single flow appropriately is important, an equally important phenomenon to capture is the sharing of bandwidth among flows using the same link(s).

Several researchers have explored the questions of bandwidth-sharing between TCP flows [15, 20]. Most works model bandwidth-sharing with fluid flows; flows are treated as continuous fluid rather than discrete packet instances. We review three well-known models for bandwidth sharing. We show the results by the models on the classical example of a linear network as shown in Figure 3.1. The network is represented as a set of links $\mathcal{L}$ where link $l \in \mathcal{L}$ has a capacity $C_l > 0$. A flow is defined by a sequence of links, that is, a subset of $\mathcal{L}$. Let $\mathcal{F}$ be the set of flows. Let $\lambda_f$ be the data transfer rate for flow $f$. A feasible bandwidth allocation must satisfy the following constraints:

$$\forall l \in \mathcal{L}, \sum_{f \ni l} \lambda_f \leq C_l.$$ 

This constraint states that links cannot deliver more bandwidth than their capacities.
MaxMin Fairness

MaxMin fairness is the traditional bandwidth-sharing principle. The objective is to maximize the minimum of \( \{\lambda_f\} \). \( \lambda_f \) is MaxMin fair if and only if an increase of any \( \lambda_f \) within the domain of feasible allocations must be at the cost of a decrease of some \( \lambda_{f'} \) such that \( \lambda_{f'} < \lambda_f \). This leads to the following formula:

\[
\forall f \in \mathcal{F}, \quad \exists l \in f, \quad \sum_{f' \neq l} \lambda_{f'} = C_l \quad \text{and} \quad \lambda_f = \max\{\lambda_{f'}, f' \ni l\}.
\]

In the linear network shown in Figure 3.1 under MaxMin fairness, all flows achieve the same data transfer rate:

\[\forall l, \quad \lambda_l = \frac{C}{2}.\]

Proportional Fairness

In [15], Kelly questioned the validity of MaxMin fairness as a way to model TCP behavior. Indeed MaxMin fairness allocates more network resources to long flows than to short flows. However, TCP is known to do just the opposite. As an alternative to MaxMin fairness, Kelly proposed proportional fairness, which is defined as follows. The objective of proportional fairness is to maximize

\[\sum_{f} \lambda_f \log(\lambda_f).\]

The solution must satisfy the following criteria: \( (\lambda_f)_{f \in \mathcal{F}} \) is unique and for any other feasible allocation \( (\lambda'_f)_{f \in \mathcal{F}} \),

\[\sum_{f \in \mathcal{F}} \frac{\lambda'_f - \lambda_f}{\lambda_f} \leq 0.\]

In the linear network shown in Figure 3.1 under proportional fairness, we have

\[\lambda_0 = \frac{C}{L+1},\]

\[\forall l \neq 0, \quad \lambda_l = \frac{C(L-1)}{(L+1)}.\]

Potential Delay Minimization

Another idea is to minimize the time to complete all transfers. Assume that the data size transferred is fixed. One must then minimize the potential delay \( 1/\lambda_f \) for all flows.
Such an allocation minimizes $\sum_{f} 1/\lambda_f$. In the linear network shown in Figure 3.1 under potential delay minimization, we have

$$\lambda_0 = \frac{C}{1 + \sqrt{L}},$$

$$\forall l \neq 0, \quad \lambda_l = \frac{C\sqrt{L}}{(1 + \sqrt{L})}.$$

### 3.3 Network simulation in SIMGRID

In this section, we review the network model used in SIMGRID. SIMGRID is a toolkit for the simulation of Grid applications [16]. SIMGRID uses an analytical network simulation model. In SIMGRID, a network is represented as an ensemble of network links. Each link is characterized by a physical latency (in seconds), $\alpha$, and a physical bandwidth (in bytes per seconds), $\beta$. The network model in SIMGRID basically calculates data transfer time from these two physical parameters. The network model in SIMGRID consists of two parts: modeling for a single flow (or multiple flows that do not compete for resources), and modeling bandwidth sharing.

#### 3.3.1 Modeling a single flow

Consider a single flow over a single link, $l$, whose physical latency is $\alpha_l$ and physical bandwidth is $\beta_l$. The time, $T_l$, to transfer $S$ bytes of data over the link is modeled as:

$$T_l = \alpha_l + \frac{S}{\beta'_l},$$

where $\beta'_l$ is the experimental bandwidth. Inspired by the work described in Section 3.2.2, in SIMGRID, $\beta'_l$ is modeled as

$$\beta'_l = \min\left(\beta_l, \frac{W_{\text{max}}}{\text{RTT}}\right),$$

where $W_{\text{max}}$ is the maximum advertised TCP window size, RTT is the round-trip time. This upper bound can be roughly explained by the TCP mechanism: the sender sends $W_{\text{max}}$ bytes of data, and then waits for an acknowledgment, that is, waits one RTT. Hence, the average throughput is equal to $W_{\text{max}}$/RTT. As an approximation of the RTT, two times of the physical latency is used in SIMGRID. In SIMGRID, router queue times are assumed to be included in latencies.
3.3.2 Bandwidth-Sharing Model

In the previous section, we have explained the network model for a single flow. In this section, we extend it to multiple flows on multiple links.

In Section 3.2.3, we reviewed three well-known bandwidth sharing models. Which one should be used? It is said that the TCP protocol is "close" to proportional fairness as it favors short flows. However, Chiu [8] shows that TCP does not exactly implement proportional fairness. Indeed, the analytical models for TCP throughput in [12, 29] approximate to

\[ B(p) = \frac{c}{\text{RTT} \sqrt{p}} \]

where \( p \) is the fraction of packets lost and \( c \) is some constant, if \( p \) is not too high. Assuming all flows experience the same loss rate, \( p \), this formula suggests that bandwidth is shared in inverse proportion to RTT. In [7], experiments are presented that compare five candidate bandwidth-sharing related these bandwidth sharing models. SIMGRID implements the INV-RTT-BOUNDED model, which led to the best results, as the bandwidth sharing model. In this model, the bandwidth allocated to flows competing over a bottleneck link is inversely proportional to the ratio of their RTTs: each flow \( f_i \) is allocated a share of the bandwidth on the bottleneck link with a weight \( w_i \) such that:

\[ w_i = \frac{1}{\sum_{l \in f_i} \alpha_i}. \]

The bandwidth allocated to \( r_i \) is bounded by:

\[ W_{\text{max}} / \sum_{l \in f_i} \alpha_i. \]

Here, a link is a bottleneck if the sum of the bandwidths allocated to the routes over this link is equal to the total bandwidth of the link.

3.3.3 Limitations of the network model in SIMGRID

The network model in SIMGRID requires little computation, and so enables fast simulation. To enable fast simulation, however, the network model in SIMGRID makes some assumptions and ignores some features of TCP flows. In this section, we review those limitations.

- SIMGRID does not provide an abstraction for routing. The users have to specify routes statistically.
- The analytical network model in SIMGRID assumes infinite flows in steady state, and thus, simulation of short flows may be inaccurate. This is also true of most previously proposed analytical models.

- In SIMGRID, characteristics of the network are basically determined by two parameters, physical latency, $\alpha$, and physical bandwidth, $\beta$. Therefore, it is assumed that TCP features are somehow parameterized by these two values. For instance, RTT is assumed to be twice the physical latency, which is constant. But in fact, RTT varies in real networks. Also, $\alpha$ includes router queue times, which also vary depending on network congestion.
Chapter 4

Experimental Evaluation

In this chapter we perform a quantitative comparison of results obtained via packet-level simulation using three packet-level simulators (ns-2 [26], SSFNet [9], and GTNetS [31]) and results obtained via the analytical models implemented in SIMGRID [16]. We performed three series of experiments:

- one and multiple flows over a single link;
- two flows over a dogbone topology;
- various numbers of flows over randomly generated topologies.

Note that some of our results reproduce results in [7]. However we use a newer version of SIMGRID in which several bugs have been fixed thanks to our own investigation. Also we explore much wider ranges of parameters.

4.1 A single network link

4.1.1 Experimental design

First, we conduct experiments with a single link. Figure 4.1 shows the topology used in this experiment. The TCP sender sends $S$ bytes of data to the receiver over a single link with a latency of $\alpha$ and a bandwidth of $\beta$. We measure the transfer time, $T$, for the following cases:

- data size, $S$ (KB): 1, 10, 100, $10^3$, $10^4$, $10^5$, $10^6$
- latency, $\alpha$ (ms): 0, 1, 10, 20, 40, 60, 80, 100
Figure 4.1: Topology for a single flow over a single link.

- bandwidth, $\beta$ (KB/sec): 100, 200, 400, 600, 800, $10^3$, $10^4$, $10^5$, $10^6$

The advertised TCP window size is set to 20 KB in all simulators. TCP Reno is used as the TCP model and DropTail is used as the queue model in ns-2, SSFNet, and GTNetS. For SSFNet, the parameter TCP_FAST_INTERVAL($D$ hereinafter), which determines the delayed-ack time, is set to 0.2 and 0.01.

From the measured transfer time, we calculate the throughput, $\Theta$, and the relative error, $E$, which are defined as follows:

$$\Theta = \frac{S}{T},$$
$$E = \frac{(\Theta_{SIMGRID} - \Theta_{packet-level})}{\Theta_{packet-level}}.$$

Therefore, a negative (positive) $E$ value means that SIMGRID underestimates (overestimates) achieved throughput.

4.1.2 Results

Figure 4.2 shows the throughput as a function of $\alpha$ and $\beta$ for GTNetS, ns-2, and SSFNet for data size $10^5$ KB and 100 KB. The mesh shows the throughput by the analytical network model used by SIMGRID, i.e.,

$$\Theta_{SIMGRID} = \frac{S}{\alpha + \frac{S}{\beta}},$$

(4.1.1)

where $\alpha$ is latency and $\beta'$ is defined by Eq. (3.3.2). When the data size is $10^5$ KB (Figure 4.2, upper panel), the throughputs of the packet-level simulators and those of SIMGRID are almost the same. However, when the data size is 100 KB (Figure 4.2, lower panel), the throughputs of the packet-level simulators are smaller than those of SIMGRID.

In order to examine these results in more detail, we replot the graph in 2D as a function of data size for $\alpha = 10$ ms and $\beta = 10^5$ KB/sec in Figure 4.3. According to Eq. (4.1.1), when $\alpha$ is small compared to $S/\beta'$, that is, when $S$ is large compared to the fixed $\alpha$ and
Figure 4.2: Throughput for ns-2, SSFNet, and GTNetS as a function of bandwidth and latency. The mesh shows the throughput by the analytical network model used by SIMGRID. The upper panel shows the throughput for data size $= 10^5$ KB and the lower panel for data size $= 100$ KB. For the lower panel, to show the data points where the throughputs are smaller than the mesh, the z-axis is reversed. Squares ($\square$) show the throughput with GTNetS; Circles ($\bigcirc$) with ns-2; Crosses ($\times$) with SSFNet ($D = 0.2$); and Pluses ($+$) with SSFNet ($D = 0.01$).
$\beta^\prime$, $\Theta_{\text{SimGRID}}$ approaches $\beta^\prime$. For $\alpha = 10\text{ms}$ and $\beta = 10^5\text{ KB/sec}$, $\beta^\prime = 10^3\text{ KB/sec}$, which is bounded by $\alpha$. And indeed, the upper panel in Figure 4.3 shows that the throughput approaches $10^3\text{ KB/sec}$ as the data size increases for all the simulators. When the data size is larger than $10^4\text{ KB}$, all throughputs are between $0.98 \times 10^3\text{ KB/sec}$ and $10^3\text{ KB/sec}$.

The lower panel in Figure 4.3 shows the relative error in throughput achieved in the SimGRID simulation when compared to the ones achieved in the ns-2, SSFNet, and GTNetS simulations, versus data size. The error is positive for all cases, which means that SimGRID is always optimistic (it gets higher throughput than the packet-level simulators). When the data size is large, the error is small: below 30\% for data size $= 10^3\text{ KB}$, below 2.5\% for $10^4\text{ KB}$, and below 1\% for $10^5\text{ KB}$.

Figure 4.4 shows the throughput and the error for $\alpha = 10\text{ ms}$ and $\beta = 100\text{ KB/sec}$. In this case, $\beta^\prime$ is bounded by $\beta$, and thus $\beta^\prime = 100\text{ KB/sec}$. The trend of the throughput is similar to that in the previous case: SimGRID is always optimistic when compared to the packet-level simulators, and the throughput increases as the data size becomes larger. However, the throughputs of the packet-level simulators approach a value around 95\text{ KB/sec} rather than 100\text{ KB/sec} for SimGRID, which means that the error becomes no less than 6\% even though the data size is larger than $10^5\text{ KB}$. We show graphs for other combinations of $\alpha$ and $\beta$ in Appendix A.

Note that we use $D = 0.2$ and 0.01 (the former value is the default) to see parameter sensitivity. The results show that the throughputs with the value of 0.2 are smaller than those with the value of 0.01 when the data size is smaller than $10^4\text{ KB}$. Also, the results with the value of 0.01 are more similar to the other packet-level simulators, GTNetS and ns-2.

We conducted the same experiments for different TCP window sizes ($10\text{ KB}$ and $30\text{ KB}$). The throughput increases as the TCP window size increases, but the trends for the error are the same as those above. We show these results in Appendix B.

The fact that the error decreases as the data size increases, can be explained by the TCP slow start mechanism. With this mechanism, the TCP congestion window starts from a small size. Each time the sender receives an ACK from the receiver, it increases the window size by a factor 2 until the size reaches the slow start threshold or until congestion occurs. Hence, temporary throughput in slow start is smaller than that in the steady state. When the data size is small, this temporary small throughput affects the entire throughput. When the data size is large, time in the steady state is relatively large.
Figure 4.3: The throughput (upper panel) and the relative error in throughput (lower panel) for ns-2, SSFNet ($D = 0.2, 0.01$), GTNetS, and SimGrid as a function of data size. $\alpha = 10\text{ ms}$ and $\beta = 10^5\text{ KB/sec}$. 
Figure 4.4: The throughput (upper panel) and the relative error in throughput (lower panel) for ns-2, SSFNet ($D = 0.2, 0.01$), GTNetS, and SimGRID as a function of data size. $\alpha = 10$ ms and $\beta = 100$ KB/sec.
when compared to time in slow start, and thus, the effect of slow start is small. Since SimGrid does not implement the TCP slow start algorithm, when the data size is small, its transfer rate is much larger than that with the packet-level simulators, and so is the error.

The results also shows that when $\beta'$ is bounded by its physical bandwidth, $\beta$, as the data size increases, the error rates approach 4–6%, while they approach 0% when $\beta'$ is bounded by its latency. It indicates that the physical bandwidth should be scaled by some factor to reflect the experimental bandwidth.

### 4.1.3 Single network link, multiple flows

We extend our experiments to multiple flows over a single link. The parameters are set to the following:

- latency, $\alpha$ (ms): 0, 1, 10, 100
- bandwidth, $\beta$ (KB/sec): 100, $10^3$, $10^4$, $10^5$, $10^6$
- number of flows, $N_f$: 2, 16, 32

Since we already know from Section 4.1 that SimGrid does not agree with the packet-level simulators when the data size is smaller than $10^3$ KB, we use $10^5$ KB as the data size for these experiments.

The results show that the error in throughput is small for all the parameters above (see Appendix C for details.) The throughputs measured in these experiments are simply the same as those in the single flow experiment, but divided by the number of flows, when $(\beta/N_f) \leq (W_{\text{max}}/2\alpha)$. When $(\beta/N_f) > (W_{\text{max}}/2\alpha)$, each flow is bounded by its latency.

### 4.2 Dogbone topology

#### 4.2.1 Experimental design

In order to validate the analytical bandwidth sharing model, we run experiments on a five-link network topology: the popular Dogbone topology shown in Figure 4.5. As in the previous section, we use $10^5$ KB as the data size for these experiments. Also, we pick GTNetS as our packet-level simulator for comparison SimGrid for the following
experiments. In the topology, the bandwidth of Link 2 (the shared link), $\beta$, and the latency of Link 5, $\alpha$, are as follows:

- latency, $\alpha$ (ms): 0, 1, 10, 20, 50, 100
- bandwidth, $\beta$ (KB/sec): 10, 100, $10^3$, $10^4$, $10^5$, $10^6$

The characteristics of the three other links are fixed as shown in Figure 4.5. We consider two flows, flow A and B, which start simultaneously. We measure the transfer time for each flow and calculate their throughputs for both SIMGRID and GTNetS.

4.2.2 Results

Figure 4.6 plots $\Theta$ versus $\alpha$ for $\beta = 10^5$ KB/sec and $S = 10^5$ KB. The upper panel shows the throughput of flow A. The throughputs are almost constant, at $25 \times 10^3$ KB/sec, for all the latencies for both SIMGRID and GTNetS. The throughput of flow B (Figure 4.6, lower panel) decreases as the latency increases. For this value of $\beta$, $\beta > W_{\text{max}}/(2 \times \alpha)$ for all latencies used in this experiment except for $\alpha = 0$, and thus, $\beta$ is constrained by $\alpha$; i.e., this is a latency bound case. Also, $\beta'_{A} = W_{\text{max}}/(2 \times 40 \text{ ms}) = 25 \times 10^3$ KB/sec for flow A and $\beta'_{B} = W_{\text{max}}/[2 \times (30 \text{ ms} + \alpha)] \leq 33.4 \times 10^3$ KB/sec for flow B, and thus, $\beta'_{A} + \beta'_{B} < 10^5$ KB/sec; therefore, the shared link is not a bottleneck. (Recall that $W_{\text{max}} = 20$ KB.) Hence, with the analytical model in SIMGRID, the throughputs are proportional to the inverse of the sum of latencies of the links used for the flows. For flow A, the sum of the latencies is constant, and thus, the throughput is constant, which explains the results.
For flow B, the sum of the latencies varies. As the value of $\alpha$ increases, the throughput, which is proportional to the inverse of the sum, decreases, which also explains the results.

Figure 4.7 shows the relative error in throughput for $\beta = 10^5$ KB/sec. SIMGRID always overestimates throughput (optimistic) compared to GTNetS. The error of flow A is almost the same for all the latencies, while that of flow B decreases as the latency increases. However, the errors for both flows are smaller than 0.15% for all the latencies, which means that the analytical model in SIMGRID models throughput of TCP well for this case.

Figure 4.8 plots $\Theta$ vs. $\alpha$ for $\beta = 100$ KB/sec and $S = 10^5$ KB. With both SIMGRID and GTNetS, we observe the following trends:

1. When $\alpha > 10$ ms (i.e., when flow A is shorter than flow B), flow A has more bandwidth than flow B.
2. When $\alpha < 10$ ms (i.e., when flow B is shorter than flow A), flow B has more bandwidth than flow A.

A difference between the results of the two simulators is that shorter flows with GTNetS do not get as much bandwidth as those with SIMGRID. For instance, for $\alpha = 100$ ms, the throughput of flow A is about 76 KB/sec and the throughput of flow B is about 45 KB/sec with SIMGRID, and thus, the throughput of flow A is 1.7 times larger than that of flow B. With GTNetS, the throughput of flow A is about 57 KB/sec and that of flow B is about 45 KB/sec, and thus, the throughput of flow A is 1.2 times larger than that of flow B, which is smaller than that with SIMGRID. For this value of $\beta$, $\beta'$ is constrained by $\beta$ for all latencies, i.e., this is a physical bandwidth bound case and the shared link is the bottleneck link. Hence, the bandwidth is allocated to flow A and B in proportion to the inverse of the sum of their latencies in SIMGRID. From the results, however, the bandwidth allocation with SIMGRID is different from that with GTNetS, though the trends that shorter flows get more bandwidth than longer flows are similar.

Figure 4.9 shows the throughput error for $\beta = 100$ KB/sec. SIMGRID always overestimates throughput (optimistic) as it was the case for $\beta = 10^5$ KB/sec. The error is larger than 5% for all the latencies except for flow B with latency 100 ms. This value of 5% is consistent with the error over the single link shown in Section 4.1. When the difference of the latencies of two flows is large, the error becomes large: the error for flow A reaches 35% when the latency is 100 ms.
Figure 4.6: Throughput for SimGRID and GTNetS as a function of latency. $\beta = 10^5$ KB/sec, $S = 10^5$ KB. The upper panel shows the throughput for flow A and the lower panel for flow B.
These results are consistent with the results in [7], which indicates that when the transfer rate is limited by its physical bandwidth, the error is higher than when it is limited by latency.

### 4.3 Random topology

We now validate the analytical network model on randomly generated topologies. A universal topology generator, Boston university Representative Internet Topology gEnerator (BRITE) [6, 22] is used to generate the topologies. We simulate various numbers of flows that start simultaneously on the topologies, and their transfer time are measured. The throughputs with SimGRID and GTNetS are compared.

#### 4.3.1 Experimental design

We conduct experiments with the following parameters:

- data size, $S$ (KB): 100, $10^5$
Figure 4.8: Throughput for SimGRID and GTNetS as a function of latency. $\beta = 100$ KB/sec, $S = 10^5$ KB. The upper panel shows the throughput for flow A and the lower panel for flow B.
Figure 4.9: Throughput error of SIMGRID as a function of latency.

- number of nodes, $N_n$: 50, 200
- number of flows, $N_f$: 10, 20, 50, 100, 200

The ASWaxman model implemented in BRITE is used to generate the topologies. The physical bandwidths are sampled between 10 MB/sec and 128 MB/sec with uniform distribution for $N_n = 50$. For $N_n = 200$, two topologies with two different ranges of bandwidths are generated: one is from 100 MB/sec to 128 MB/sec, and the other is from 10 MB/sec to 128 MB/sec. There is no option in BRITE for the users to set parameters for the physical latencies: BRITE automatically samples the values for them.

The procedure of the experiments is as follows:

1. generate a random topology of $N_n$ nodes;
2. generate $N_f$ flows on the topology by picking random pairs of nodes;
3. simulate;
4. measure the transfer time for each flow;
5. repeat from step 2 to 4 for 10 times.
4.3.2 Results for $S = 10^5$ KB

First, we look at each flow one by one for some of the experiments. Figure 4.10 shows the error in throughput for each flow for three cases:

1. $N_n = 200$ and $N_f = 200$ with $\beta = 100-128$ MB/sec (larger $\beta$'s);
2. $N_n = 200$ and $N_f = 200$ with $\beta = 10-128$ MB/sec (smaller $\beta$'s);
3. $N_n = 50$ and $N_f = 200$ with $\beta = 10-128$ MB/sec.

Ten experiments with different combinations of flows, which are randomly picked, are conducted for each case; we show here only one of the ten experiments for each case. One data point corresponds to one flow. The results are sorted by the value of error. The upper panel in Figure 4.10 shows the relative error for case 1. For this case, the percentage of flows whose error is within $\pm 10\%$ (red dashed lines) is about 90% out of the 200 flows. For case 2, the percentage is about 70% (Figure 4.10, middle panel). For case 3, however, the percentage of flows whose error are within $\pm 10\%$ is around 50%. This difference comes from the difference in the number of flows whose error are smaller than $-10\%$ (that is, where SIMGRID is pessimistic). If we broaden the border from $\pm 10\%$ to $\pm 50\%$ (green dashed lines), 90% of the flows are within the borders even for case 3.

Although the patterns of error for the same $N_n$ and $N_f$ cases vary with different combinations of flows, the trends are similar. We show the results for the other cases in Appendix E.

In order to examine the results in more detail, we define a threshold, $\gamma$, for throughput error, and categorize the flows into the following three cases:

A. flows whose error are less than $-\gamma$;

B. flows whose error are within $\pm \gamma$;

C. flows whose error are larger than $\gamma$.

We define the percentage of flows categorized into the three categories, A, B, and C, as $P_A$, $P_B$, and $P_C$ respectively. By definition, $P_A + P_B + P_C = 100\%$.

Figure 4.11 shows the percentage of each category as a function of $N_f$ for $\gamma = 0.01$ for the following three cases:

1. $N_n = 200$ with $\beta = 100-128$ MB/sec;
Figure 4.10: Relative error in throughput of SIMGRID for each flow. The upper panel shows the error for one of the experiments in case 1, the middle panel for case 2, and the lower panel for case 3. The red dashed lines show $E = \pm 0.1$, and the green lines show $E = \pm 0.5$. 
2. \( N_n = 200 \) with \( \beta = 10-128 \text{MB/sec} \);

3. \( N_n = 200 \) with \( \beta = 10-128 \text{MB/sec} \).

Each data point shows \( P_A (\times), P_B (\bigcirc), \) or \( P_C (\square) \) for one experiment. Since ten experiments with different combinations of flows are conducted for each case, there are ten points per category per \( N_f \). For small number of flows such as \( N_f = 10 \) and \( N_f = 25 \), the variances within the same number of flows are large. For \( N_f = 100 \) and \( 200 \), whose variances are relatively small, we observe the following trends:

- In case 1 (Figure 4.11, upper panel), \( P_B \) is the largest for all the experiments. For \( N_f = 200 \), \( P_A \) and \( P_C \) are between 10\% to 20\%. \( P_B \) extends from 65\% to 80\%, that is, error of 65\% to 80\% of flows are within ±1\%.

- In case 2 (Figure 4.11, middle panel), \( P_C \) is the largest for most of the experiments, and \( P_A \) is the second largest. For \( N_f = 200 \), \( P_A \) extends from 42\% to 65\%. \( P_B \) extends from 15\% to 20\%.

- In case 3 (Figure 4.11, lower panel), \( P_A \) is the largest for more than half of the experiments, and \( P_C \) is the second largest. For \( N_f = 200 \), \( P_A \) extends from 42\% to 60\%. \( P_B \) extends from 15\% to 30\%.

Figure 4.12 shows \( P_A, P_B, \) and \( P_C \) for \( \gamma = 0.1 \). For this value of \( \gamma \), \( P_B \) is the largest for most of the experiments. For \( N_f = 200 \), \( P_B \) extends from 80\% to 90\%, from 65\% to 80\%, and from 40\% to 60\% in cases 1, 2, and 3, respectively.

The results above indicate that the throughput error is larger with smaller \( \beta \)'s if \( N_n \) and \( N_f \) are the same, and with smaller \( N_n \) if \( N_f \) and \( \beta \)'s are the same.

In order to examine what situation causes the large error, we replot the graph with a different x-axis, the minimum value, \( \eta \), of physical bandwidth divided by the number of flows that share the link on the network path of a flow (Figure 4.13). Figure 4.13 summarizes all the flows. The flows are categorized into the three categories, A, B, or C, in the same manner as above. Figure 4.13 shows the cases for \( \gamma = 0.01, 0.1, 0.2, \) and 0.5 from top to bottom respectively. For all the values of \( \gamma \), when \( \eta < 10 \text{MB/sec} \), \( P_A \) and \( P_C \) are the largest. When \( \gamma = 0.01 \), \( P_A \) and \( P_B \) become smaller as \( \eta \) becomes larger. When \( \gamma = 0.1 \) and 0.2, \( P_A \) becomes smaller as \( \eta \) becomes large, while \( P_C \) is almost independent with \( \eta \). When \( \gamma = 0.5 \), more than 95\% of flows are categorized into B, that is, the error of most of the flows are within ±50\%.

31
Figure 4.11: Distribution of error as a function of $N_f$, for case 1 (upper), case 2 (middle), and case 3 (lower) for $\gamma = 0.1$. Crosses ($\times$) show the percentages of flows such that $E < -0.01$; Circles (○), $-0.01 \leq E \leq 0.01$; Squares (□), $E > 0.01$. 
Figure 4.12: Distribution of error as a function of $N_f$, for case 1 (upper), case 2 (middle), and case 3 (lower) for $\gamma = 0.1$. Crosses ($\times$) show the percentages of flows such that $E < -0.1$; Circles ($\bigcirc$), $-0.1 \leq E \leq 0.1$; Squares ($\square$), $E > 0.1$. 
Figure 4.13: Distribution of error as a function of $\eta$, physical bandwidth divided by the number of flows that share the link on the network, for $\gamma = 0.01$ (upper left), $\gamma = 0.1$ (upper right), $\gamma = 0.2$ (lower left), and $\gamma = 0.5$ (lower right).
These results indicate that the throughput error depends on network contention, i.e., how crowded the topology is. The comparison between cases 1 and 2 indicates that the error depends more on the network contention than on the complexity of the topology. (Note that the number of nodes of cases 1 and 2 are the same but the bandwidths of case 1 are larger than those of case 2.) Here we assume that the complexity of the topology is determined by the number of nodes and the number of flows. If $\beta$s are large enough, SIMGRID models TCP throughput well, even though the numbers of nodes and flows are large. SIMGRID may lead to larger error with a relatively simple, but high contention network than with a complex, but less contention network.

4.3.3 Results for $S = 100$ KB

Figure 4.14 shows the throughput error for each flow for $S = 100$ KB. The upper panel shows the results for case 1, the middle panel for case 2, and the lower panel for case 3, where case 1, 2, and 3 are the same as for $S = 10^5$ KB. The trends are similar to those for $S = 10^5$ KB except that the percentage of the flat part in the middle is about 0.8 rather than 0. This value, 0.8, is assumed to be the error in a single network model for data size 100 KB. More flows have lower error than the "flat" rate (0.8) in case 3 than in cases 1 and 2. This may be explained by the fact that network contention is higher in case 3 than in case 1 and 2.

4.3.4 Cost

In order to evaluate the computational cost, we measure the simulation time as well as the simulated time for case 1 ($N_n = 200$ and $\beta = 100$–128 MB/sec). For each number of flows, one combination of flows is picked from the ten different flow combinations used in Section 4.3. Four different sizes of data, 100 KB, $10^3$ KB, $10^4$ KB, $10^5$ KB, are transferred. The measurements are repeated ten times, and the average is taken for the simulation time. The simulations are performed on a machine with four 3.2 GHz Intel Xeon CPUs and 3.6 GB of physical memory running Linux 2.6.8. Table 4.1 shows the simulated and simulation time with GTNetS (left side) and SIMGRID (right side) as well as their ratio. The ratio is defined as simulation time divided by simulated time, and thus, a high ratio means a high simulation cost.
Figure 4.14: Relative error in throughput of SimGRID for each flow. $S = 100$ KB. The upper panel shows the error for one of the experiments in case 1, the middle panel for case 2, and the lower panel for case 3.
Table 4.1: Simulated time and simulation time observed with a packet-level simulator (GTNetS) and an analytical model (SIMGRID). The time to transfer 10³ KB data is measured for a randomly generated topology with 200 nodes and various numbers of flows. Ratio \( \equiv (\text{simulation time})/(\text{simulated time}) \). The unit of time is seconds.

<table>
<thead>
<tr>
<th>( S ) (KB)</th>
<th>( N_f )</th>
<th>GTNetS</th>
<th>SIMGRID</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>simulated</td>
<td>simulation</td>
</tr>
<tr>
<td>100</td>
<td>10</td>
<td>0.141</td>
<td>0.073</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>0.175</td>
<td>0.181</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>0.187</td>
<td>0.409</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>0.186</td>
<td>0.839</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>0.248</td>
<td>1.694</td>
</tr>
<tr>
<td>10³</td>
<td>10</td>
<td>0.773</td>
<td>0.661</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>0.965</td>
<td>1.661</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>1.030</td>
<td>3.697</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>1.024</td>
<td>7.649</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>1.364</td>
<td>15.705</td>
</tr>
<tr>
<td>10⁴</td>
<td>10</td>
<td>7.094</td>
<td>6.549</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>8.856</td>
<td>16.31</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>9.457</td>
<td>36.50</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>9.403</td>
<td>75.31</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>12.522</td>
<td>155.44</td>
</tr>
<tr>
<td>10⁵</td>
<td>10</td>
<td>70.31</td>
<td>65.34</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>87.77</td>
<td>163.1</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>93.73</td>
<td>364.7</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>93.19</td>
<td>753.4</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>124.10</td>
<td>1562.9</td>
</tr>
</tbody>
</table>

For the same \( S \), the simulated time and simulation time both with GTNetS and SIMGRID increase as the number of flows increase. Also, the ratio increases as the number of flows increases.

For the same \( N_f \), the simulated time and simulation time with GTNetS increases as the data size increases. However, the simulation time with SIMGRID does not increase as the data size increases, while the simulated time increases. As a result, the ratio with SIMGRID decreases as the data size increases.

The ratio with SIMGRID is larger than that with GTNetS for all cases. The difference between these ratios increases as the data size increases.
4.4 Discussion

The results of the experiments on a single network link indicate that throughput error decreases as the data size increases. When $\beta'$ is bounded by its physical latency (latency bound), the error approaches 0% as the data size becomes large. By Eq. (3.3.2), $\beta'$ is bounded by its physical latency when the physical latency is large and/or the physical bandwidth is large. For those links and when data size is large ($> 10^4$ KB), SimGrid models TCP throughput well. When $\beta'$ is bounded by its physical bandwidth (bandwidth bound), that is, when the physical latency is small and/or physical bandwidth is small, the error converges to a positive value. Although the value is not so large for bandwidth bounded-cases for large data size, the network model would better reflect TCP throughput if $\beta$ is scaled by some factor. Also, since the advertised TCP window size affects the throughput, it is important to set an appropriate value for it in simulation.

The results of the dogbone experiments indicate that the bandwidth sharing model in SimGrid does not model TCP bandwidth sharing well when the shared link is the bottleneck. Shorter flows are allocated more bandwidth with the network model in SimGrid than with GTNetS. As the difference of the length (i.e., the sum of their physical latencies) of the two flows increases, the error increases. This indicates that when the network is crowded and its contention is high, and thus, many links become bottleneck links, the accuracy of the network model in SimGrid is poor.

The results of the random topology experiments indicate that the throughput error depends on network contention, i.e., how crowded the topology is. The results of cases 1 and 2 indicate that the error depends more on the network contention than on the complexity of the topology. If $\beta$s are large enough, SimGrid would model TCP throughput well, even though the numbers of nodes and flows are large. SimGrid may lead to larger error with a relatively simple, but high contention network than with a complex, but less contention network.

Hence, it is important to obtain information on network contention prior to conducting simulation, when communication times largely affect the simulation results.

The results of the random topology experiments also show that SimGrid often underestimates TCP throughput. This may seem strange, as SimGrid always overestimates TCP throughput in the experiments over a single link and the dogbone links. But indeed, from the results of the experiments on dogbone links, shorter flows with SimGrid have
more bandwidth, and thus, longer flows have less bandwidth than with GTNetS, which may cause the underestimation for the longer flows.

In situations in which the current network model in SimGRID is not accurate, other methods may be suitable to compute TCP throughput are desired. Although some researchers have suggested TCP analytical models for short flows and other bandwidth sharing models, it is not easy to employ them because we do not usually know the parameters required for these models. Another option is to use packet-level simulation. For small data size transfer, the cost may not be prohibitive. A problem may occur for transferring large data over highly contended network as simulation may take a very long time. However, it would be desirable for users to be able to choose between analytical and packet-level models in SimGRID to achieve a desired the trade-off between accuracy and cost. We discuss such a development in the next chapter.
Chapter 5

Integration of SIMGRID and GTNetS

5.1 Introduction and Motivation

In the previous chapter, we observed and discussed the trade-offs between packet-level and analytical network simulations. The choice between packet-level and analytical methods depends on the purpose of the experiments and configuration. Hence, it is desirable that users be given the choice. To this end, we add packet-level simulation support to the SIMGRID tool, by integrating a packet-level simulator into it. We chose GTNetS as the packet-level simulator to be integrated, because it is scalable and written in C++; the latter property is convenient for SIMGRID, which is written in C.

5.2 Overview of GTNetS and SIMGRID

We integrate GTNetS into SIMGRID in such a way that SIMGRID controls GTNetS. If the user chooses to use GTNetS for network simulation, SIMGRID runs GTNetS to get flow throughputs instead of using the analytical model of the current SIMGRID implementation. SIMGRID does not need the details regarding individual packets, but merely needs to know when a flow completes. In this section, we first review the architectures of both simulators, and then we explain their integration.

5.2.1 SIMGRID

The SIMGRID toolkit consists of five main components: SURF, MSG, GRAS, SimDag, and SMPI as shown in Figure 5.1. The last four components provide APIs to simulate several kinds of application and we refer the interested reader to http://simgrid.gforge.inria.fr/
for more details on these. SURF is the simulation kernel and provides the core functionalities to simulate a virtual platform including network resources.

In SIMGRID, the user is required to specify the application and the physical platform. Specification of the physical platform is done by creating hosts, links, and routing tables, which are defined by the user through XML specification files. A host is defined by its CPU speed and CPU availability. A link is defined by a latency and a bandwidth. A route is defined as a series of links.

Once the physical platform is specified and the application model is defined, SURF simulates resource sharing. In network simulation, messages of various sizes are sent and received according to API calls placed by the user code. The sharing of network resources is determined by the analytical model explained in Section 3.3.

SURF simulates actions (flows, computations) on resources (workstations, CPUs, network links). The main idea is to determine the next time at which there will be a change in the system, to simulate action execution on resources until that time, and to repeat. The pseudocode for SURF is described in Algorithm 1.
Table 5.1: GTNetS layers

<table>
<thead>
<tr>
<th>Layer</th>
<th>Models</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application layer</td>
<td>FTP, web browsing</td>
</tr>
<tr>
<td>Transport layer</td>
<td>TCP (Tahoe, Reno, NewReno, and SACK), UDP</td>
</tr>
<tr>
<td>Network layer</td>
<td>IPV4</td>
</tr>
<tr>
<td>Link layer</td>
<td>IEEE 802-3, IEEE 802.11</td>
</tr>
</tbody>
</table>

```
while simulation not finished do
  min_c = time to earliest action completion;
  min_a = time to next action activation;
  min = min(min_c, min_a);
  simulate resource usage for min seconds;
  update all action statuses;
end
```

Algorithm 1: SURF simulation

5.2.2 GTNetS

GTNetS is designed as real networks are: it provides interfaces for a number of popular protocols at the application layer, transport layer, network layer, and link layer. Table 5.1 summarizes the models.

To build and run a simulation in GTNetS, the user is required to specify the topology by network elements such as nodes and links. A node represents an end-system such as desktop workstations or a router. A link represents a point-to-point link, Ethernet, or wireless. A link connects nodes. Once a topology is specified, the flow of data over the topology is described. This is done by adding applications to nodes and specifying data demands on them. For instance, the TCPSend class is one of these applications. After the network topology and the flows are specified, the Simulator class, which is the basic simulation engine, schedules and executes the simulation events. The pseudocode for the main loop of the simulation is described in Algorithm 2.
while simulation not finished do
    event = dequeue the first event;
    handle the event;
    advance the simulation time;
end

Algorithm 2: GTNetS simulator run method

5.3 Design and implementation

In this section, we discuss the integration of GTNetS and the SURF component of SimGrid. We first create an interface to GTNetS that is the basis for the integration. Unfortunately, we had to modify the GTNetS source code for a successful integration. However, we kept these modifications to a minimum (~100 lines) so that future upgrades in GTNetS can easily be accommodated.

The requirements for integrating GTNetS into SimGrid are:

- Translate between SimGrid and GTNetS objects. Since the abstraction models are different between SimGrid and GTNetS, we need to translate SimGrid objects to GTNetS objects so that the network in both simulators are identical.

- Suspend and resume GTNetS objects. SimGrid, as the controller, needs to suspend and resumes GTNetS simulation objects in order to get information from GTNetS at a certain time, but retain full control of the simulation.

- Speculative execution of GTNetS. SimGrid needs to find out what network events may happen next. (For example, SimGrid needs to know when a flow completes.) Therefore, we enable speculative GTNetS simulations that can be rolled back.

In the following, we explain the three requirements above in detail.

5.3.1 Object translation

Objects in SimGrid are translated into those in GTNetS. Since objects in SimGrid are more abstract, we add several attributes to the corresponding objects in GTNetS. There are two translations: topology translation and flow translation.
Table 5.2: Object translation

<table>
<thead>
<tr>
<th>SIMGRID</th>
<th>GTNetS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host: name</td>
<td>Node: address</td>
</tr>
<tr>
<td>Router</td>
<td>Node</td>
</tr>
<tr>
<td>Link: bandwidth, latency</td>
<td>p2pLink: bandwidth, latency</td>
</tr>
<tr>
<td>Path: series of links</td>
<td>Routingtable: static routing</td>
</tr>
</tbody>
</table>

Topology translation:

The GTNetS topology is generated from the configuration file of SIMGRID. Relations between SIMGRID and GTNetS objects are summarized in Table 5.2. Hosts and routers in SIMGRID are translated into nodes in GTNetS. Hosts in SIMGRID are identified by their name, which are specified in the configuration file, and nodes in GTNetS are identified by their IP Addresses; Our interface maintains the relationship between the two. In SIMGRID, a path is defined as a series of links, and routing tables are generated in GTNetS based on the path information. GTNetS implements various routing algorithms, but in order to maintain the correspondence between SIMGRID paths and GTNetS routes, we use only the static routing method in GTNetS.

Flow translation:

Communication (sending and receiving data) between nodes in SIMGRID can be represented as a flow. In GTNetS, several protocols such as TCP and UDP are implemented, but we choose TCP as the corresponding protocol to SIMGRID. GTNetS provides several applications that use a single TCP protocol such as the TCPSend class. We use the TCPServer and the TCPSend classes in GTNetS to represent communication in SIMGRID. The TCP protocol has several parameters, but because SIMGRID does not have all the information, we use the values in Table 5.3 as the defaults. Our interface provides functions to change these parameters.

5.3.2 Interrupting and resuming GTNetS

In order to control GTNetS, SIMGRID needs to suspend and resume GTNetS. The current GTNetS does not have these functions. (It has a stop function, but this function discards all objects, leaving no possibility of resuming.) We add suspend and resume
Table 5.3: TCP parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP variant</td>
<td>TCP Reno</td>
</tr>
<tr>
<td>Advertised window size</td>
<td>20 KB</td>
</tr>
<tr>
<td>Slow-start threshold size</td>
<td>65 KB</td>
</tr>
<tr>
<td>Transmit buffer size</td>
<td>128 KB</td>
</tr>
<tr>
<td>Receive buffer size</td>
<td>128 KB</td>
</tr>
</tbody>
</table>

capabilities to GTNetS by generating a “suspend” event. When the simulator finds a suspend event in the event queue, it stops processing events, leaving the current event list. In order to resume, the simulator has only to start to process the remaining event list.

5.3.3 Speculative execution of GTNetS

While SIMGRID simulates actions on resources, it needs to know what event will occur next as described in section 5.2.1. There are two situations depending on whether a flow completes before any computation-related events occur. When a flow completes before any computation-related events occur, SIMGRID runs GTNetS until the flow completes. When a computation-related event occurs before a flow completes, SIMGRID runs GTNetS until the computation-related event occurs, and then it may add another flow (action) to GTNetS.

In order to find out which event will occur next, we enable speculative GTNetS simulations. We implement a function in GTNetS that returns the time when a flow completes next. Since we need to know this information without modifying the state of the GTNetS simulation, we fork GTNetS and let the child process proceed with the simulation. When the child completes, it passes the next flow completion time to the parent through a pipe, and exits.

5.4 Interfaces between SIMGRID and GTNetS

In this section, we briefly list the interfaces on top of GTNetS, which are called in SIMGRID.

/* initialize the GTNetS interface and environment */
int gtnets_initialize();
/* add a link */
int gtnets_add_link(int link, double bandwidth, double latency);

/* add a route between a source and a destination as an array */
/* of link indices */
int gtnets_add_route(int src, int dst, int links, int nlinks);

/* create a new flow on a route */
/* one can attach arbitrary metadata to a flow */
int gtnets_create_flow(int src, int dst, int datasize, void* metadata);

/* get the time (double) until a flow completes (the first such flow) */
Time_t gtnets_get_time_to_next_flow_completion();

/* run until a flow completes (returns that flow's metadata) */
int gtnets_run_until_next_flow_completion(void** metadata);

/* run for a given time (double) */
void gtnets_run(Time_t deltat);

/* clean up */
void gtnets_finalize();

5.5 Conclusion

In this chapter, we have described our integration of SIMGRID and GTNetS. The two main challenges were the translation between simulation objects in the two simulators, and the addition of suspendable/resumable simulation and of speculative simulation capabilities to GTNetS. We addressed the latter by modifying the GTNetS source code, and addressed the former by implementing an API on top of the modified GTNetS. The result for SIMGRID users is a new ability to switch easily (i.e., with no user code modification) between analytical and packet-level simulation, thereby enacting the trade-offs identified in the previous chapter.

Note that our integration at this time does not support all network resource management functionalities in SIMGRID. For instance, in SIMGRID the user can cancel TCP flows on the fly. However, our integration with GTNetS does not allow such cancellations. Further modification of GTNetS are conceivable to increase the compatibility between the two simulators, but we leave such modifications for future work.
Chapter 6

Conclusion

6.1 Summary of contributions and impacts

We investigated the trade-offs between simulation cost and accuracy for packet-level and analytical simulations by comparing three existing packet-level network simulation tools, GTNetS, SSFNet, and ns-2, with the analytical simulation models implemented in SIMGRID. We found that the following three factors affect the accuracy of SIMGRID when compared to packet-level simulation: (1) the size of the data transferred by TCP flows, (2) the complexity of the network topology, and (3) the network contention. Overall, our conclusion is that the analytical network simulation implemented in SIMGRID is a viable alternative to packet-level simulation in many relevant scenarios. We found that the network model of SIMGRID does achieve comparable accuracy to the packet-level simulators for large data (larger than 1MB) and low network contention. The error rate tends to become larger when the links are more congested. Our results provide a new understanding of the relative merit of analytical network simulation to packet-level simulation.

In order to enable the SIMGRID users to switch between packet-level and analytical network simulations and make their own decision regarding the trade-offs, we have integrated GTNetS into SIMGRID. The two main challenges were the translation between simulation objects in the two simulators, and the addition of suspend/resume and of speculative execution capabilities to GTNetS. We addressed the latter by modifying the GTNetS source code, and addressed the former by implementing an API on top of the modified GTNetS. The result for SIMGRID users is a new ability to switch easily (i.e., with no user code modification) between analytical and packet-level simulations.
6.2 Future work

There are several ways in which this work can be extended. We envision three important directions for future work.

**New GTNetS modification:** We added functions for SIMGRID to add/suspend/resume flows in GTNetS. However, our integration at this time does not support all network resource management functionalities in SIMGRID. For instance, in SIMGRID the user can cancel TCP flows on the fly. Further modification of GTNetS are conceivable to increase the compatibility between the two simulators.

**Combination of analytical and packet-level simulations:** Currently, the user has to choose either analytical or packet-level simulation. Both simulations have strengths and weaknesses. It would therefore be convenient if SIMGRID could automatically switch between the two so that it uses analytical simulation for flows of large data over low network contention, and uses packet-level simulation for flows of small data or flows over high network contention.

**Comparison with other analytical models:** There are other analytical models that may represent network bandwidth sharing better than the model used by SIMGRID in its current implementation. It may therefore be possible to improve the accuracy of analytical simulation if we were to implement them in SIMGRID. Thanks to our integration with GTNetS, the evaluation of the new models by comparing them with the packet-level simulation would be straightforward.
Appendix A

Single network link
Figure A.1: Throughput (left panels) and relative error in throughput (right panels) for ns-2, SSFNet \((D = 0.2, 0.01)\), GTNetS, and Simgrid as a function of data size for \(\beta = 100\,\text{KB/sec}\). The upper panels show the results for \(\alpha = 0\,\text{ms}\), the middle panels for \(\alpha = 1\,\text{ms}\), and the lower panels for \(\alpha = 100\,\text{ms}\).
Figure A.2: Throughput (left panels) and relative error in throughput (right panels) for ns-2, SSFNet \((D = 0.2, 0.01)\), GTNetS, and SimGRID as a function of data size for \(\beta = 10^3\) KB/sec. The top panels show the results for \(\alpha = 0\) ms, the upper middle panels for \(\alpha = 1\) ms, the lower middle panels for \(\alpha = 10\) ms, and the bottom panels for \(\alpha = 100\) ms.
Figure A.3: Throughput (left panels) and relative error in throughput (right panels) for ns-2, SSFNet ($D = 0.2, 0.01$), GTNetS, and SimGRID as a function of data size for $\beta = 10^4$ KB/sec. The top panels show the results for $\alpha = 0$ ms, the upper middle panels for $\alpha = 1$ ms, the lower middle panels for $\alpha = 10$ ms, and the bottom panels for $\alpha = 100$ ms.
Figure A.4: Throughput (left panels) and relative error in throughput (right panels) for ns-2, SSFNet ($D = 0.2, 0.01$), GTNetS, and SimGRID as a function of data size for $\beta = 10^5$ KB/sec. The upper panels show the results for $\alpha = 0$ ms, the middle panels for $\alpha = 1$ ms, and the lower panels for $\alpha = 100$ ms.
Figure A.5: Throughput (left panels) and relative error in throughput (right panels) for ns-2, SSFNet (\(D = 0.2, 0.01\)), GTNetS, and SimGRID as a function of data size for \(\beta = 10^6\) KB/sec. The top panels show the results for \(\alpha = 0\) ms, the upper middle panels for \(\alpha = 1\) ms, the lower middle panels for \(\alpha = 10\) ms, and the bottom panels for \(\alpha = 100\) ms.
Appendix B

Single link – different window sizes

Figure B.1: Throughput for ns-2, SSFNet \((D = 0.2, 0.01)\), GTNetS, and SimGRID as a function of data size, for \(\beta = 100 \text{ KB/sec}\) (left) and \(\beta = 10^5 \text{ KB/sec}\) (right). The TCP advertised window size is 10 KB. When the data size is larger than \(10^4 \text{ KB}\), the throughput with SimGRID approaches the throughput with ns-2 for \(\beta = 100 \text{ KB/sec}\). The throughput with SimGRID approaches the throughput with GTNetS and SSFNet.
Figure B.2: Same as Figure B.1, but the window size is 30 KB. The trends of the throughputs are similar to those in Figure B.1.
Figure B.3: Throughput with three different TCP advertised window sizes, 10 K, 20 K, and 30 K, for SIMGRID (upper panels), ns-2 (middle), GTNetS (lower) as a function of data size. Left panels are for $\beta = 100\text{ KB}$ and right panels for $\beta = 10^5\text{ KB}$. For all the simulators, when $\beta = 100\text{ KB}$, the throughput does not depend on the window size. When $\beta = 10^5\text{ KB}$ (latency bound case), the throughput varies depending on the window size: higher throughput with larger window size.
Figure B.4: Same as Figure B.3, but for SSFNet ($D = 0.2$ (upper panel), 0.01 (lower)). The trend of the throughputs are similar to those in Figure B.3.
Appendix C

Multiple flows on a single link

Figure C.1: Throughput for ns-2, SSFNet, GTNets, and SimGRID as a function of bandwidth, for the number of flows is 1 (upper left), 2 (upper right), 16 (lower left), and 32 (lower right). $\alpha = 0$ ms and $S = 10^5$ KB. Average throughputs are shown for the experiments with more than one flows.
Figure C.2: Same as Figure C.1, but for $\alpha = 1$ ms.
Figure C.3: Same as Figure C.1, but for $\alpha = 10\text{ ms}$. 
Figure C.4: Same as Figure C.1, but for $\alpha = 100$ ms.
Appendix D

Dogbone topology
Figure D.1: Throughput for flow A (upper), flow B (middle), and error in throughput (lower) for $\beta = 10$ KB/sec.
Figure D.2: Throughput for flow A (upper), flow B (middle), and error in throughput (lower) for $\beta = 10^3$ KB/sec.
Figure D.3: Throughput for flow A (upper), flow B (middle), and error in throughput (lower) for $\beta = 10^4$ KB/sec.
Figure D.4: Throughput for flow A (upper), flow B (middle), and error in throughput (lower) for $\beta = 10^6$ KB/sec.
Appendix E

Random topology
Figure E.1: Relative error in throughput of SimGrid for $S = 10^5$ KB. The left panels show the error for $N_f = 200$, and the right panels for $N_f = 100$. The upper panels show the error for case 1, the middle panels for case 2, and the lower panels for case 3. Each curve corresponds to one experiment. Each panel includes ten curves.
Figure E.2: Relative error in throughput of SimGrid for $S = 100$ KB. The left panels show the error for $N_f = 200$, and the right panels for $N_f = 100$. The upper panels show the error for case 1, the middle panels for case 2, and the lower panels for case 3. Each curve corresponds to one experiment. Each panel includes ten curves.
Bibliography


