Speed and Accuracy of Network Simulation in the SimGrid Framework[*]

Kayo Fujiwara
Information and Computer Sciences Dept.
University of Hawai‘i at Manoa
1680 East-West Rd, POST 317
Honolulu, HI 96822, U.S.A.
kayof@hawaii.edu

Henri Casanova
Information and Computer Sciences Dept.
University of Hawai‘i at Manoa
1680 East-West Rd, POST 317
Honolulu, HI 96822, U.S.A.
henric@hawaii.edu

ABSTRACT

The experimental study of distributed systems and algorithms for large-scale internet platforms typically requires simulation of compute and network resources. This paper focuses on network simulation issues. While many packet-level network simulators are available and enable high-accuracy simulation, they can lead to prohibitively long simulation times. Consequently, a number of simulation frameworks have been developed that simulate networks at higher levels, thus enabling fast simulation but losing accuracy. One such framework, SimGrid, uses a flow-level approach that approximates the behavior of TCP networks, including TCP's bandwidth sharing properties.

In this paper we perform a quantitative evaluation of SimGrid and compare it to popular packet-level simulators. We identify the regimes in which SimGrid's accuracy is comparable to that of these packet-level simulators, and the regimes in which SimGrid's accuracy may not be acceptable. We then describe an integration of the GTNetS packet-level simulator and SimGrid, which allows SimGrid users to easily opt either for fast but potentially inaccurate flow-level simulation or for accurate but potentially prohibitively slow packet-level simulation.

Categories and Subject Descriptors

I.6.4 [**Model Validation and Analysis**]: Network Simulation; I.6.7 [**Simulation Support System**]: SimGrid, ns-2, SSFNet, GTNetS

General Terms

Experimentation, Measurement, Performance

1. INTRODUCTION

Large-scale distributed systems, that is, systems that consist of hundreds to tens of thousands of hosts on wide-area networks, have become feasible due to improvements in networking and middleware technology. These systems provide tremendous opportunity in many domains and have already been deployed for particular applications. Example applications include distributed file sharing on peer-to-peer networks, distributed volunteer computing using idle cycles of home or office computers, deployment of grid platforms to support collaborative e-Science applications, and distributed gaming. In spite of these successful deployments, many research challenges remain for achieving higher levels of scalability, efficiency, and resilience in a view to better support current applications as well as to enable future ones. These challenges boil down to the development of better systems (e.g., better overlay topologies for peer-to-peer systems, better resource management architectures in grid platforms) and of better algorithms (e.g., more advanced search algorithms in peer-to-peer networks, better scheduling algorithms for e-Science applications).

To investigate the relative merit of competing designs for the aforementioned systems and/or algorithms, one must conduct experiments. Indeed, the systems are often so complex and the algorithms so often relying on heuristics, that analytical comparisons are at best possible only with very stringent and ultimately unrealistic assumptions. An approach to obtain valid experimental results is to conduct experiments on production platforms, or at least on large testbeds. Unfortunately, this approach often proves infeasible. Real-world platforms may not be available for experimentation purposes so as not to disrupt production usage. Experiments are often non-repeatable due to resource dynamics (e.g., time-varying and non-deterministic network usage, unpredictable host failures). Even if a stable platform is available, experiments can only be conducted for the platform configuration at hand, which may be not sufficient to gain all necessary insight in the relative effectiveness of various distributed system or algorithm designs. Finally, experiments on real-world platforms may be prohibitively time consuming especially if large numbers of experiments are needed to explore many scenarios with reasonable statistical significance. Given all these difficulties, while researchers strive to obtain experimental results in real-world systems, the majority of published results are obtained in simulation.

Simulating a large-scale distributed system and its accompanying distributed algorithm(s) entails the simulation of

[*]This work is supported in part by the National Science Foundation under award 0546688.

NSTools '07, October 22, 2007, Nantes, France
Copyright 2007 ICST 978-963-9799-00-4.

both compute and network resources. In this work we focus mostly on the network simulation component. The network community has employed simulation for many years in order to study network protocols. The standard approach is to use *packet-level network simulators* that use event-driven simulation of individual packets going from end-point to end-point via network links and routers. While these simulators make it possible to prototype and study new network protocols, they typically also provide full implementations of existing protocols such as TCP or UDP. Therefore, these simulators are usable for practical distributed systems simulation purposes. For instance, the work in [10] builds a simulator of the Gnutella peer-to-peer network based on the well-known ns-2 [22] packet-level simulator. It is widely acknowledged that packet-level simulators lead to accurate simulation results because they capture virtually all aspects of underlying network protocols.

The main problem with packet-level simulation is that simulation speed is low, with simulation times possibly orders of magnitude larger than simulated times. Expectedly, simulation time increases with the number of packets to simulate [15]. For some simulation domains, the number of packets is not large and thus does not lead to prohibitively long simulation times. For instance, many simulations of peer-to-peer networks consist in simulating overlay network routing protocols by which peers exchange short messages that consist of only a few packets. Therefore, in spite of large numbers of end-points and of routers, packet-level simulation leads to acceptable execution times. Unfortunately, this is not true in all domains. For instance, consider the work in [30] that studies data caching and replication schemes for e-Science applications in grid platforms. The authors ran 72 simulation experiments to explore scenarios in which data files between 500MB and 2GB are communicated among 30 sites on an internet overlay network for 6000 jobs requiring data. For such simulations, which we term *grid simulations* for lack of a better term, using a packet-level approach would require a prohibitive amount of time due to the enormous number of individual packets. As a result, the authors in [30] developed their own ad-hoc simulator, which does not attempt to model TCP behavior although their work ultimately targets TCP networks. For their particular simulations the use of this ad-hoc simulator is justifiable because their overlay network topology is hierarchical, is congruent to the hierarchy of the physical network, and flows sharing links and routers are between the same endpoints. However, their simulator is not applicable if one hopes to capture the behavior of a TCP network in general.

As seen in the discussion above, there is a clear need for network simulation technology that is orders of magnitude faster than packet-level simulation and that captures the behavior of real networks well. In this work we focus on TCP networks. The alternative to packet-level simulation, which we term *flow-level simulation*, consists in reasoning directly about TCP flows and in computing their throughputs within a theoretical framework. Flow-level simulation is orders of magnitude faster than packet-level simulation. Unfortunately it makes several simplifying assumptions that do not hold in real networks (e.g., ignoring TCP slow-start), therefore reducing accuracy.

A simulation tool, SimGrid [13], has gained prominence in the community of researchers that study distributed systems and algorithms in the broad area of grid computing. Part of its acceptance by an increasing number of users stems from its fast simulation capability. Indeed, due to its use of a flow-level network simulation approach, SimGrid makes it possible to perform large numbers of statistically significant experiments even on large TCP networks. A crucial question is whether the accuracy of SimGrid network simulations is "reasonable" when compared to that of packet-level simulations. To answer this question we make the following contributions in this paper:

1. We perform a qualitative and quantitative evaluation of the flow-level network simulation component of SimGrid by comparing it to packet-level simulators;
2. We identify the regimes in which the accuracy of the flow-level approach is significantly worse than that of the packet-level approach; and
3. We enhance the SimGrid tool by integrating it with the GTNetS packet-level simulator [29] so that users can opt for either packet-level or flow-level simulation seamlessly.

This paper is organized as follows. In Section 2 we discuss related work and further motivate our work. Section 3 presents the results of our evaluation. Section 4 describes our integration of SimGrid and GTNetS. Finally, Section 5 concludes the paper with a discussion of future directions.

2. RELATED WORK

In this section we briefly review existing packet-level simulators, endeavors to improve the speed of packet-level simulation, and non-packet-level simulators used by researchers in the area of grid computing. We then give some details about network simulation in the SimGrid simulator as it is the target of our evaluation. Finally we review works that, like ours, have compared accuracy of packet-level and flow-level simulation, including works that have attempted to combine the two approaches.

2.1 Packet-level Simulation

Packet-level simulators use discrete-event simulation by which a flow over a network path can be represented as a sequence of events, such as packet arrivals and departures at end-points and routers. End-points and routers both implement full-fledge network protocols. Simulation time typically increases in proportion to the number of events [15]. Popular such simulators include ns-2 [22], GTNetS [29], and SSFNet [6]. As mentioned in Section 1, the main problem with these simulators is that simulation time can be orders of magnitude larger than simulated time for simulations that involve realistic topologies with many flows. For instance, using GTNetS, which is known for good scalability, simulating 200 flows each transferring 100MB between two random end-points in a random 200-node topology for 125 sec of simulated time takes approximately 1500 sec on a 3.2GHz Xeon processor. While this is acceptable for researchers studying network protocols, it is prohibitive for many researchers studying distributed systems and algorithms on large-scale platforms for application that are long-running and/or that involve large amounts of communication. This problem is often compounded by the need to rely on results from over thousands of simulation experiments to compute valid statistics regarding the relative effectiveness of competing algorithms (see for instance the scheduling study in [23], which uses over one million simulation experiments, with each experiments requiring over 1,000 sec of simulated time).

Several researchers have attempted to increase the speed of packet-level simulations. For instance, in [16] the authors developed the MaSSF framework, which combines the DaSSF packet-level simulator [5] with message passing [33] to accelerate and increase the scalability of network simulation by running in parallel on large clusters of workstations. MaSSF is the main component of the MicroGrid [34] tool for simulating Grid platforms and applications. Others have proposed emulation techniques by which traffic flows on physical devices, introducing delay, bandwidth and packet loss characteristics of the network to be simulated. A well-known example of such work is ModelNet [38].

While the above works do increase the speed and scalability of packet-level simulation without compromising simulation accuracy, many users performing grid simulations need simulations orders of magnitude faster. In the face of such requirements, simulators that relax the notion of a packet were developed. For instance, the Bricks simulator [36] uses ideal queuing networks to simulate real networks. While the user can specify a packet size in this simulator, Bricks packets do not correspond to real network packets and Bricks does not implement real network protocols. Large packets lead to fast simulation but obviously low accuracy (in the extreme, multi-path network communications use a store-and-forward approach with no pipelining across network links). Although lower packet size leads to behavior presumably qualitatively closer to that of real networks, nothing in this simulator ensure that the behavior is quantitatively close to that of, for instance, TCP. Another simulator, GridSim [35] implements a protocol that includes some elements of UDP and allows for variable packet size. Like Bricks, GridSim requires small packet size to hope to gain accuracy close to that of true packet-level simulators on realistic network topologies, but then suffers from high simulation costs. Many other "grid" simulators exist, such as OptorSim [2], GangSim [7], Neko [25], or HyperSim [27] (see [28] for a survey). All implement some network model, but to the best of our knowledge (i.e., based on publications and/or on inspection of source codes), these simulators either use packet-level simulation or do not attempt to implement a model that realistically tracks the behavior of TCP networks.

2.2 Flow-level Simulation

To increase the speed of network simulation one approach is to use theoretical models to compute the throughput of each flow in a network topology at a given time. Models have been proposed [26, 19, 24] that model the throughput of a TCP flow as a function of packet loss and round trip delay, as well as some parameters of the network and of the TCP protocol. Unfortunately, some of these parameters are difficult to measure and/or instantiate for the purpose of grid simulations. Furthermore, it is not clear how the model can be applied to arbitrary network topologies with many simulated flows competing for network resources. Instead, one desires reasonable models that capture the bandwidth sharing behavior induced by TCP among flows on arbitrary topologies and that are defined by a few simple parameters, namely link physical bandwidths and TCP congestion window size. This notion of *macroscopic* models of bandwidth sharing is challenging [18]. A key question is: which type of "fairness" does TCP implement when flows share bandwidth on bottleneck links? The first widely recognized model is the simple Max-Min fairness model [1], which computes a bandwidth allocation in which increasing the allocation of a flow would require decreasing the allocation of another. However, it is well-known that TCP does not implement Max-Min fairness, as shown for instance by Chiu [4]. Indeed, the analytical models for TCP throughput in [8, 26] approximate the throughput, $B(p)$, to

$$B(p) = \frac{c}{\mathrm{RTT}\sqrt{p}},$$

where p is the fraction of packets lost, RTT is the round-trip time (RTT), and c is some constant, provided that p is not "too high". Assuming that all flows experience the same loss rate, p, this formula suggests that bandwidth is in fact shared in inverse proportion to the RTT. This thus suggests a Max-Min scheme that is modified to account for flow RTTs.

Based on the above considerations, the designer of the SimGrid simulation tool [13, 32], have opted for a RTT-aware Max-Min flow-level model. (See Section 5 for a discussion of other promising models of fairness, such as proportional fairness.) In this model, the bandwidths allocated to flows competing over a bottleneck link is inversely proportional to the flows' RTTs. (A link is defined as a bottleneck if the sum of the bandwidths allocated to the flows over this link is equal to the total bandwidth of the link.) We refer the reader to [3] for full details on the model and for initial validation results via which this particular model was selected among several alternatives. The model is instantiated solely based on network link physical characteristics (latencies and bandwidths) and on the size of the TCP congestion window size. As a result, SimGrid is, to the best of our knowledge, the first simulation framework designed for the study of distributed systems and algorithms for large-scale platforms that uses a flow-level network simulation approach that attempts to capture the true behavior of TCP networks and that decreases simulation costs by orders of magnitude when compared to packet-level simulation.

2.3 Flow-level Simulation Accuracy

Although SimGrid has garnered a sizeable user base, its flow-level network simulation scheme has several limitations. It does not capture the slow-start feature of TCP and instead assumes that flows are backlogged and in steady state. Also, it assumes that the RTT of each flow is constant and is equal to twice the sum of the link latencies on the flow's network path. And of course, the model cannot account for any detail pertaining to the behavior of individual packets. Therefore, one may wonder how accurate the network simulation in SimGrid is. For instance, the simulation of short TCP flows should incur large inaccuracies since slow-start is not accounted for. In this paper we quantify such inaccuracy and identify the regimes in which simulation results deviate significantly from those of packet-level simulation.

Other works present quantitative comparisons between packet-level non packet-level simulation approaches, and are therefore related to this work. Liu et al. [15] evaluated the relative performance of a sophisticated fluid-based simulation scheme over packet-level simulation. Their results show that fluid models usually outperform packet-level simulators. However, when flows compete for resources, which increases computation, fluid models may be less efficient than packet-level simulation. Nicol et al. [21] also studied the performance/accuracy trade-offs between fluid models and packet-level simulation. Their results showed that the relative error of fluid simulation is very small compared to the results of packet-level simulation. Nevertheless, the cost of their fluid simulation would be prohibitive if applied for simulating long-running large-scale distributed applications, such as the ones studied by SimGrid users. Schwefel et al. [31] compared three different analytical TCP models and their results showed that all of the analytical models have deficiencies in some scenarios for which their assumptions are violated. The analytical TCP models in their study were instantiated using parameters such as loss rate, which were themselves determined through simulation experiments. The network model used in SimGrid is simpler than these models as it uses fewer parameters, which makes it straightforward for users to instantiate but also mandates our study of its accuracy.

Finally, in [12] Kim combines packet-level with so-called network calculus simulation. He achieves accuracy within a few percents of packet-level simulation with an increase in simulation speed up to a factor 20. While a factor 20 is impressive, SimGrid achieves factors up to 4 orders of magnitude. For instance, for the same 200-node topology example as in Section 2.1, while the GTNetS simulation takes approximately 1500 sec SimGrid performs the same simulation in approximately 0.5 sec. Such speed is key to SimGrid's acceptance by many of its users. Nevertheless, it could be interesting for SimGrid to use the network calculus approach proposed in [12]. Later in this paper we describe how we integrate GTNetS and SimGrid to give users the opportunity to easily opt for either packet-level or flow-level approaches at the onset of the simulation. An integration with the network calculus approach in [12] would provide users with yet another option for trading off simulation speed and simulation accuracy.

3. EXPERIMENTAL EVALUATION

In this section we present experimental results quantifying network simulation speed and accuracy for SimGrid and three packet-level simulators: ns-2 [22], SSFNet [6], and GTNetS [29]. All simulators are configured to use the same TCP window size (20KB by default in our experiments). We ran experiments with TCP window sizes of 10K and 30K but we obtained similar results with all three. TCP Reno is used as the TCP model (without the use of a delayed ack algorithm, which is the default behavior for ns-2 and GTNetS). In all our experiments we ignored receiver window limitations, and the communications are backlogged.

We performed three types of experiments:

1. Experiments with one or more flows on one-link topologies;

2. Experiments with two flows on classical 5-link topologies; and

3. Experiments with multiple flows on random topologies.

We summarize key results in the following three sections, and we refer to the Master's thesis of the first author for full details [9].

3.1 One-Link Topology

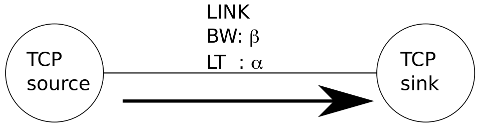

Figure 1: One-link topology with a single TCP flow.

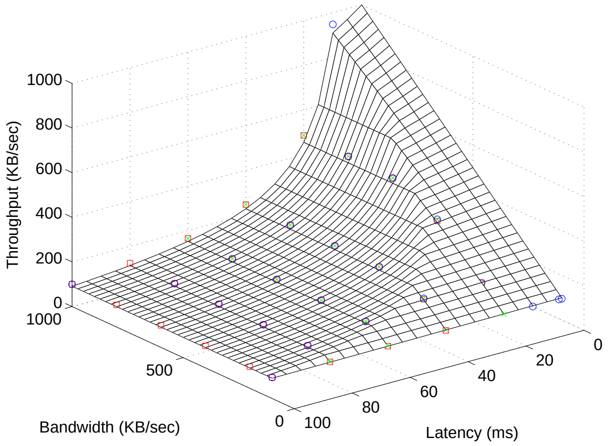

Figure 2: Single flow throughput as a function of link latency (α) and bandwidth (β), for a transferred data size S = 100MB. The mesh shows the throughput computed by the SimGrid model. Squares (□) show the throughput simulated by GTNetS; Circles (○) by ns-2; and Crosses (×) by SSFNet. Many data points are identical and thus overlapped.

Our first set of experiments is for a single TCP flow on the topology depicted in Figure 1. The TCP sender (the "source") sends S bytes of data to the receiver (the "sink") over a single link with latency α and bandwidth β. For each simulator we compute the flow's throughput, that is, the ratio of S to the time for the flow to complete. Figure 2 shows a 3D plot of the throughput (z axis) as a function of α and β, for S=100MB. The mesh is the throughput obtained by SimGrid. It turns out that in this case SimGrid computes the throughput analytically as:

$$\mathit{Throughput} = \frac{S}{\alpha + S/\min(\beta, \frac{W}{2\alpha})}, \tag{1}$$

where W is the TCP congestion window size. This formula is simply the ratio of the data size to the transfer time. The transfer time is equal to the latency plus the data size divided by the effective bandwidth. The effective bandwidth, $\min(\beta, W/(2\alpha))$ is bounded above by the physical bandwidth and by the TCP window size divided by the RTT. The data points on the mesh correspond to the throughput simulated by the three packet-level simulators. We see on the figure that the throughputs computed by SimGrid are very close to those simulated by the packet-level simulators, with relative differences between them under 1%.

While the results shown in Figure 2 indicate that SimGrid achieves accuracy comparable to the packet-level simulators, these results are for a data size of 100MB. As mentioned in Section 2.3, the network model in SimGrid assumes that

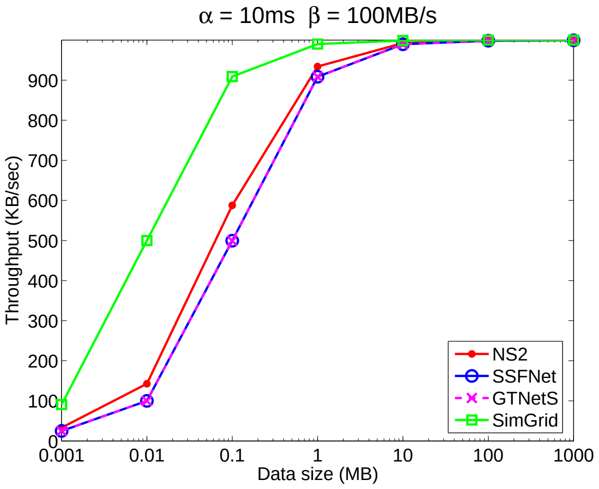

Figure 3: Throughput versus data size as simulated by SimGrid, ns-2, GTNetS, and SSFNet, for $\alpha =$ 10ms and $\beta =$ 100MB/sec.

TCP flows are in steady-state so that the impact of the TCP slow-start mechanism is negligible. This assumption is realistic for large data sizes, but breaks down for small data sizes. Figure 3 plots the throughput versus the data size for all simulators. The plot is for two particular values of α and β, but similar plots were obtained for other values of these two parameters. We see that, indeed, for small data sizes the throughput by the SimGrid simulation is much higher (e.g., a factor 5) than those by the packet-level simulations. When data size is larger than 10MB, the relative difference between the throughputs becomes lower than 1%. Note that although the packet-level simulators lead to similar results, they are not in perfect agreement.

Another interesting result, not shown in the figure, is that if $\beta \leq \frac{W}{2\alpha}$, meaning that the minimum in Eq. 1 is equal to β, the relative difference between the throughput achieved by SimGrid and that of the packet-level simulators is a few percent ($\sim$ 5%), even for large data sizes. This corresponds to cases when the throughput is bounded by the link's physical bandwidth. When the throughput is instead bounded by latency, that is, when $\frac{W}{2\alpha} \leq \beta$, as is the case for the results shown in Figure 3, then this relative difference is equal to zero. This is consistent with the results in the next section.

Finally, when adding multiple flows on the link results remain essentially unchanged in terms of the relative difference between SimGrid and the packet-level simulators. In the following two sections, to avoid clutter, we omit results obtained with ns-2 and SSFNet. From now on we term the relative difference between throughputs computed by SimGrid and those simulated by GTNetS the *relative error* of SimGrid, assuming for the sake of discussion that GTNetS results are perfectly realistic. We also limit results to data sizes of 100MB to isolate other sources of discrepancy between SimGrid and SSFNet from the TCP slow-start issue discussed above. (Recall that full results are available in [9].)

3.2 Dumbbell Topology

In this section we present results for two TCP flows, A and B, on a dumbbell topology defined by two parameters, α and β, as depicted in Figure 4. This topology, which is sometimes referred to as a "dogbone", is used in many TCP studies to evaluate the basic bandwidth sharing behavior of flows that share a bottleneck link [14].

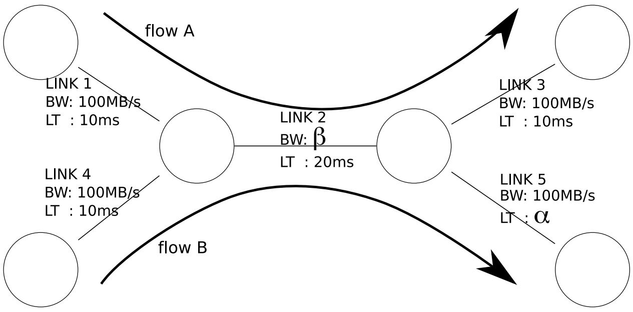

Figure 4: Dumbbell topology with two TCP flows, A and B. All bandwidths and latencies are fixed, but for β, the bandwidth of the middle link, and for α, the latency of the bottom right link.

Figure 5 shows the throughputs for flow A and flow B, as computed by SimGrid and simulated by GTNetS, versus α, for $\beta =$ 100MB/sec and a transferred data size of 100MB. We can see in the figure that both SimGrid and GTNetS are in perfect agreement. Figure 6 shows the same results for $\beta =$ 100KB/sec, which corresponds to the case in which the throughput is bounded by the physical bandwidth β rather than by $\frac{W}{2\alpha}$. As discussed for the one-link topology in Section 3.1 we observe a discrepancy between SimGrid and GTNetS. However, both simulators follow similar trends (e.g., the flow with the larger RTT achieves a lower throughput than the flow with the lower RTT).

3.3 Random Topology

Our previous two sets of experiments have highlighted expected flaws of flow-level simulations, namely that they do not account for TCP slow-start, as well as the lack of accuracy in the case of throughput limited by physical bandwidths rather than by latencies. While this is interesting, it is difficult to envision what the impact would be for the simulation of an application over many hosts in a multi-router network topology with many TCP flows. To quantify this impact we perform the following experiments. We generate random but representative internet topologies using the BRITE [20] topology generator. We then generate flows between random pairs of end-points in the topology, which all start simultaneously and communicate 100MB of data. We then measure the transfer time and thus the achieved throughput for each flow. We repeated each experiment 10 times, but only show data for individual experiments here (it turns out that results are essentially unchanged across repetitions).

Figure 7 shows the SimGrid relative error for three experiments with 200 flows. Negative (resp. positive) values mean that SimGrid computes throughputs that are lower (resp. higher) than those obtained with GTNetS. In the case of a 200-node topology and high bandwidths (the top plot), we see that for 90% of the flows SimGrid achieves relative errors under 10%. The middle plot shows that these errors increase when the bandwidths decrease (70% of flows

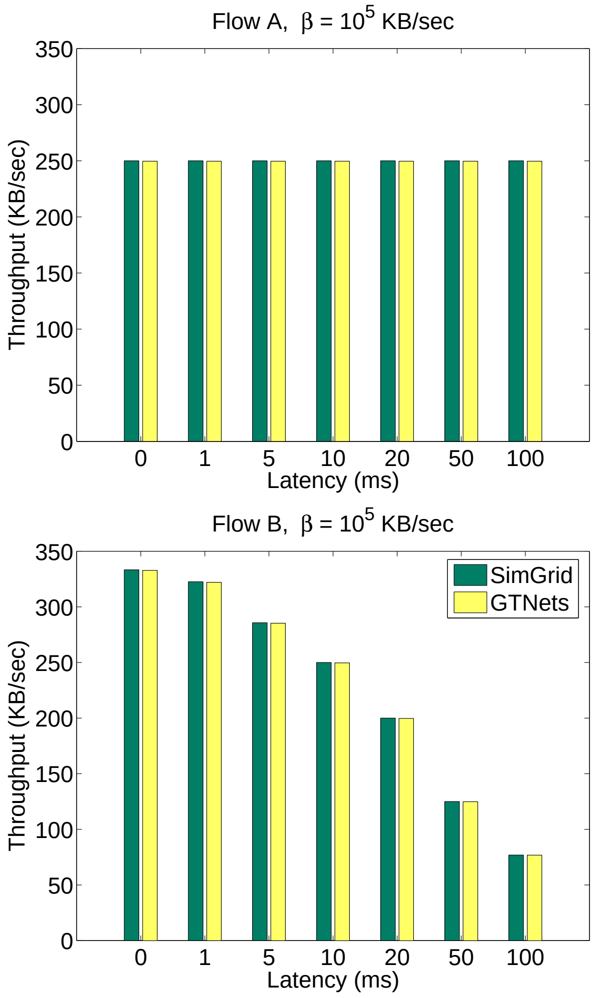

Figure 5: Throughput for SimGrid and GTNetS versus latency, α, with β=100MB/sec, S = 100MB, for the dumbbell topology, which corresponds to a latency-bound scenario. The upper panel shows the throughput for flow A and the lower panel for flow B.

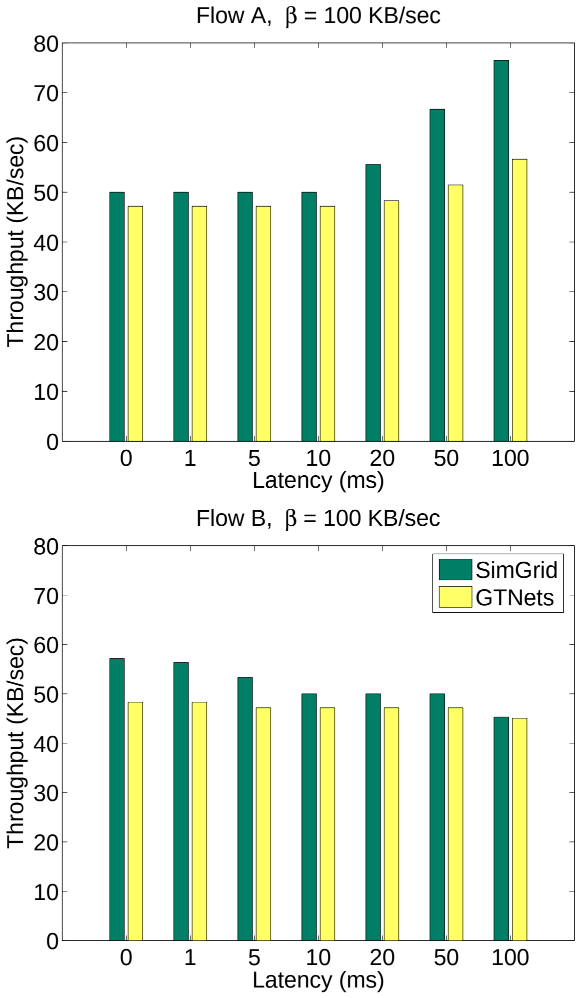

Figure 6: Throughput for SimGrid and GTNetS versus latency, α, with β=100KB/sec, S = 100MB, for the dumbbell topology, which corresponds to a bandwidth-bound scenario. The upper panel shows the throughput for flow A and the lower panel for flow B.

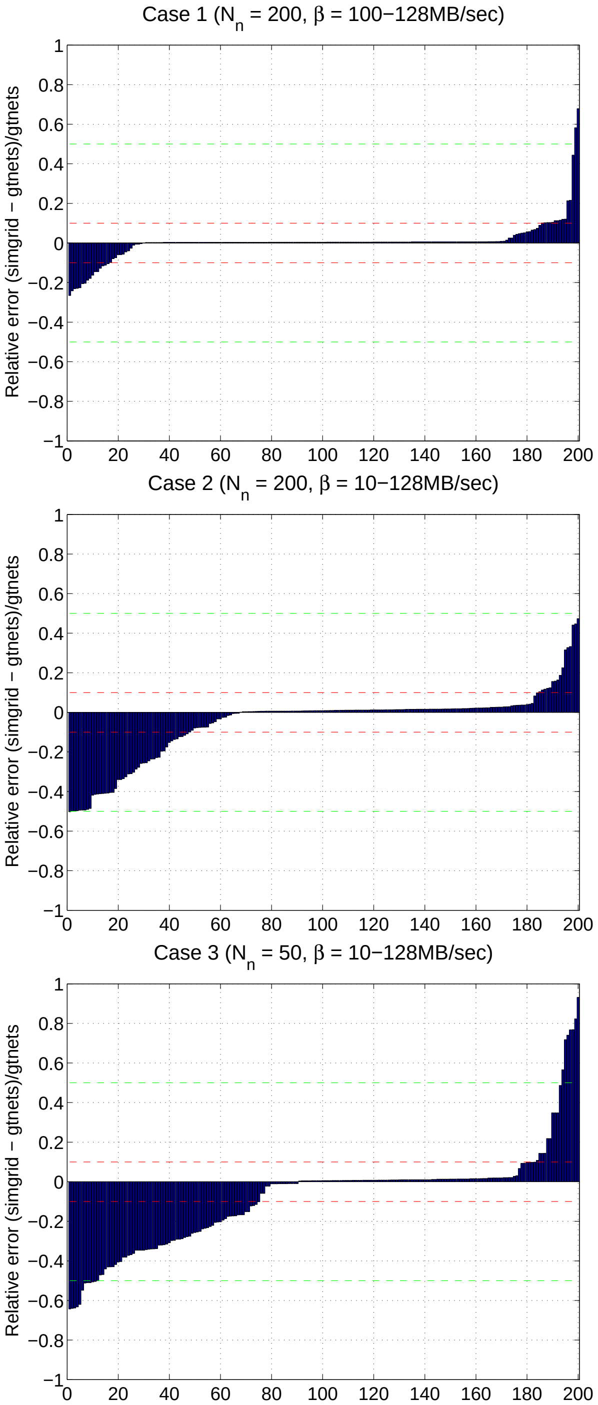

Figure 7: SimGrid's relative error for 200 flows on random topologies, where flows are shown on the x axis, sorted by increasing error. The upper plot shows the error for an experiment with $N_n = 200$ nodes and high bandwidths between 100MB/sec and 128MB/sec. The plot below it shows the error for an experiments in which bandwidths are lower, between 10MB/sec and 128MB/sec. The bottom plot shows the error for an experiment with the lower bandwidths but for a topology with $N_n = 50$ nodes. The horizontal dash lines correspond to relative errors between -10% and $+10\%$, and between -50% and $+50\%$.

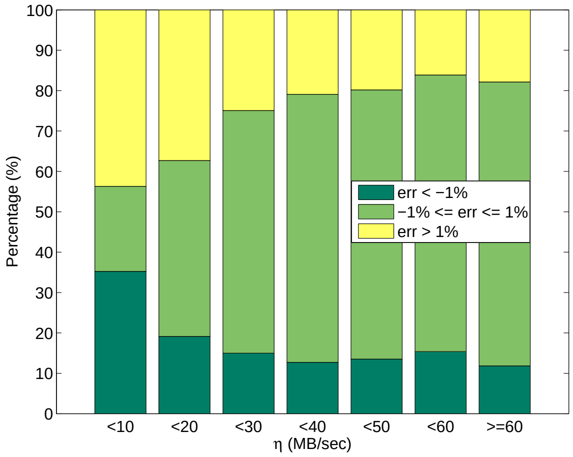

Figure 8: Distribution of the relative error between SimGrid and GTNetS, averaged over all flows, versus η, into three categories: lower than -1%, between -1% and $+1\%$, and above $+1\%$.

have relative errors below 10%). The bottom plot shows that these errors increase further (50% of flows have relative errors below 10%) when the number of nodes decreases. From theses results and others like them we hypothesized that the relative error between SimGrid and GTNetS increases as network contention increases.

To validate this hypothesis further, we present the results in another fashion. We define η as the minimum ratio between the physical bandwidth of a link and the number of flows that use that link, over all links in the topology. η thus quantifies network contention, with high values corresponding to high bandwidths and/or few flows, and low values corresponding to low bandwidths and/or many flows. Figure 8 presents the distribution of relative errors in three categories: within 1%, below -1%, and above 1%, versus η. We see that, indeed, as network contention decreases, that is, as η increases, the fraction of flows whose relative error is between -1% and $+1\%$ increases. We refer the reader to [9] for other results that further confirm our hypothesis.

While our results so far have focused on highlighting scenarios in which SimGrid leads to reduced accuracy when compared to GTNetS, a compelling advantage of SimGrid is high simulation speed. Table 1 shows that, even for our modest 200-node topology and for TCP flows that transfer only 100MB of data, SimGrid achieves simulation speed higher than GTNetS by factors 3,000 and above. This corresponds to a major shift from packet-level simulation, whose simulation time is orders of magnitude larger than simulated time, to flow-level simulation, whose simulation time is orders of magnitude lower than simulated time. Note that SimGrid simulations can become arbitrarily faster than GTNetS simulations when amounts of transferred data become arbitrarily large.

4. SIMGRID/GTNetS INTEGRATION

The results in the previous section illuminate the trade-off between simulation accuracy and speed for packet-level and flow-level network simulation. For instance, when simulat-

Table 1: Simulated time, simulation time, and their ratios observed with GTNetS and SimGrid for a 200-node random topology with various numbers of flows that transfer 100MB of data. The measurements are repeated ten times, and the average is taken. Results are obtained on a 3.2GHz Xeon processor.

# of flows	GTNetS			SimGrid		
	simulated time (sec)	simulation time (sec)	ratio	simulated time (sec)	simulation time (sec)	ratio
10	70.31	65.34	0.92	72.72	0.00176	0.00002
25	87.77	163.10	1.86	80.67	0.00836	0.00010
50	93.73	364.67	3.89	98.44	0.02839	0.00029
100	93.19	753.42	8.08	97.21	0.13810	0.00142
200	124.10	1562.90	12.59	134.06	0.53833	0.00402

ing an application in which flows transfer more than 10MB of data and in which links are not too heavily contended, then using SimGrid is most likely acceptable in terms of accuracy and is orders of magnitude faster (this is the case for many grid simulations). However, for an application in which flows exchange small amounts of data, then packet-level simulation is needed for its accuracy (this is likely the case for many peer-to-peer simulations). In order to enable the use of SimGrid for both usages, we integrate GTNetS as part of SimGrid. Note that our work enables SimGrid users to opt for either packet-level or flow-level network simulation but not a combination of both. Indeed, the latter raises many difficult challenges and is outside the scope of this work. Details of our implementation are available in [9] and we focus here on key technical aspects.

The main challenge for the integration is that SimGrid simulates both computations and communications. SimGrid, like GTNetS, is event-driven and thus must repeatedly determine the next event in this simulation. This next even may be either the completion of a computational task (or task) or the completion of a flow. Therefore, although SimGrid treats GTNetS as a black box, it must carefully control the GTNetS simulation. To enable this control, which will be detailed shortly, we had to modify GTNetS to add to it three new functionalities:

Functionality #1: The ability to run a GTNetS simulation until a flow completes, after which the simulation is suspended;

Functionality #2: The ability to run a GTNetS simulation for a given lapse of simulated time, after which the simulation is suspended;

Functionality #3: The ability to resume a suspended GTNetS simulation.

These modifications to GTNetS required altering under 100 lines of code, essentially to create a "stop" event and to implement functions to insert this event in the GTNetS event queue, and to remove it upon resuming the simulation.

In its original implementation, SimGrid uses the same framework to simulate computations and communications. This framework makes it easy to determine whether the next event is a task completion or a flow completion and the date of this event without changing the state of the simulation. This ability, which stems from the fact that SimGrid uses analytical simulation models both for computations and computations, is key for allowing the simulation to progress correctly. In our integration with GTNetS we have modified SimGrid so that the user can replace the original SimGrid network simulation component with GTNetS (via a single parameter to the SimGrid API function that initializes the simulation). The difficulty is then to determine the date of the next flow completion. To determine this date SimGrid uses functionality #1 above but runs the GTNetS simulation "speculatively": SimGrid saves the state of the GTNetS simulator, runs the GTNetS simulation to determine the date of the next flow completion, and reverts to the saved state. It turns out that speculative execution is straightforward to implement by cloning the GTNetS simulation with the `fork()` system call, letting the child simulation proceed until the next flow completion, retrieving the date of that completion from the child via a pipe, and terminating the child.

After the speculative GTNetS execution, SimGrid has determined the date of the next flow completion. Using its original method SimGrid has also determined the date of the next task completion. Both dates have been determined without changing the simulation state. If the next flow completion occurs after the next task completion, then SimGrid runs the GTNetS simulation only until the next computation completion (using functionality #2), in case the computation completion leads to the creation of a new flow that would change the state of the network simulation. If instead the next flow completion occurs before the next computation completion, then SimGrid runs the GTNetS simulation until the next flow completion, using functionality #1 again. After all necessary bookkeeping and handling of the simulation of compute resources, this process is repeated so that the simulation can progress until the next event. This requires that SimGrid resume the GTNetS simulation (using functionality #3).

With this integration of SimGrid and GTNetS, SimGrid users retain all SimGrid capabilities, including the simulation of both compute and network resources, but have the added ability to opt for flow-level or packet-level network simulation at the onset of the simulation. This is done seamlessly without any modification to the user's simulation code. Note that speculative execution of the GTNetS simulation increases the network simulation time by roughly a factor 2 (i.e., the times in the GTNetS portion of Table 1 are multiplied by 2 when using GTNetS packet-level simulation within SimGrid).

5. CONCLUSION

In this paper we have compared the speed and the accuracy of packet-level and flow-level network simulation. The overriding question is whether flow-level approaches, which are faster by orders of magnitude, lead to reasonable accuracy when compared to their packet-level counterparts. Our study compared the flow-level simulation implemented in the SimGrid simulation framework to three packet-level simulators: ns-2 [22], SSFNet [6], and GTNetS [29]. We found that in many scenarios SimGrid leads to acceptable accuracy when compared to packet-level simulators. We also found that this accuracy would likely becomes unacceptable for many users when data sizes are small (due to the TCP slow-start mechanism) or when networks are highly contended (i.e., low physical bandwidths and/or many flows). We also observed decreases in accuracy when throughput on bottleneck links is bounded by physical bandwidths rather than by latencies. To enables users to cope with such scenarios, and to trade simulation speed for simulation accuracy, we have integrated the GTNetS simulator as part of SimGrid. This integration will be part of an upcoming SimGrid software release.

One interesting new direction for this work is the integration into SimGrid of network calculus schemes as proposed in [12], to increase simulation speed by a factor $\sim$20 while retaining most of the accuracy when compared to pure packet-level simulation. This would be of interest to a subset of the SimGrid users, although many of them require much more than a factor 20 increase in speed from packet-level simulators. One practical issue is that the work in [12] was done as part of the ns-2 simulator, which is notoriously difficult to integrate with other software tools (for instance, while the first version of MicroGrid [34] used ns-2, the developers switched to DaSSF [5] in later versions due to the difficulty of integrating ns-2 with their software). Some of these integration issues may be aleviated with ns-3.

A promising but challenging future direction is to improve the flow-level model in SimGrid so that its inaccuracies are reduced. For instance, one could modify the model, perhaps in an ad-hoc manner, to account for the TCP slow-start mechanisms. More importantly, one could replace the RTT-aware Max-Min model by another model. Indeed, it is known that when routers use the Drop Tail or RED scheme, and for TCP Reno, the bandwidth allocation is computed by maximizing the sum of the arctangents of the flow throughputs [17]. Furthermore, more recent versions of TCP, like TCP Vegas, produce allocations computed by maximizing the sum of the logarithms of the flow throughputs, which corresponds to the proportional fairness model proposed by Kelly [11]. Bandwidth sharing in real TCP networks is thus more complex than the simple RTT-aware Max-Min model implemented in SimGrid. Enabling these bandwidth sharing schemes is challenging (both in terms of implementation complexity and computational complexity) and must be based either on semi-definite programming or on Lagrangian optimization and gradient methods [37]. Such developments are planned for future versions of SimGrid. Note that the validation of such models will be straightforward thanks to the integration of SimGrid and GTNetS presented in this paper.

6. REFERENCES

[1] D. Bertsekas and R. Gallager. *Data Networks.* Prentice-Hall, 1992.

[2] D. G. Cameron, A. P. Millar, C. Nicholson, R. Carvajal-Schiaffino, F. Zini, and K. Stockinger. Optorsim: a Simulation Tool for Scheduling and Replica Optimisation in Data Grids. In *Proceedings of the Computing in High Energy and Nuclear Physics (CHEP) conference*, September 2004.

[3] H. Casanova and L. Marchal. A Network Model for Simulation of Grid Application. Technical Report RR-2002-40, Laboratoire de l'Informatique du Parallélisme, École Normale Supérieure de Lyon, France, October 2002.

[4] D. M. Chiu. Some Observations on Fairness of Bandwidth Sharing. In *Proceedings of the 10th IEEE Symposium on Computers and Communications (ISCC)*, pages 125–131, 2000.

[5] J. Cowie, H. Liu, J. Liu, D. Nicol, and A. Ogielski. Towards Realistic Million-Node Internet Simulations. In *International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA)*, June 1999.

[6] J. H. Cowie, D. M. Nicol, and A. T. Ogielski. Modeling the Global Internet. *Computing in Science and Engineering*, 1(1):42–50, 1999.

[7] C. Dumitrescu and I. Foster. GangSim: A Simulator for Grid Scheduling Studies. In *Proceedings of the Fifth IEEE International Symposium on Cluster Computing and the Grid (CCGrid'05)*, 2005.

[8] S. Floyd and K. Fall. Promoting the use of end-to-end congestion control in the Internet. *IEEE/ACM Transactions on Networking*, 7(4):458–472, 1999.

[9] K. Fujiwara. Cost and Accuracy of Packet-Level vs. Analytical Network Simulations: An Empirical Study. Master's thesis, University of Hawai'i at Manoa, May 2007.

[10] Q. He, M. Ammar, G. Riley, H. Raj, and R. Fujimoto. Mapping Peer Behavior to Packet-level Details: A Framework for Packet-level Simulation of Peer-to-Peer Systems. In *Proceedings of the 11th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, October 2003.

[11] F. Kelly. Charging and rate control for elastic traffic. *European Transactions on Telecommunications*, 8:33–37, 1997.

[12] H. Kim. Integrating network-calculus-based simulation with packet-level simulation for TCP-operated networks. *The International Journal of Computer and Telecommunications Networking*, 50(12):1995–2012, 2006.

[13] A. Legrand, L. Marchal, and H. Casanova. Scheduling Distributed Applications: The SimGrid Simulation Framework. In *Proceedings of the Third IEEE International Symposium on Cluster Computing and the Grid (CCGrid'03)*, May 2003.

[14] Y. Li, D. Leith, and R. Shorten. Experimental Evaluation of TCP Protocols for High-Speed Networks. Technical report, Hamilton Institute, 2005. To appear in IEEE Transactions on Networking.

[15] B. Liu, D. R. Figueiredo, Y. Guo, J. F. Kurose, and D. F. Towsley. A Study of Networks Simulation Efficiency: Fluid Simulation vs. Packet-level

Simulation. In *Proceedings of IEEE INFOCOM*, pages 1244–1253, 2001.

[16] X. Liu and A. Chien. Traffic-based Load Balance for Scalable Network Emulation. In *Proceedings of the ACM Conference on High Performance Computing and Networking, SC2003*, November 2003.

[17] S. H. Low. A Duality Model of TCP and Queue Management Algorithms. *IEEE/ACM Transactions on Networking*, 11(4):525–536, August 2003.

[18] L. Massoulié and J. Roberts. Bandwidth sharing: objectives and algorithms. *IEEE/ACM Transactions on Networking*, 10(3):320–328, 2002.

[19] M. Mathis, J. Semke, and J. Mahdavi. The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm. *Computer Communications Review*, 27(3), 1997.

[20] A. Medina, A. Lakhina, I. Matta, and J. Byers. BRITE: An Approach to Universal Topology Generation. In *Proceedings of the International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems (MASCOTS)*, August 2001.

[21] D. Nicol, M. Goldsby, and M. Johnson. Fluid-based simulation of communication networks using SSF. In *Proceedings of the European Simulation Symposium*, October 1999.

[22] The Network Simulator – ns-2. `http://www.isi.edu/nsnam/ns/`.

[23] T. N'Takpé and F. Suter. Critical Path and Area Based Scheduling of Parallel Task Graphs on Heterogeneous Platforms. In *Proceedings of the Twelfth International Conference on Parallel and Distributed Systems (ICPADS)*, July 2006.

[24] T. Ott, J. Kemperman, and M. Mathis. Window Size Behavior in TCP/IP with Constant Loss Probability. In *Proceedings of 4th IEEE Workshop on High-Performance Communication Systems*, June 1997.

[25] U. P., X. DÃl'fag, and A. Schiper. Neko: A Single Environment to Simulate and Prototype Distributed Algorithms. *Journal of Information Science and Engineering,*, 18(6):981–997, 2002.

[26] J. Padhye, V. Firoiu, D. Towsley, and J. Krusoe. Modeling TCP Throughput: A Simple Model and its Empirical Validation. In *Proceedings of the ACM SIGCOMM Conference*, pages 303–314, 1998.

[27] S. Phatanapherom, V. Kachitvichyanukul, and P. Uthayopas. Fast Simulation Model for Grid Scheduling Using Hypersim. In *Proceedings of the 2003 Winter Simulation Conference*, December 2003.

[28] B. Quetier and F. Cappello. A survey of Grid research tools: simulators, emulators and real life platforms. In *Proceedings of the 17th IMACS World Congress (IMACS)*, July 2005.

[29] G. F. Riley. The Georgia Tech Network Simulator. In *Proceedings of the ACM SIGCOMM workshop on Models, Methods and Tools for Reproducible Network Research*, pages 5–12, 2003.

[30] M. Ripeanu and I. Foster. A Decentralized, Adaptive Replica Location Service. In *Processing of the 11th IEEE International Symposium on High Performance Distributed Computing (HPDC-11)*, July 2002.

[31] H.-P. Schwefel, M. Jobmann, and D. Höllisch, D. Heyman. On the accuracy of TCP performance models. In *Proceedings of ITCom*, 2001.

[32] The SimGrid project. `http://simgrid.gforge.inria.fr/`.

[33] M. Snir and W. Gropp. *MPI: The Complete Reference*. The MIT Press, second edition edition, 1998.

[34] H. J. Song, X. Liu, D. Jakobsen, R. Bhagwan, X. Zhang, K. Taura, and A. Chien. The MicroGrid: a Scientific Tool for Modeling Computational Grids. *Scientific Programming*, 8(3):127–141, 2000.

[35] A. Sulistio, G. Poduvaly, R. Buyya, and C.-K. Tham. On Incorporating Differentiated Levels of Network Service into GridSim. *Future Generation Computer Systems (FGCS)*, 23(4):606–615, 2007.

[36] A. Takefusa, S. Matsuoka, H. Nakada, K. Aida, and U. Nagashima. Overview of a Performance Evaluation System for Global Computing Scheduling Algorithms. In *In Proceedings of the 8th IEEE International Symposium on High Performance Distributed Computing (HPDC-8)*, pages 97–104, August 1999.

[37] C. Touati, E. Altman, and J. Galtier. On Fairness in Bandwidth Allocation. Technical Report RR-4269, INRIA, France, September 2001.

[38] A. Vahdat, K. Yocum, K. Walsh, P. Mahadevan, J. Kostic, D.and Chase, and D. Becker. Scalability and Accuracy in a Large-Scale Network Emulator. In *Proceedings of the 5th ACM/USENIX Symposium on Operating System Design and Implementation (OSDI)*, December 2002.