

Simulating Large Networks – How Big is Big Enough? *

George F. Riley
Mostafa H. Ammar

College of Computing
Georgia Institute of Technology
Atlanta, GA 30332
{[riley,ammar](mailto:riley,ammar@cc.gatech.edu)}@cc.gatech.edu
(404)894-6705, Fax: (404)385-0332

Abstract

Simulation has become the evaluation method of choice for many areas of computer networking research. However, most existing network simulation packages have severe limitations on the size and complexity of the network being modeled. Simulated networks of just a few thousand network elements and a few thousand data flows will quickly exhaust the computing resources in any reasonably sized computer workstation. Thus the researcher is faced with the dilemma of proving concepts designed to work efficiently on networks of tens of millions of elements, using a simulation of only a few thousand elements. The grand challenge we discuss in this paper is that of using simulation to reach credible conclusions about Internet-scale network performance. We present data that demonstrates that simulation of Internet-scale networks is not presently feasible, nor is it likely to be feasible in the near future. We present a summary of current research in the field of large scale network simulations. These recent advances, while not enabling Internet-scale simulations, do offer the tools with which one can begin to tackle the problem. We sketch one possible approach and describe the issues that need to be resolved in order to realize it.

*This work is supported in part by NSF under contract number ANI-9977544 and DARPA under contract number N66002-00-1-8934.

1 Introduction

The modeling and performance analysis of computer networks is particularly well suited for *discrete event simulation* techniques. Simulation is widely used within the networking research community to investigate the behavior of many aspects of network performance. Among the many uses of simulation in networking are:

1. To gain insight into the behavior of particular protocols and mechanisms under a variety of network conditions. For this type of research, small to medium scale simulations can be sufficient to gain considerable understanding of the behavior of such protocols.
2. To understand the true effect of a new protocol, mechanism, network service, or application when widely deployed on a large network such as the Internet. Interactions between the newly proposed methodologies and the large amount and variety of competing traffic on the Internet are important considerations in gaining such understanding. As an example, the research issue might be to understand the effect of enabling multicast in a large *ISP*. For this type of research, much larger simulations with a variety of competing traffic is needed to gain credible evidence as to the effect of the proposed change. This paper is concerned with these type of questions that require larger scale simulations.

All too often, the difference between the above two research objectives are ignored. Researchers often extend results from small simulations (as in item1) and make inferences regarding effect and behavior when extended to a large scale without performing

the large scale simulations. This dilemma is primarily due to the fact that it is simply impossible to create a model of the Internet. This is primarily due to four reasons.

1. We do not have tools capable of modeling networks of that size. All existing tools have bounded capacity, from a few hundred nodes for OpNet[1], to a few thousand nodes for *GloMoSim*[2] and *ns*[3], to a few tens of thousands of nodes for *TeD*[4], up to a few hundred thousand nodes for *pdns*[5], *SSF*[6] and *USSF*[7].
2. Even if we had the capability of constructing large enough models, we would not know how to construct models that accurately reflect the topology and performance of the Internet[8]. The topology of the Internet is changing daily, and the offered load can vary wildly even over short time periods.
3. Even if we could construct accurate models of the Internet, any simulation with packet level detail could not produce results in any reasonable amount of time. A simulation of a large number of high speed network links could easily run for days or weeks before producing any meaningful results.
4. Even if the simulation did finish, the amount of raw data collected by the simulation would be too large to be manageable.

What is needed therefore is some methodology whereby a researcher can construct a reasonably sized simulation (that runs in a reasonably short amount of time), that gives results that can then be extrapolated to predict performance of the Internet. A discussion of one possible methodology, and the difficult problems obstructing the implementation of such a methodology, is discussed later in this paper.

The remainder of this paper is organized as follows. Section 2 discusses some of the challenges encountered when simulating large, Internet-scale networks. Section 3 gives an overview of the state of the art in large scale network simulations. Section 4 discusses one possible approach that might be used to extend conclusions from small simulations to larger ones. Finally, section 5 gives some conclusions from this work.

2 Challenges in Simulating the Internet

We first consider the question of whether it is feasible (today or at some point in the future) to build a network simulation of the size of the Internet. We argue that this is infeasible. While it is straightforward to make the argument, this points to a fundamental limitation in the use of simulation as a tool for evaluation of computer networks: *By necessity, validation or evaluation using network simulation has to be able to draw its conclusions from a scaled down version of the real system.*

Sally Floyd and Vern Paxson argue in [8] that simulating the Internet is not possible. The arguments are, in essence, that nobody really knows the topology, mix of flows, or routing agreements in the Internet. In this section, we optimistically ignore all of Floyd's arguments and assume that we somehow know the exact topology, flow mix, and routing of the Internet. We make some *extremely conservative* estimates regarding the number of hosts, routers, links, and traffic loads for the Internet, and show that a simulation model of a network of the size of the Internet is intractably large. For this discussion, our assumptions are:

1. There are 110,000,000 end hosts on the Internet[9].
2. There is one router for every 100 hosts.
3. Each router has links to 4 other routers.
4. 50% of all hosts connect to their first-hop router using a 56kb modem link. The other 50% connect using a 10Mbps Ethernet link.
5. Of the router-to-router links, 50% are 10Mbps, 40% are 100Mbps, 5% are OC12 (655Mbps), and 5% are OC48 (2.4Gbps).
6. The host-to-router links are 50% busy.
7. The router-to-router links are 10% busy.
8. At any point in time, 1% of end hosts have a single data connection to one other end host.
9. The average packet size on the links is 5,000 bits.
10. The growth rate of the host count is 25% per year.
11. The traffic on the links doubles every 6 months.

We emphasize again that these estimates are extremely conservative, in that the actual count of

links, routers, and traffic are likely substantially larger than given here. Using these estimates, we can then calculate the number of simulation events per second that we would need to process, the amount of memory needed to maintain the models, and the amount of disk space needed to log the simulation results.

1. The total events that must be modeled is 2.9×10^{11} events per second. Assuming a simulator can process one million events per second (possibly attainable on a 1Ghz CPU), the execution time for such a simulation would take 290,000 seconds for each second of simulation time (almost four days).
2. The total memory required for the model is 2.9×10^{14} bytes (nearly three hundred terabytes). This calculation uses measured memory requirements for nodes, links, connections and packets using the *ns* network simulator [3]. We did *not* include memory needed for simulated routing tables at each node. See [10] for a discussion of this memory utilization and a potential solution.
3. The total disk space required for logging the simulation results is 1.4×10^{13} bytes for each second of simulation time. This calculation assumes four logged events per packet per link (enqueue, dequeue, transmit, receive), and 50 bytes per event (approximately the size of logged information in *ns*).

To put these numbers in perspective, assuming we want to model 100 seconds of activity on the Internet, we would need more than a year of CPU time, nearly three hundred terabytes of main memory, and about 1.4 petabytes of disk storage to log the results.

One approach might be to run this simulation in parallel on a network of computers working cooperatively to simulate the entire network (such as the *pdns* simulator by Riley et al.[5, 11]). Even if we could assemble 1000 systems with a high speed interconnect, each system would still need nearly 300 Gigabytes of main memory and 1.4 Terabytes of disk storage. CPU speedups for the 1000 system simulation would be nowhere near the thousand-fold increase, since overhead for time synchronization grows as the number of systems. The authors doubt if even a hundred-fold speedup could be attained in this (hypothetical) distributed computing environment.

There is a well known anecdote in distributed computing that gives two approaches one can take when a given application does not run fast enough on

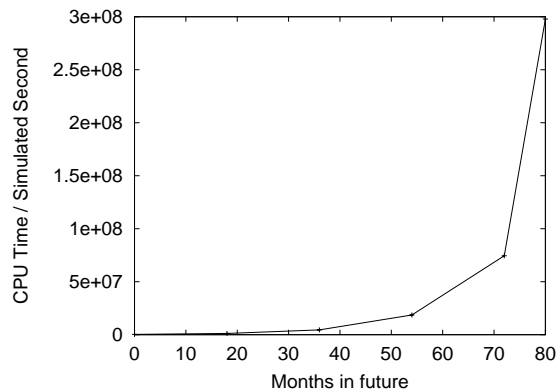


Figure 1. Growth in Simulation Time

existing hardware platforms. One can spend 18 months modifying the application to run in a distributed environment and gain a two-fold performance increase by running on two systems simultaneously. Alternately, one can do absolutely nothing for 18 months and buy a faster system (twice as fast according to Moore’s law) and gain a two-fold increase in performance. This anecdote suggests that perhaps if we simply wait long enough, we will be able to simulate the Internet by purchasing bigger and faster hardware. Unfortunately, such an approach will not work in our quest to simulate the Internet.

We used Moore’s law calculate the increase in the achievable events per second for simulation hardware. We then calculated the increase in the total number of events per second to be modeled, using the growth rates assumed above in items 10 and 11. Finally we calculated how many CPU seconds would be required to simulate a single second of Internet traffic, projected 80 months in the future (June 2008). The plot in figure 1 shows that, even with increased capabilities due to Moore’s law, the simulation time still increases exponentially. We see that by Summer of 2008, we would require nearly 300 million CPU seconds to simulate a single second of the Internet load.

3 State of the Art in Large Scale Network Simulation

In this section, we give a brief overview existing work in the area of large scale network simulation.

- Parallel / Distributed *ns* (*pdns*) was described by Riley et al. in [5] and [11]. The *pdns* simulator consists of extensions to the widely used

and publicly available *ns* network simulator. The extensions allow many existing *ns* simulations to be run in a distributed environment with minimal changes. The *pdns* implementation also takes advantage of the large body of existing network models found in *ns*, and uses those without modification. Also included in the *pdns* extensions is a novel packet routing method called NIX-Vectors that allows routing decisions without the necessity of routing tables, resulting in substantial memory savings[10]. The *pdns* simulator has been demonstrated on network models of hundreds of thousands of nodes.

- The *Scalable Simulation Framework (SSF)* is introduced by Ogielski et al. in [6] and [12]. SSF is designed to model large-scale networks, which are described in a generic modeling language called *Domain Modeling Language (DML)*. SSF can be run in a distributed environment (called *DaSSF*) on a tightly coupled shared-memory symmetric multiprocessor. *DaSSF* achieves good parallel performance by using a periodic time-stepped approach. All processors periodically enter a rendezvous, exchange messages, and process messages. The time period of the synchronization points is such that processors can safely process messages between synchronization cycles without fear of erroneous results due to unsafe events. The *SSF* simulator has support for both Java and C++. *SSF* has been demonstrated on networks of several hundred thousand nodes.
- The *Telecommunications Description Language (TED)* is described by Perumalla and Fujimoto in [13], [4] and [14]. *TED* is language for describing telecommunications networks, coupled with an optimistic network simulation engine, based on *Georgia Tech Time-Warp* [15]. It has demonstrated good performance and scalability when modeling *ATM* cell switches and *PNNI*. *TED* has been demonstrated on network models consisting of tens of thousands of nodes.
- The *Global Mobility Simulator (GloMoSim)* is introduced by Xeng and Bagrodia in [2]. *GloMoSim* uses the *Parallel Simulation Environment for Complex Systems (Parsec)* [16] simulation engine to provide parallel simulation of wireless mobile networks. The Parsec simulation engine is designed to run on a tightly coupled shared-memory symmetric multiprocessor. The Parsec simulator uses extensions to the C programming language to describe simulation entities (such as nodes and links) and the exchange of messages between those

entities. *GloMoSim* uses text based configuration files to describe the network elements in the simulation, the mobility of the elements, and the flow of data between those network elements. *GloMosim* has been demonstrated on networks of thousands of nodes.

- The *TeleSim* framework for telecommunication simulation is described by Unger and Lomow in [17]. The *TeleSim* framework uses the *Jade* optimistic simulation environment [18]. Network models for *TeleSim* include the Signaling System 7 (SS7) and the Trunk Network Model (TNM). Using the *TeleSim* simulator on a SMP system with 36 processors has shown speedup of nearly 14 as compared to the same simulation on an single processor. The *TeleSim* framework has been demonstrated on network models of hundreds of nodes.
- The *Ultra-Large Scale Simulation Framework (USSF)* is described by Rao and Wilsey [7]. The *USSF* simulator is based on the *WARPED* simulation engine [19]. *WARPED* is an optimistic simulator based on Time Warp [20]. The topology models for *USSF* are described in a *Topology Specification Language* described in [21]. *USSF* has demonstrated the potential to model large networks using a network of Dual-CPU Pentium processors connected by an Ethernet network. *USSF* has been demonstrated on network models of hundreds of thousands of nodes.
- A selective abstraction approach to modeling networks using the *ns* simulator is discussed by Huang, Estrin and Heidemann in [22]. In this method, decreased detail in the modeling of end-to-end data flows is traded for increased computational efficiency. Using this approach, the *ns* simulator has been demonstrated on multicast groups consisting of thousands of end hosts.

The preceding summary indicates that some progress is being made in developing tools that allow us to simulate large networks. *SSF*, *pdns*, and *USSF* all allow a detailed simulation of hundreds of thousands of nodes. Even though the simulation of Internet-scale networks is, and will continue to be, beyond our capabilities, these recent advances do allow some flexibility and growth in the size of networks that can be reasonably simulated.

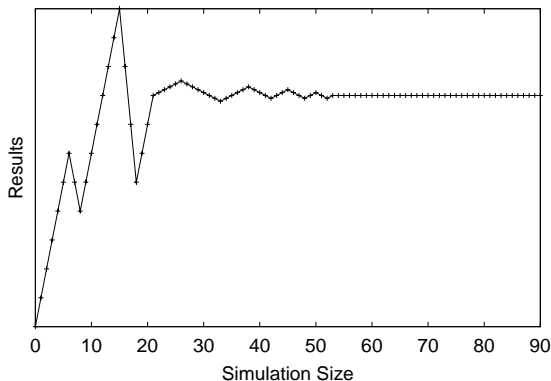


Figure 2. Size / Results Continuum

4 A Strawman Approach to Scaling Results

It should be clear from the previous sections that a simulation of the Internet is not presently possible, nor is it likely to be possible in the near future. What is needed therefore is some way to construct smaller, computationally tractable, simulations that give results that can easily be extrapolated to Internet size networks.

In this section we consider the question of how to determine the appropriate size of a simulation. One possibility is to use a *direct* approach to answer this question. This would take into account the various details about the simulation objectives and models and apply algorithms to produce the appropriate size parameters that one should use. Given the complexity of simulation models and parameters, we do not believe that this direct approach holds much promise. Instead, we focus here on *search-based* approaches in which one would run simulations of different sizes to find the appropriate scale. Such techniques will be more heuristic in nature.

One such approach, which we dub the *Iterative Approach*, assumes there is a continuum of *simulation sizes*, with a corresponding continuum of results from running these various size simulations. Figure 2 gives a hypothetical graph demonstrating these continua. This graph is for illustrative purposes only, since it indicates a single dimension for both “size” and “results”. As will be discussed, there is no single scalar measurement for either of these axes, but the graph is adequate for this discussion.

First, the simulationist should construct a fairly small simulation (say for example at size 10 in the graph), to debug the simulation methods and

observe results on a small and manageable scale. Once the simulationist is confident in the networking models and has collected some simulation results on a small scale, a larger network model would be created, (for example size 20 in our graph). In our example, the larger simulation gives “different” results indicating we should repeat the iteration. A simulation at size 40 again gives different results, and finally the simulation at size 80 matches approximately what was obtained from size 40. At this point no further iterations are needed.

Clearly, this is a contrived example; there is no single scalar measure of “results”, nor are the performance metrics likely to converge as neatly and conveniently as we show in our graph. However, further research on the effect of simulation size scaling on measured results could lead to insights into when such an approach might be useful.

The iterative approach requires that we increase the size of our simulations at each iteration and compare results to prior iterations. But how does the simulationist do this size increase in such a way that the results are comparable? For example, what does it mean to “double” the size of a simulation? Does it mean twice as many of everything (end hosts, routers, links, flows)? Clearly, simply doubling the number of network elements and flows will affect in some way the offered load on each of the links in the topology.

To use a concrete example, we have used simulation methods to compare the performance of the widely used *Drop Tail* queuing discipline to the more sophisticated *Random Early Detection (RED)*[23] discipline. To begin with, we used a small scale topology consisting of a single bottleneck link and 94 end system nodes, divided into 47 web servers and 47 web clients. The link delays on the simulated links were fixed at a number of different values, giving a wide variation in round-trip-time delays between pairs of clients and servers. Each of the web client nodes models a number of web browsers simultaneously, resulting in several thousand web browsers sharing the bottleneck link. The web browsing models are as described by Mah in [24].

This topology matches closely that used in a similar experiment described in [25]. The purpose of the experiment is to measure the average end-to-end delay experienced by web browser clients, to compare the performance of the two queuing disciplines.

A representative set of results from our small scale experiments are shown in figures 3 and 4. These match roughly those reported in [25], and show that, in this particular set of input parameters, the

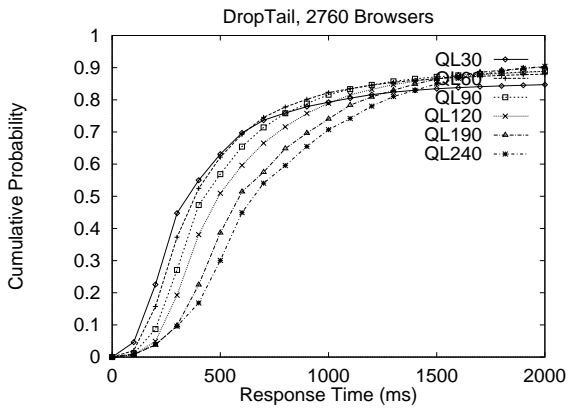


Figure 3. Cumulative Response, DropTail Queuing

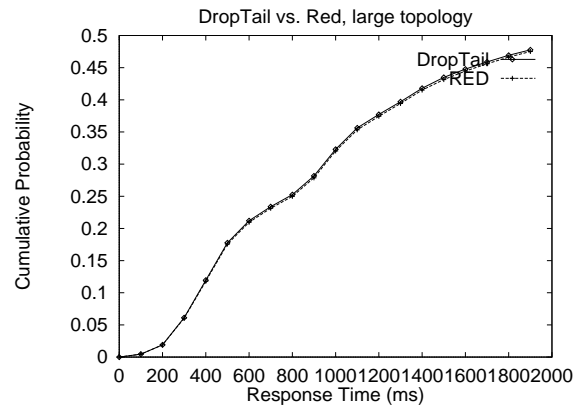


Figure 5. Cumulative Response, DropTail vs. Red, Large Topology

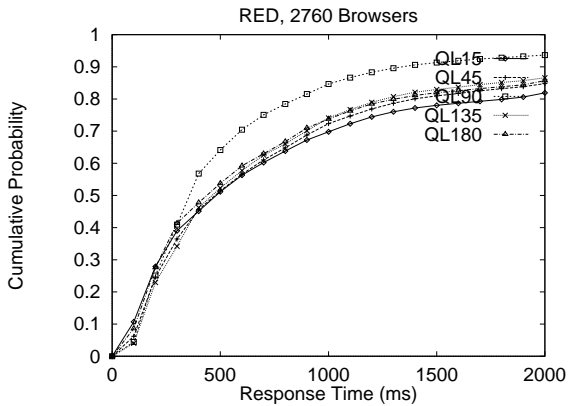


Figure 4. Cumulative Response, RED Queuing

simple *Drop Tail* method outperforms slightly the more complex *RED* method. Simulation (and laboratory) experiments on such a small scale are easy to construct, easy to control, and easy to interpret. However, these small scale experiments fail to account for a number of conditions encountered by web browsers in the actual Internet, including but not limited to:

1. More than one bottleneck link on the path.
2. Presence of *cross-traffic* (other *TCP* flows that share some portion of the traffic path).
3. Extreme variation of queue lengths at routers on the path (combinations of lightly loaded links and more heavily loaded links).

We then set out to increase the scale of this experiment, to try to overcome the shortcomings listed above. We created a much larger topology, and used *pdns* to distribute the simulation on a set of

sixteen high performance workstations. The model consisted of eight leaf subnetworks and eight transit (*ISP*) subnetworks, of varying sizes. A typical leaf subnetwork (we experimented with a number of different configurations) contained about 5000 total nodes arranged in a tree-like topology, about 4000 web clients, and about 400 web servers. The transit subnetworks typically contained fewer nodes connected with higher speed links, in an attempt to model a typical *ISP* network.

Obtaining meaningful and comparable results in this environment turned out to be more difficult than we first believed (and in fact is ongoing research). In the smaller scale simulations, the differences in behavior between *RED* and *Drop Tail* were clear within a fairly small range of congestion levels on the bottleneck links. With the larger topologies, the congestion level varied substantially over short time intervals, due mostly to a two-order-of-magnitude increase in the number of web browsers starting and completing requests. Thus we often ended up with results similar to figure 5, showing absolutely no difference in the relative performance. Other times we saw extreme variations in performance (not shown), sometimes favoring *RED* and sometimes favoring *Drop Tail*. Our belief is that these varying results are due to our inability to create a realistic workload that can maintain a specific desired level of congestion on a specific set of links. One possible solution to this problem is to develop some way to insure that certain *invariants* remain constant as the scale of the simulations increases. For example for simulations evaluating *RED* deployment and parametrization, the invariant desired may be that the offered load on each bottleneck link is the same in the small and large simulations (as seems to be indicated by our experience).

Clearly the above scenario points out the necessity for comparing results of differing size simulations under somewhat similar conditions. What is needed is some way to insure that certain *invariants* remain constant as the scale of the simulations increases.

Consider that the invariant desired for a given simulation is that the offered load on each bottleneck link is the same in the small and large simulations. One way to do this might be to use *dynamic load adjustment*.

The method for dynamic load adjustment is as follows:

1. While simulating in iteration $k - 1$, detailed measurements are kept to track the *offered load* per unit time on each link in the simulated topology. This is done by measuring the offered bytes from each *TCP* source in the simulation, and assigning this load to each link in the path from the source to the destination.
2. While simulating iteration k , the offered load on the links is again measured and compared to the measurements in iteration $k - 1$.
3. If links are found to have substantially smaller offered load in iteration k , more *TCP* flows are assigned to end systems that use the underutilized link.
4. If links are found to have substantially larger offered load in iteration k , fewer *TCP* flows are assigned to end systems that use the overutilized link.
5. When the offered loads on all links are approximately the same, the performance measurements are taken, and a valid comparison can be done.

Clearly, there is some potential for interaction between the load increases in step 3 and the load decreases in step 4. Since the load on each link is dependent in some part on the load on other links, possibilities exist for instabilities and oscillations in offered load adjustments. Further research is needed to find a good method for the dynamic load adjustment presented here.

More generally, further research is needed to understand how to scale network simulations in a reasonable fashion, and how concepts such as “network invariants” may be needed for this scaling.

5 Conclusions

Modeling and simulation of the Internet as a whole is indeed a grand challenge, and beyond the capabilities of any existing simulation method or hardware platform. It is clearly hard to solve as discussed in section 2. Recent advances in building larger simulation models as discussed in section 3 have made it possible to simulate larger and larger networks. However, this only begins to provide us with the tools with which we can tackle the problem of simulating Internet-scale networks. Our strawman approach in section 4 points to one possible direction for such simulations. There are undoubtedly other approaches.

A simulation based approach providing a better understanding of the behavior of the Internet, and of any proposed additions or modifications, will clearly have far-reaching impact and benefits.

References

- [1] S. Bertolotti and L. Dunand, “Opnet 2.4: an environment for communication network modeling and simulation,” in *Proceedings of the European Simulation Symposium*, October 1993.
- [2] X. Zeng, R. Bagrodia, and M. Gerla, “GloMoSim: a library for parallel simulation of large-scale wireless networks,” in *Proceedings of the 12th Workshop on Parallel and Distributed Simulations*, May 1998.
- [3] S. McCanne and S. Floyd, “The LBNL network simulator.” Software on-line: <http://www.isi.edu/nsnam>, 1997. Lawrence Berkeley Laboratory.
- [4] K. Perumalla, R. Fujimoto, and A. Ogielski, “Ted - a language for modeling telecommunications networks,” *ACM SIGMETRICS Performance Evaluation Review*, vol. 25, March 1998.
- [5] G. F. Riley, R. M. Fujimoto, and M. H. Ammar, “Parallel/Distributed ns.” Software on-line: www.cc.gatech.edu/computing/compass/pdns/index.html, 2000. Georgia Institute of Technology.
- [6] J. H. Cowie, D. M. Nicol, and A. T. Ogielski, “Modeling the global internet,” *Computing in Science and Engineering*, January 1999.
- [7] D. M. Rao and P. A. Wilsey, “Simulation of ultra-large communication networks,” in *Proceedings of Seventh International Symposium on Modeling, Analysis and Simulation of*

- of Computer and Telecommunication Systems*, October 1999.
- [8] V. Paxson and S. Floyd, “Why we don’t know how to simulate the Internet,” in *Proceedings of the 1997 Winter Simulation Conference*, pp. 1037–1044, 1997.
- [9] Internet Software Consortium, “Internet domain survey.” <http://www.isc.org/ds/WWW-200101/index.html>, Jan 2001.
- [10] G. F. Riley, M. H. Ammar, and R. M. Fujimoto, “Stateless routing in network simulations,” in *Proceedings of the Eighth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, August 2000.
- [11] G. F. Riley, R. M. Fujimoto, and M. H. Ammar, “A generic framework for parallelization of network simulations,” in *Proceedings of Seventh International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS’99)*, October 1999.
- [12] 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*, June 1999.
- [13] K. S. Perumalla and R. M. Fujimoto, “Efficient large-scale process-oriented parallel simulations,” in *Proceedings of the Winter Simulation Conference*, December 1998.
- [14] K. Perumalla, M. Andrews, and S. Bhatt, “Ted models for ATM internetworks,” *ACM SIGMETRICS Performance Evaluation Review*, vol. 25, March 1998.
- [15] R. M. Fujimoto, “Time warp on a shared memory multiprocessor,” *Transactions of the Society for Computer Simulation*, vol. 6, pp. 211–239, July 1989.
- [16] R. Bagrodia, R. Meyer, M. Takai, Y. Chen, X. Zeng, J. Martin, B. Park, and H. Song, “Parsec: A parallel simulation environment for complex systems,” *IEEE Computer*, vol. 31, pp. 77–85, October 1998.
- [17] B. W. Unger and G. A. Lomow, “The Telecom framework: A simulation environment for telecommunications,” in *Proceedings of the 1993 Winter Simulation Conference*, December 1993.
- [18] D. Baezner, G. Lomow, and B. Unger, “A parallel simulation environment based on time-warp,” *International Journal in Computer Simulation*, 1993.
- [19] R. Radhakrishnan, D. E. Martin, M. Chetlur, D. M. Rao, and P. A. Wilsey, “An object-oriented time warp simulation kernel,” in *Proceedings of International Symposium on Computing in Object-Oriented Paralle Environments*, December 1998.
- [20] D. R. Jefferson, “Virtual time,” in *ACM Transactions on Programming Languages and Systems*, vol. 7, pp. 404–425, July 1985.
- [21] D. M. Rao, R. Radhakrishnan, and P. A. Wilsey, “FWNS : A framework for web-based network simulation,” in *International Conference on Web-based modelling and simulation*, Jan 1999.
- [22] P. Huang, D. Estrin, and J. Heideman, “Enabling large-scale simulations: selective abstraction approach to the study of multicast protocols,” in *Proceedings of the International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, July 1998.
- [23] S. Floyd and V. Jacobson, “Random early detection gateways for congestion avoidance,” *IEEE Transactions on Networking*, vol. 1, pp. 397–413, August 1993.
- [24] B. A. Mah, “An empirical model of http network traffic,” in *Proceedings of IEEE INFOCOMM*, pp. 592–600, 1997.
- [25] M. Christiansen, K. Jeffay, D. Ott, and F. D. Smith, “Tuning RED for web traffic,” in *Proceedings of ACM SIGCOMM 2000*, pp. 139–150, Aug 2000.