Realistic Large-Scale Online Network Simulation

Xin Liu and Andrew A. Chien
Department of Computer Science and Engineering
and Center for Networked Systems
University of California, San Diego
{xinliu, achien}@cs.ucsd.edu

Abstract

Large-scale network simulation is an important technique for studying the dynamic behavior of networks, network protocols, and emerging classes of distributed application (e.g. Grid, peer-to-peer, etc.) Large-scale and realism are two critical requirements for network simulations of Grid application studies. Our work here extends previous efforts in three key ways. First, we study networks 100x larger than in our previous studies (20,000 routers). Second, at this scale, we study realistic network structures (100 AS's, BGP4 and OSPF routing) versus flat OSPF routing. Finally, we describe and evaluate a new profile-based load-balancing approach called hierarchical profile-based load balance.

Our extensive large-scale experiments with profile-based load balance (PROF) on flat-routed (OSPF) networks show that PROF outperforms several other techniques based on topology and static application information. However, these results and those for multi-AS networks motivate our invention of a new hierarchical technique (HPROF) which clusters network nodes to achieve a desired minimum link latency (MLL), a key determinant of simulation parallelism, then applies the graph partitioner. HPROF explicitly controls the tradeoff between simulation efficiency and available parallelism, producing robust and superior performance for large-scale networks, including both single-AS and multi-AS networks. HPROF can improve load imbalance by 40%, and reduce the simulation time by about 50% in our 20,000 router simulations executed on 128-node clusters. The parallel efficiency achieved by these simulations is over 40%, providing substantial capabilities for simulating large networks. In summary, these advances demonstrate that realistic large-scale network simulation for networks of 20,000 routers (comparable to a large Tier-1 ISP network like AT&T) can be accomplished with our system.

1. Introduction

Historically, network simulations/emulations have been used extensively to explore the behavior of network protocols[1-3]. Because of the difficulty of modeling application behavior in detail, most of these simulations use simple application models to exercise the protocols and networks. However, with the advent of large numbers of applications which tightly couple the use of compute, storage, and network, techniques to study these resources together are emerging. In particular, large-scale network simulation is an important technique for studying the dynamic behavior of networks, network protocols, and emerging classes of distributed applications, including Peer-to-Peer [4] and Grid applications [5] – where the network is an important contributor to application performance, applications generate large amounts of network traffic, and overall application performance is critical. A wide variety of simulation systems have been built to model network behavior based on discrete event simulation[6-9].

0-7695-2153-3/04 $20.00 (c)2004 IEEE

The MaSSF, a network simulation tool [10] is a key component of the MicroGrid system[11] built by our group at UCSD to study the dynamic behavior of Grid applications. The MicroGrid enables the execution of complete Grid or distributed applications. There are two key requirements for a network simulator targeted for large-scale study of such applications and resource infrastructures.

The first requirement is that it must scale to Internet-scale network. As in many other network simulation projects, the MaSSF utilizes cluster systems to achieve scalable performance. By harnessing scalable compute resources, the MaSSF system and user applications together are themselves an interesting distributed application, and load balance of network simulation itself is one key problem for scalability. In our previous work[10], we formulated the load balance problem as a graph partitioning problem and applied classical graph partition algorithms [12-15] to solve it. Three approaches exploiting topology only, topology and application placement, and profile-based were presented and evaluated for moderate-sized networks. The results showed that exploiting static topology and application placement information improves load balance, but a profile-based approach further improves the load balance achieved. In this paper, we improve on all of these with a new hierarchical approach and evaluate all of them on much larger networks (100x).

The second requirement for large-scale network simulation is that it must simulate in detail the structure of realistic networks. Our previous published work on MaSSF [10] addresses simulation accuracy (validation) in this paper we will address the issue of realistic network topology and routing selection. While much research explores realistic Internet-like topology generators and background traffic, few efforts explore realistic network routing with most large-scale simulations pursuing only shortest-path routing (OSPF). It is well-known that in large, multi-AS networks, routing amongst different AS domains is controlled by BGP and policy routing, therefore connectivity does not equal reachability. A realistic

Internet network simulation must support BGP routing among Autonomous Systems (AS) and must have reasonable BGP routing policy configuration. In MaSSF, we support the detailed BGP4 routing protocol, and here address the remaining problem of how to generate a reasonable BGP routing policy for large networks.

In this paper, we demonstrate techniques that enable realistic large-scale online network simulation. These techniques together make a realistic large-scale simulation study of the networks and coupled application performance possible. The more specific contributions of this paper include:

- evaluating our previous load-balancing techniques (TOP, PROF) for online simulation using networks 100 times larger (20,000 routers),
- study of more realistic network routing structures (100 AS's, BGP4 and OSPF routing) versus flat OSPF routing,
- describing and evaluating a new load-balancing approach called hierarchical profile-based load balance (HPROF),
- developing a set of heuristics for automatic realistic BGP routing configuration as an improvement to Internet-like topology generation,
- evaluating a range of load balancing techniques (TOP, HTOP, PROF, HPROF) in simulations of both single-AS and multi-AS networks, which demonstrate HPROF can improve load imbalance by 40%, and reduce simulation time by 50% in 20,000-router simulations executed on 128-node clusters, and
- demonstrate that realistic large-scale network simulation for networks of 20,000 routers (comparable to a large Tier-1 ISP network like AT&T) can be accomplished with our MaSSF system

The remainder of the paper is organized as follows. Section 2 provides background on MicroGrid/MaSSF and Internet hierarchy. Section 3 describes the load balance approaches for scalability challenges, summarizing existing partition algorithms and presenting our hierarchical partition approach for larger scale networks. Experiments in Section 4 demonstrate the scalability of our partition approaches on single-AS networks. In Section 5, we first introduce a set of heuristic rules for automatic BGP routing configuration, then provide evaluation results for our load balance approaches against Internet-like multi-AS networks with realistic BGP routing. The results are discussed, along with related work in Section 6, and finally Section 7 summarizes our work and points out some future directions for research.

2. Background

2.1 MicroGrid and MaSSF

We have designed and implemented a tool called the MicroGrid [11, 16] which enables accurate and comprehensive study of the dynamic interaction of applications, middleware, resources, and networks. The MicroGrid creates a virtual grid environment by accurately modeling networks, resources, and information services to enable users, grid researchers, or grid operators to study arbitrary collections of resources and networks. In addition, the MicroGrid virtualizes transparently, allowing the direct study of complex applications or middleware whose internal dynamics are difficult to model accurately. That is, real application software and middleware can be used unchanged and executed on arbitrary virtual grid structures. In short, the MicroGrid provides a virtual grid infrastructure that enables scientific and systematic experimentation with dynamic resource management techniques and adaptive applications by supporting controllable, repeatable, and observable experiments. Because the rate of execution of all components of the system (applications, network, etc.) can be controlled, a wide range of relative performance and system combinations can be modeled using MicroGrid.

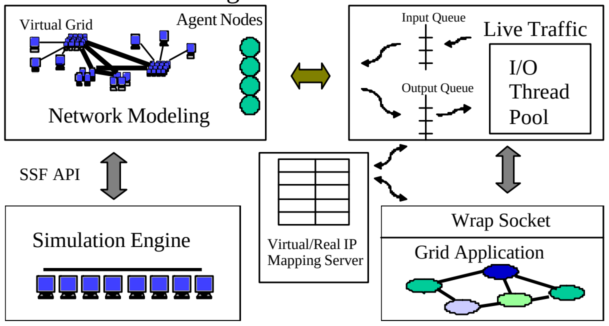

Figure 1. The MaSSF Scalable Network Simulation System

The key component of MicroGrid is the online network simulator MaSSF. MaSSF (pronounced "massive") is a scalable packet-level network simulator that supports direct execution of unmodified applications. MaSSF consists of four parts.

- **Simulation Engine**: MaSSF uses a distributed simulation engine based on DaSSF[17]. It utilizes MPI-connected cluster systems to achieve scalable performance. A soft real-time scheduler is used to emulate virtual computer resources, allocating CPU proportionately. This scheduler can also be used to run in a scaled-down (slowdown) mode when the simulated system is too large to run in real time on the available hardware. With the global coordination of the MicroGrid, this feature provides tremendous flexibility to simulate a wide range of networks and resources accurately.

- **Network Modeling**: MaSSF provides the necessary protocol modules for detailed network modeling, such as IP, TCP/UDP, OSPF, and BGP4. We have built basic implementations of these protocols which maintain their behavior characteristics. We also use a network configuration interfaces similar to a popular Java network simulator implementation, SSFNet[18], for user convenience.

- **Online Simulation Capability**: To support simulation of traffic from live applications, we employ an Agent which accepts and dispatches live traffic from application wrapper to the network simulation. Traffic is also sent back to the application through the Agent module.

- **Live Traffic Interception**: Application processes use a wrapper library called WrapSocket to intercept live network streams at the socket level. The WrapSocket then talks with the Agent module to redirect traffic into the network simulator and vice versa. WrapSocket can be either statically or dynamically linked to application processes and requires no application modification.

These components and their relationship are illustrated in Figure 1. For more details of MaSSF, the interested reader is referred to [11].

2.2 Background on Internet Hierarchy

We summarize some background information on Internet routing[19].

Autonomous System(AS): The Internet consists of more than 10,000 ASes, and the relationship between ASes is decided by commercial agreements. Two basic relationships are Provider-and-Customer and Peer-and-Peer. A pair of ASes that one offers Internet connectivity and delivers traffic to the other is said to have a provider-and-Customer relationship; a pair of ASes that provides connectivity and delivers traffic between their respective customers is said to have a Peer-and-Peer relationship.

Internet Topology Hierarchy: According to [20], ASes can be classified into 5 categories: Customers, Small Regional ISPs, Outer Cores, Transit Cores, and Dense Cores. The Customers count for about 90% of total ASes, and Dense Cores only count for 2%. The Dense Cores are roughly equal to Tier-1 ISP, and they have almost full connection between each other.

Policy-based Routing: BGP4 is widely used inter-AS routing, which exchanges reachability information between ASes in the form of route announcement. Each route announcement contains some attributes, such as AS Path, Multi-Exit-Discriminator (MED), and Next Hop. The most important attribute, AS Path, is a list of AS numbers to a network. Other attributes are used to define routing policies. The key feature of BGP protocol is policy routing, which allows each AS to choose its own policy in accepting routes, selecting the best route, and announcing routes to its neighbors.

3. Hierarchical Load Balance for Large-Scale Network Simulation

3.1 Scalability Challenge in Network Simulation

To achieve scalable performance, MaSSF uses a distributed simulation engine running on a cluster. Given a virtual network topology and a set of cluster nodes, MaSSF partitions the virtual network into multiple blocks, assigns the blocks to cluster nodes, and simulates in parallel, as shown in Figure 2. Every cluster node runs a discrete event simulation engine and exchanges events with other cluster nodes. To maintain accurate simulation, cluster nodes must synchronize periodically.

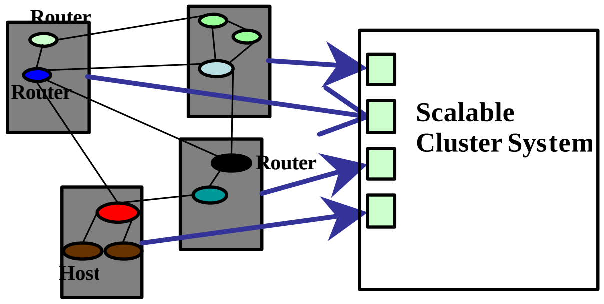

Figure 2. Mapping routers to physical resources

For large simulations, the network mapping is too complex to be done manually. But in such simulations, good load balance is critical to achieving good parallel efficiency. Load balance for network simulations is known to be difficult because the network traffic workload on each physical node varies greatly, depending both on the virtual mapping and network traffic in that subset of virtual network (Figure 3). However, beyond achieving good load balance, we need to consider two more optimization goals to achieve good network simulation performance. First, we want to maximize the Minimal Link Latency (MLL) across partitions. Maximizing MLL reduces the frequency of synchronization among simulation engines, increasing concurrency, a critical element of scalability for large scale simulations. This is an attribute of our MaSSF system and all other network simulators based on conservative discrete event simulation engines. Second, we want to minimize the communication of events between simulation engine nodes. Transferring a simulation event across physical nodes is expensive both in terms of computation overhead and communication latency. Also, the physical network of the simulation engine nodes is often a performance bottleneck for the whole simulation. Hence, it is important to minimize this communication.

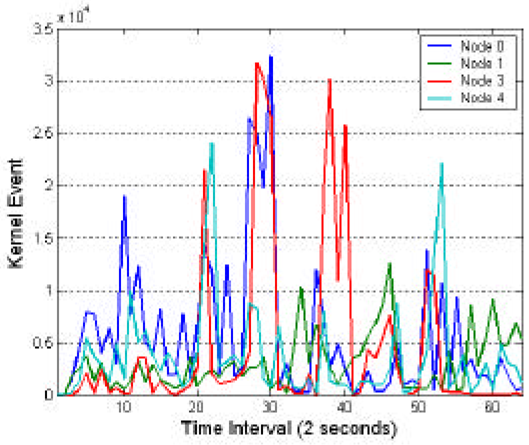

Figure 3. Load Variation over the Lifetime of Simulation

3.2 Modeling Load Balance as a Graph Partitioning Problem

A network mapping problem can be naturally modeled as a graph partitioning problem and solved with the classical graph partitioning algorithms.

Given a weighted graph G with constraint and optimization objectives, a typical graph partitioner can partition G and achieve balanced total weights and minimized edge-cut across partitions. The challenge here is how to apply the graph partitioning algorithm to solve the mapping problem by defining the suitable input graph G, constraint conditions, and optimization objectives for the graph partitioning algorithm. As shown in Figure 4, the mapping process first takes the network structure and traffic information as input, creates a graph G, and builds objectives and constraints. Then it applies the partitioning algorithm to get a partitioned network. The partitioned network defines the mapping of simulated network nodes to physical resources (subject to additional arbitrary choices of placement amongst symmetric physical resources).

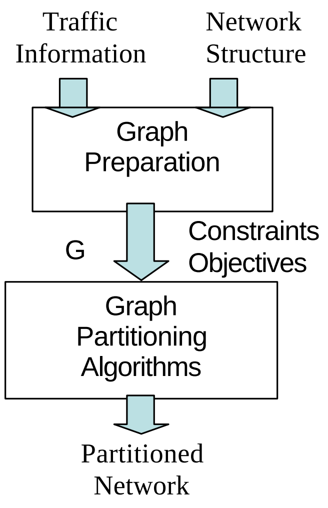

Figure 4. Process of Network Mapping

3.3 Existing Load Balance Approaches

In our previous work, we explored approaches based on a range of static and dynamic network information, namely, Topology-Based approach (TOP) and Profile-Based approach (PROF)[10]. This information was used to estimate the traffic load in virtual network. Key ideas of these approaches are summarized as:

- TOP: uses static information for partition, such as the virtual network topology, link bandwidth, and latency. Each virtual node is weighted with the total bandwidth in and out of it. The optimization objective is to maximize the MLL between simulation engine nodes. This maximizes decoupling, supporting efficient parallel simulation.
- PROF: uses traffic profiling to obtain traffic-level information automatically from simulation experiments. The profiles are used to estimate future network use and determine the weights used. Because they are more accurate, this improves the network mapping. Typically profiling involves an initial simulation experiment using a naive initial partition and traffic monitoring. The simulation yields detailed traffic information, and improves subsequent network partitions.

3.4 Hierarchical Load Balance Approach

3.4.1 The Small Achieved MLL Problem

When we apply the TOP and PROF approaches to larger networks (e.g. 10,000 routers running on 100 nodes), neither of them gets satisfactory results. Checking the partition output manually reveals that the common reason for poor performance is that the achieved Minimal Link Latency (MLL) across partitions is too small when compared to the synchronization cost. This produces an overall simulation efficiency that is quite low. For example, for one network of 10,000 routers, the achieved MLL was only 0.1ms; far less than the synchronization cost of ~0.58ms for 100 simulation engine nodes (see Figure 5). In such a situation, the majority of the time will be spent doing synchronization – even perfect load balance would only moderate efficiency. This situation is quite different from the 1ms MLL for a 160 router network and 0.9ms synchronization cost for 8 simulation engine nodes in our previous experiments[10].

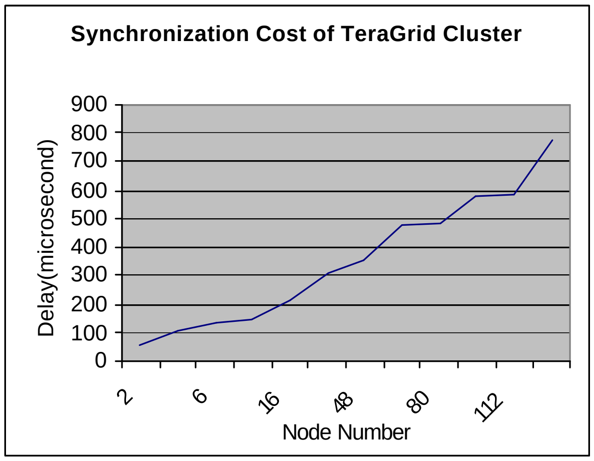

Figure 5. Synchronization Cost of the TeraGrid NCSA Cluster: the time used by the simulation engine nodes for global synchronization, which need to be executed every MLL time.

3.4.2 The Reason

The example above exposes a major problem with the existing load balance approaches. In TOP and PROF mappings, the link latency is converted to edge weight of the graph G, and smaller link latency leads to a larger edge weight. When the graph partitioner archives minimal edge-cut across partitions, it is less likely to partition across the link with small link latency, since it corresponds to a large edge weight. However, the optimization goal is the not the MLL, but the minimum edge-cut (the sum of all edge weights that cross partitions). When we have a large graph, the partitioner becomes less sensitive to the MLL, since even the large edge weight from a link with MLL may only be a small part of the final edge-cut.

We may tune the converting algorithm to make the edge weight of small link latency so large that it is unlikely to across partitions, but this highly depends on the network topology, the simulation engine node number, and the physical synchronization cost.

3.4.3 The Solution

To address the issue of small achieved MLL, we design a new hierarchical partition algorithm. To avoid partitions across edges with small link latencies, we remove edges with latency smaller than a threshold, LL, from the input graph (by merging nodes) to the partitioner and add them back to the partitioned output. In this way, we can guarantee the worst-case of MLL. However, this produces a new problem —how to choose the latency threshold, LL. If the threshold is too large, it will damage load balance, but if it is too small it will achieve a smaller MLL than possible. Instead of guessing the threshold, our approach is to simply try all reasonable thresholds, create a partition for each, evaluate these partitions, and then pick up the best partition. This is feasible because we can do the partition fast, even for large networks, and we can evaluate different partition outputs without running the simulation. The pseudo code for hierarchical partitioning is:

```
Input: graph G, partition N, and synchronization cost C
Output: the best partition P of graph G

Hierarchical Partition:
    Set the initial Threshold of MLL ($T_{mll}$)
    Loop through all reasonable $T_{mll}$:
        Get the dumped graph $G_d(T_{mll})$
        Partition the $G_d(T_{mll})$ using an existing partitioner,
        and get $P(T_{mll})$
        Evaluate the partition result $P_d(T_{mll})$
    Pickup the best partition $P_d(T_{mll})$
    Get the best partition P of original G
```

This algorithm requires the graph, G, the partition number, N, and the synchronization cost of the simulation engine nodes, C. Figure 5 shows the synchronization cost of the TeraGrid SDSC cluster, which is used for all simulations in this paper. We use the synchronization cost to set the initial threshold of MLL (T_{mll}) based on knowledge of the desired number of simulation engines. We require a T_{mll} to be larger than the synchronization cost, otherwise all time will be spent on synchronization, giving poor efficiency. Given the T_{mll}, the original graph G is reduced to a dumped graph G_d by collapsing nodes with link latency less then T_{mll} into a single node. Then any existing partition can be applied to the dumped graph G_d and get the partitioner output. By increasing the T_{mll} step by step (0.1ms in our experiments), we can get a sequence of partitions, and the remaining question is how to select amongst them.

To evaluate the candidate partitions, we use an efficiency metric *Efficiency* (E), which consists of two factors, Es and Ec. The first factor (Es) represents the efficiency decided by the achieved MLL and is calculated:

$$Es = (MLL - C_N)/MLL,$$

where C_N is the synchronization cost of N simulation engine nodes. The latter (Ec) represents the result of computational load balance and is calculated by:

$$Ec = C_{average}/C_{max},$$

where $C_{average}$ is the estimated average load (simulation event rate) on all nodes, and C_{max} is the max load of all nodes. The final efficiency E is Es * Ec, where larger values of E correspond to better partitions. Maximizing Es and Ec separately does not work because they represent the tradeoff between simulation efficiency and available parallelism. Larger Es means better simulation efficiency, but it also means less parallelism available, since smaller MLL leads to a more coarse-grained partition graph.

In summary, our hierarchical partitioning approach balances the parallelism and decoupling concerns in generating a good network partition. To do so, it generates and evaluates many possible partitions which is possible because we can create graph partitions and evaluate graph partitions quickly. The METIS graph partitioner[21] used in MaSSF can partition a graph with 10,000 vertexes in about 10 seconds. Thus it is fast enough to enable us to consider thousands of possible T_{mll}.

4. Simulation of Large Single-AS Network

To demonstrate the scalability of MaSSF and to study the performance of these partitioning and mapping approaches, we apply them to a range of network topologies and background traffic conditions. First we consider the simulation of a large flat network, which corresponds to a large single AS network and uses the OSPF protocol for routing (shortest path routing).

4.1 Evaluation Metrics

The first metric is the ***application simulation time T,*** which is the time taken to simulate an application in a specific network simulation. As faster simulation is the ultimate goal of our scalability studies, it is the most important metric.

To get deeper insight into the efficacy of our partition and load balance techniques, we also use three other metrics: achieved MLL, load imbalance, and parallel efficiency.

The second metric ***achieved MLL*** shows the effect of the hierarchical load balance approaches in increasing parallelism and is reported directly by the partitioner.

For the third metric ***load imbalance***, we define the load of a simulation engine node as the event rate of the simulation kernel (essentially one per network packet). Using these counters, we calculate the overall *load imbalance* across all the physical nodes in the actual simulation. Assuming the simulation kernel event rates are k_1, k_2, …, k_n, for n nodes used by the simulation engine, the load imbalance is normalized by the standard deviation of {k}.

The last metric is the ***parallel efficiency***, PE(N, L) for a problem of size L on N nodes is defined in the usual way[22] by $PE(N,L) = \frac{Tseq(L)}{N * T(L,N)}$,

where T(L, N) is the runtime of the parallel algorithm, and Tseq(L) is the runtime of the best sequential algorithm. Tseq(L) cannot be measured directly since the network is too large to be simulated on a single machine, thus, we approximate the Tseq(L) by

$$Tseq(L) = \frac{TotalEventNumber}{MaximalEventRateOnEachNode}.$$

4.2 Experimental Setup: Topologies, Traffic load, and Simulation Engines

We generate network topologies for our experiments with an adapted BRITE tool [23], a degree-based Internet topology generator following the Power-Law[24]. The flat network topology includes 20,000 routers and 10,000 hosts, which are spread over a geographic area of 5000milex5000mile (roughly the size of North American continent). This router count is comparable to the size of a large Tier-1 ISP, such as the AT&T network [25].

For background traffic, there are 8,000 clients continuously sending HTTP file requests to 2,000 servers. The average time gap between two successive requests of a client is 5 seconds and average file size is 50KB. Foreground traffic is created live from real Grid applications, including ScaLapack[26] and GridNPB3.0[27]. GridNPB3.0 is a set of grid benchmarks in a workflow style composition in data flow graphs encapsulating an instance of a slightly modified NPB task in each graph node, which communicates with other nodes by sending/receiving initialization data. GridNPB includes a range of computation types and problem sizes, and in our experiments we use the combination of Helical Chain (HC), Visualization Pipeline (VP), Mixed Bag (MB) applications, all run at class S size. These programs run for about 30 minutes on our platform.

The experiments use the TeraGrid Itanium-2 cluster for simulation engine nodes. The cluster nodes are dual 1.3GHz Itanium-2 processors with 2Gigabytes of memory, linked with Myrinet 2000 using MPICH-GM. We use 90 nodes as the simulation engines, and 7 nodes for application execution.

4.3 Results

Application workloads are executed on the single-AS network with moderate background traffic, and we study the performance of four mapping approaches: topology-based mapping (TOP), profile-based mapping (PROF), hierarchical topology-based mapping (HTOP), and hierarchical profile-based mapping (HPROF). The TOP and PROF partitioners achieve such small MLL that their performance is extremely poor and the simulations cannot be completed in a reasonable time limit. So we adjusted the link latency to edge weight converting algorithm for the large scale network simulation, so partitions are less likely to across edges with small link latency. It is not a general solution and has to be done according different topologies manually. The results are labeled as TOP2 and PROF2.

Application Simulation Time

The application simulation time of both applications is shown in Figure 6. For ScaLapack, the use of PROF2 mapping reduces overall simulation time of TOP mapping by 14%, and the use of the hierarchical mapping (HPROF) further reduces the simulation time up to 40%.

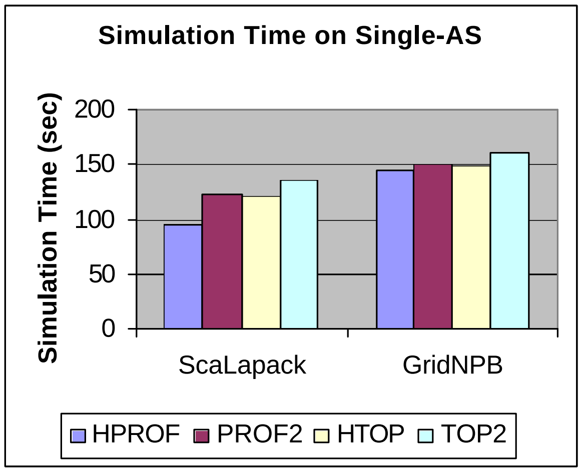

Figure 6. Simulation Time on the Single-AS Network

Achieved MLL

The achieved MLL is shown in Figure 7, and we can see both TOP2 and PROF2 still have much smaller MLL (about 0.6ms) comparing to HTOP and HPROF. It is clear that the hierarchical approaches can significantly increase the MLL, producing enough parallelism for large-scale simulation. These MLL values show that there is enough parallelism achievable for networks of ~20,000 routers in 5000M by 5000M area using 90 simulation nodes. These simulations will provide good efficiency with slowdown of 8 times.

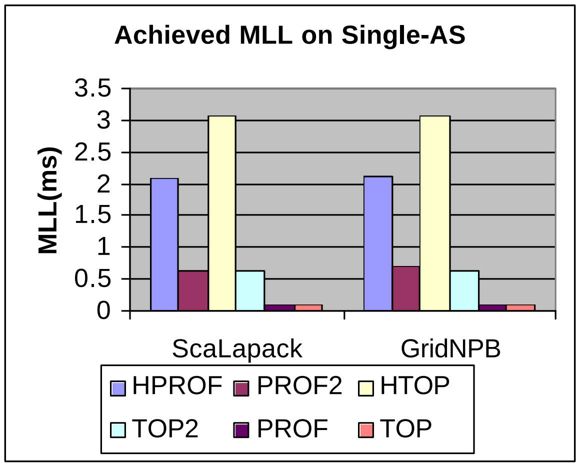

Figure 7. Achieved MLL on the Single-AS Network

Despite the fact that it produces the largest MLL (3ms), HTOP does not work very well compared to HPROF. The inaccurate load prediction in HTOP produces a much larger load imbalance which hurts performance.

Load Imbalance

The measured load imbalance for both applications is shown in Figures 8. The figure reports the normalized load imbalance across the physical simulation engine nodes for each combination of mapping approach and network topology. Each mapping approach produces significantly different results. Compared to TOP2, PROF2 improves load imbalance by about 7%. The HPROF mapping also improves the load imbalance by 11% over HTOP. It is clear that the use of detailed traffic information from a previous simulation execution provides a critical advantage in achieving effective network partitions.

It is also shown that the HPROF mapping produces better load balance than TOP2 and PROF2. This improvement is surprising because the hierarchical approaches use a simpler graph with coarse-grained node weights. So they should have less chance to achieve better load balance. We believe the explanation is that the underlying graph partitioner METIS does a better job for smaller graphs, since reduced graphs have many fewer vertexes.

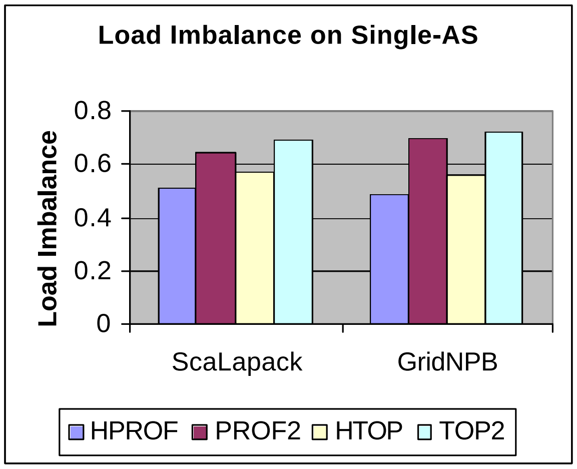

Figure 8. Load Imbalance on the Single-AS Network

Parallel Efficiency

The parallel efficiency of both applications is shown in Figure 9. While the overall efficiency of network simulation at this scale does not reach 100%, these values are excellent for parallel discrete event simulations on irregular loads. The HPROF for ScaLapack achieves about 40% parallel efficiency, a dramatic 64% improvement over TOP2. These levels of parallel efficiency enable effective large-scale network simulations.

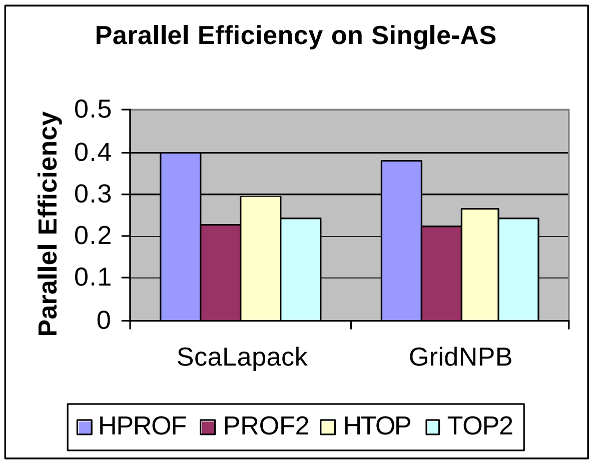

Figure 9. Parallel Efficiency on Single-AS Network

5. Simulation of Large Multi-AS Network

In Section 4 we demonstrate that the MaSSF and HPROF, a hierarchical profile-based partitioning algorithm, can produce scalable simulations for large single-AS networks. However, the Internet is not a flat network with shortest path routing. Instead, it is organized as a collection of ASes with traffic shaped by BGP policy routing. In such networks, connectivity does not mean reachability and the real dynamics are quite different from single-AS network. Such networks present greater challenges to achieving load balance because the traffic load is less coupled to network topologies. Despite its importance, to our knowledge multi-AS networks have never been simulated in large-scale because of the complexity involved.

5.1 Realistic BGP Routing Configuration

While there is much research on Internet-like topology generation [28-30], these studies focus on physical connectivity and pay little attention to routing configuration (particularly BGP). There are two major reasons for this situation. First of all, prior to our MaSSF simulator, no existing network simulator supports large scale simulation with detailed BGP routing. Simulators either have no support for BGP routing (DaSSFNet[31], ModelNet[9]), or they are limited by scalability to such a degree that BGP policy routing is less relevant (NS2[32], SSFNet[18, 33]). Second, real Internet BGP routing configurations are not publicly available, since the routing policy are closely tied to commercial contract terms that are considered highly confidential by ISPs. Fortunately, recent research has explored inferring AS relationships and BGP routing policy from publicly available information, such as BGP routing tables. Several of these efforts have made significant progress [34], making it possible for us to automatically generate realistic BGP routing policies into our network generator.

5.1.1 BGP Routing Policies

To generate realistic BGP routing, let us first check how the Internet routing is setup. One of the key features of BGP protocol is policy, which allows each AS to choose its own policy in accepting routes, selecting the best route, and announcing routes to its neighbors. Two kinds of routing policies are as follows: Import Policy and Export Policy.

Import Routing Policy: When receiving a route announcement from its neighbor, a router applies its import policies to the route, which include denying, or permitting a route, and assigning a local preference to indicate how favorable the route is. Local preference is used to differentiate routes received from different neighbors, since a BGP router may receive routes to the same destination from different neighbors and it must choose the best route to be used in its local routing table. BGP incorporates a sequential decision process to pickup the best route from a set of candidates to a given prefix. For example, the highest local preference, the shortest AS path, the lowest origin type, and the smallest MED for routes with the same next hop AS. There is a long list of criteria to set the preferential order of routes, and the first and the most important rule is the local preference. In practice, network administrators usually use local preference to enforce their import routing policies. According to [34], there are two general rules:

- **Route Preference between Provider, Customers, and Peers**: Network operators usually assign different local preferences to route learned from provider, customers, and peers. Customer routes have the highest local preference, and peer routes have higher local preference than providers.
- **Consistency of Local Preference with Next Hop ASes**: Operators may set local preference configuration based on prefix level or next hop AS level. Since it is easier to maintain the provider, customers, and peers preference based on next hop AS level, most ISPs use this approach in practice.

Export Routing Policy: BGP routers use export policies to decide which routes to be propagated to their neighbors. The policies usually are directly transformed from ASes relationships.

- **Exporting to a Provider**: An AS can export its local routes and routes of its customers, but can not export routes learned from its peers or providers
- **Exporting to a Peer**: An AS can export its local routes and routes of its customers, but can not export routes learned from its peers or other providers
- **Exporting to a Customer**: An AS should export all routes it knows to its customers

These basic export policy rules are the direct requirement of commercial agreements. For example, the first rule guarantees that a provider will not use its customer network to transit traffic, and the last rule guarantees that the customer can get full Internet access through its provider.

5.1.2 Using Heuristic Rules for Automatic Routing Configuration

Following the heuristic rules above, we automatically configure Internet-like network topologies with realistic routing configuration, and expect to get similar routing pattern of real Internet. The procedure of network topology generation and automatic routing configuration are shown in the following:

1) *Generate AS level topology following the Power Law*
2) *Classify ASes according connection degrees.*
    a. *Core: ASes with connection degrees of top 2*
    b. *Stub: ASes with connection degree of 1 or 2*
    c. *Regional ISP: all the other ASes*
3) *Decide AS relationships*
    a. *Provider-and-Customer:*
        i. *Core -- Stub,*
        ii. *Regional ISP – Stub,*
        iii. *Core – Regional ISP*
    b. *Peer-and-Peer: between all ASes in the same level*
4) *Setup Import Routing Policy*
    a. *Accept all incoming routes*
    b. *Set Local Preference according to Next Hop AS, which prefer routes from Customer, over routes from Peer, and over routes from Provider*
5) *Setup Export Routing Policy*
    a. *To Provider: Export local and Customer routes*
    b. *To Peer: Export local and Customer routes*
    c. *To Customer: Export all routes*
6) *Create topology for every Stub AS*
    a. *Follow the Power Law*
    b. *Use OSPF routing inside the AS*
    c. *Use default routing to hosts outside local AS*
    d. *Pickup default/backup routers for multi-homed ASes*

This is just a high level abstract of our implementation in the maBrite[1] topology generator, which is based on BRITE tool. To create a real functional topology, there are more details need to be addressed. For example, at Step 3, we must guarantee that every non-Core AS has a path including Provider-and-Customer links to a Core AS so that this AS has full connection to the whole network. Furthermore, we should also guarantee that the Core ASes form a clique as observed for the Dense Cores, and additional links between Core ASes are added when necessary.

After the AS relationships are defined, the routing policy setup is straightforward. The only problem is how these policies can be expressed in the simulator input Domain Model Language (DML) file. For a detailed discussion of this, the interested reader is referred to the MicroGrid user manual.

The last thing we want to emphasize is the default routing embodied in Step 6. It is very important to use default routing

[1] The maBrite package is available at http://www-csag.ucsd.edu/projects/grid/microgrid.html

in Stub ASes so the huge external BGP routes need not be injected into the OSPF routing tables. This approach can reduce the overhead of Stub AS routers greatly and is widely used in real world practice.

5.2 Hierarchical Network Simulation Results

These experiments evaluate both flat and hierarchical load balance approaches for large-scale multi-AS networks with Internet-like routing configurations.

5.2.1 Experiments Setup

The network topology is created by our maBrite topology generator with BGP routing configuration as described above. It includes 100 ASes, each containing 200 routers. In addition, 10,000 hosts are randomly attached to Stub ASes for background traffic generation and live traffic agent. All these routers and hosts are spread to a geographic area of 5000milex5000mile.

We use the same background traffic and the same application, ScaLapack and GridNPB, in the experiments as described in Section 4. We also use the same TeraGrid Itanium-2 cluster, 90 nodes as the simulation engines, and 7 nodes for application execution.

5.2.2 Results

Application workloads are executed on the multi-AS network with moderate background traffic, and we evaluate the performance of four mapping approaches: TOP, PROF, HTOP, and HPROF. Again, both TOP2 and PROF2 mappings are tuned for the large scale network simulation.

Application Simulation Time

The simulation time of both applications is shown in Figure 10. For ScaLapack, the use of PROF2 mapping reduces overall simulation time of TOP2 mapping by 21%, and the use of the hierarchical mapping (HPROF) further reduces the simulation time up to 41%. The GridNPB has less improvement, since it has less communication compared to ScaLapack.

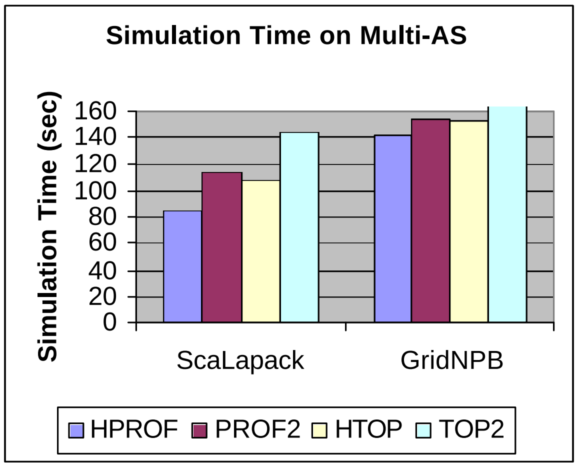

Figure 10. Simulation Time on the Multi-AS Network

Achieved Minimal Link Latency

The achieved MLL is shown in Figure 11. Like on the Single-AS network, the original TOP and PROF produce small MLL's and our data reflects the resulting poor simulation efficiency. The hierarchical approaches achieve much larger MLL's, in some cases ten times larger. MLL's of this size support good simulation efficiency.

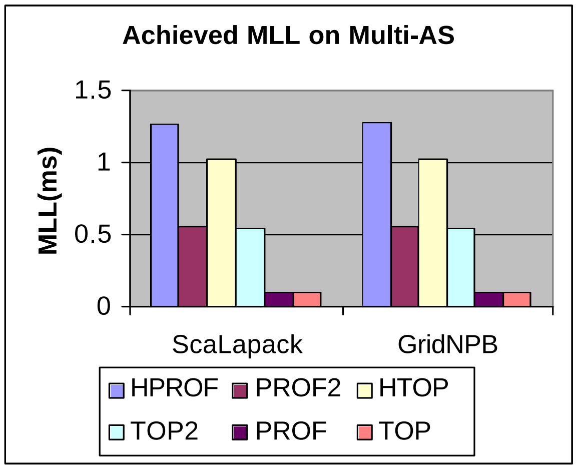

Figure 11. Achieved MLL on the Multi-AS Network

Load Imbalance

The measured load imbalance for ScaLapack and GridNPB is shown in Figure 12. The figure reports the normalized load imbalance across the physical simulation engine nodes for each combination of mapping approach and network topology. Each mapping approach produces significantly different results. Compared to the TOP2 mapping, the PROF2 mapping improves the load imbalance by about 15%. The HPROF mapping improves the load imbalance over HTOP by 31%.

As we anticipated, the load imbalance for this multi-AS network is much larger than the single-AS network due to the use of BGP routing, and it makes the improvement from profile-based techniques significant compared to that of the single-AS network in Section 4.3.

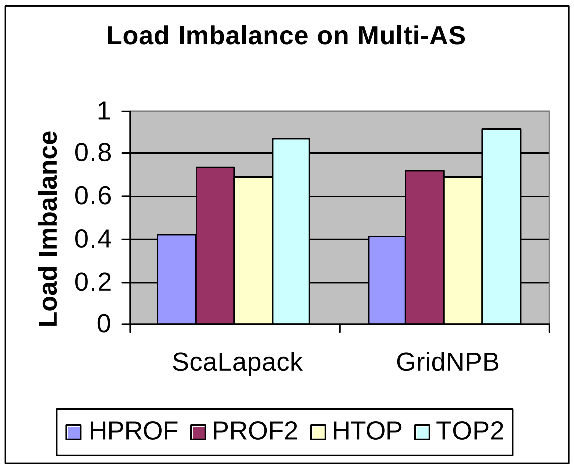

Figure 12. Load Imbalance on the Multi-AS Network

Parallel Efficiency

The parallel efficiency of the simulation of both applications is shown in Figure 13. While the overall efficiency of network simulation does not approach 100%, HPROF for ScaLapack can achieve about 40% parallel efficiency, about a 64% improvement from TOP2. This level of parallel efficiency enables simulation of large-scale Multi-AS networks.

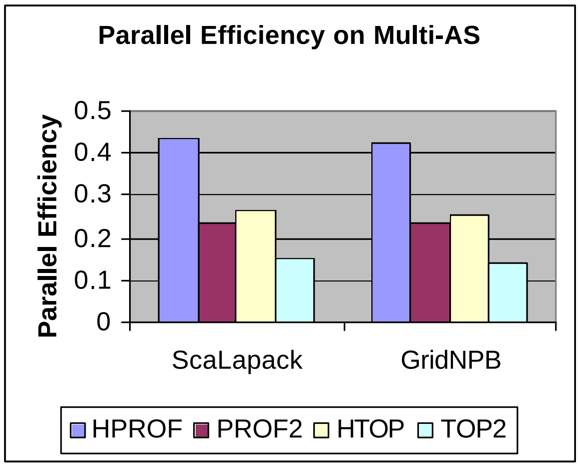

Figure 13. Parallel Efficiency on Multi-AS

In summary, these experiments show that our hierarchical load balance approaches still work well for large multi-AS networks with realistic BGP routing configuration.

6. Discussion and Related Work

Several recent research efforts are most related to the MicroGrid/MaSSF, including Albatross[35], Emulab[8], and ModelNet[9]. While these systems also support execution of real application over a modeled network, there are significant differences between these efforts and the MicroGrid. The network modeling in these systems either use approximation models[36] or have limited scalability[32]. These approximations reduce the simulation accuracy (compared to MicroGrid's global synchronized packet-level simulation) to achieve faster execution. For example, Emulab uses a set of real routers, switches and configurable software routers to emulate wide area network. This approach has the advantage of speed of emulation, but provides little in the way of detailed control of speed and modeling to the experiment designer. The largest automatically-configured Emulab experiment[21] we are aware of has 520 virtual nodes (routers) mapped to 44 PCs. The ModelNet project at Duke University (and now also at UCSD) is a software emulator. Their approach to scalability simplifies both network topology (a network of pipes) and routing (assuming a simple routing protocol based on shortest path) and then maps the resulting network of queues onto a set of emulation *cores*. This summarized network is an approximation to actual detailed network behavior. Further, there is no synchronization between these cores, so the number of cores can be used without affecting accuracy is unknown. In contrast, MaSSF uses full-scale detailed packet simulation based on a distributed discrete-event simulation (PDES) engine. The largest emulation on ModelNet we know has 1120 virtual nodes (routers) on 4 cores. While there have been many efforts which use PDES for network simulation[37], we know of no other modeling efforts that achieve detailed online network simulation of the documented scale.

Load balance is known to be an important problem for the scalability of distributed network simulations or emulations, however there are only a few efforts in network simulation/emulation community to solve this problem. Many projects use either manual partitioning or simple graph partitioning based on network topology. The DaSSF simulator uses the METIS graph partitioning package and link latencies for load balance. It does not use link capacities or any further detailed traffic information. ModelNet[38] uses the greedy k-cluster algorithm: for k nodes in the core set, randomly selects k nodes in the virtual topology and greedily selects links from the current connected component in a round-robin fashion. They also use an approach similar to our PLACE mapping, but it is focused on minimizing Network traffic between *cores*. Emulab's *assign* maps virtual topologies which include endpoint resources as well as network structures onto a heterogeneous combination of routers, switches, and computers. Critical issues are time to compute mapping, physical resources used, and sufficient link capacity. Thus, *assign* chooses specific endpoint and network resources to optimize their quantity subject to the constraints. Load balance is not a direct focus.

Realistic topologies are of considerable importance to network and Grid application studies. There are mainly two types of topology generators in use, hierarchical and degree-based. Hierarchical generator like Tiers and Transit-Stub are more close to the logical structure of the Internet. The degree-based generators like Inet[29], BRITE[28], and PLRG[39] generate graph that follow the Power-Law and are more close to the physical connectivity of real Internet, but have less clear hierarchical structure. The latest GridG[40] generator tries to combine both approaches by enhancing a hierarchical graph to following Power-Law. The goal is quite similar to our approach to topology, but we achieve it by setting up hierarchical relationship out of a graph created from the Power-Law. However, all previous efforts focus on physical connectivity generation, and none of them provide realistic routing configurations based on BGP4 policy-based inter-domain routing. Exploiting recent research which infers AS relationship and BGP routing policy from publicly available information such as BGP routing tables [34, 41] our maBrite generator automatically builds realistic BGP routing policy in topology generator. To the best of our knowledge, we are the first to provide this kind of topology generator. These heuristic rules can also be used to enhance other topology generators.

7. Summary and Future Work

Large-scale and realism are two critical requirements for network simulation for Grid application studies. In this paper, we first study networks 100x larger than in our previous studies (20,000 routers). Then, at this scale, we study realistic network structures (100 AS's, BGP4 and OSPF routing) versus flat OSPF routing. Finally, we describe and evaluate a new profile-based load-balancing approach called hierarchical profile-based load balance (HPROF). These load balance approaches are evaluated against large-scale networks, including both single-AS network and multi-AS network. The best of these, HPROF, can improve the load imbalance by 40% and reduces the simulation time by about 50%. This provides a great chance for scalable network simulation. We also provide an Internet-like topology generation with realistic BGP routing configuration. Combining with our packet-level hop-by-hop network simulator and detailed BGP4 protocol support, we demonstrate that we can provide realistic large-scale network simulation for networks including about 20,000 routers.

Our automatic BGP configuration is based on a set of heuristics used in by many network administrators. While we believe our method captures the major components of realistic BGP configuration and routing, a natural next step is to validate it directly. One such approach would be to use the AS level topology of the real Internet [25] and feed it into our BGP configuration procedure, allowing direct comparison of routing in the Internet and our generated configuration. Two types of studies will be valuable for the validation. The first is to compare the static status of BGP routing, such as the similarity of route entries in BGP routing table. The second is to compare the dynamic behavior of BGP. For example, there is a Beacon project [42] which automatically announces/withdraws a prefix at a given time every day. And we can observe what real BGP does to beacon activities from a public observation point. Both of these studies can be simulated in MaSSF.

Due to physical resource limitation, we only use a 128-node cluster in our experiments. However, it is clear that there is still more parallelism in the large-scale network simulation. In future work, we will use MicroGrid to study larger networks and application, specifically using a 256-node Itanium-2 Linux cluster to simulate a network with 100,000 network entities, which can be taken as a significant fraction of the real Internet with hundreds of ASes. Under this scale of a network, we expect to experience much larger load balance challenge and we have to develop a traffic-based load balance solution for better scalability. While we selected the GridNPB benchmarks for our experiments, our evaluation could be improved by studies with better benchmarks suites or larger real grid applications. In the future, we will also use MicroGrid to study larger scale real Grid applications, including resources scheduling and overlay network behaviors.

8. Acknowledgement

Supported in part by the National Science Foundation under awards NSF EIA-99-75020 Grads and NSF Cooperative Agreement ANI-0225642 (OptIPuter), NSF CCR-0331645 (VGrADS), NSF ACI-0305390, and NSF Research Infrastructure Grant EIA-0303622. Support from Hewlett-Packard, BigBangwidth, Microsoft, and Intel is also gratefully acknowledged.

The authors also acknowledge the contributions of Anil Kapur, Huaxia Xia, and Alex Olugbile to the system infrastructure which made this work possible.

9. References

1. Tao Ye, Shivkumar Kalyanaraman, David Harrison, Biplab Sikdar, Bin Mo, Hema Tahilramani Kaur, Ken Vastold, and Boleslaw Szymanski, *Network Management and Control Using Collaborative On-line Simulation.* Proc. IEEE International Conference on Communications,, June 2001.
2. D. Katabi, M. Handley, and C. Rohrs. *Internet congestion control for future high bandwidth-delay product environments*. in *Proc. ACM SIGCOMM*. 2002. Pittsburgh, PA.
3. Christina Parsa and J.J. Garcia-Luna-Aceves. *Improving TCP Congestion Control over Internets with Heterogeneous Transmission Media*. in *Proceedings of the 7th IEEE International Conference on Network Protocols (ICNP)*. 1999.
4. Andy Oram, *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*. March 2001: O'Reilly.
5. Ian Foster and Carl Kesselman, *The Grid: Blueprint for a New Computing Infrastructure*. 1999: Morgan Kaufmann.
6. L. Ni P. Zheng. *EMPOWER: A Network Emulator for Wireline and Wireless Networks*. in *IEEE InfoCom 2003*. 2003. San Francisco.
7. Russell Bradford Rob Simmonds, and Brian Unger. *Applying parallel discrete event simulation to network emulation*. in *14th Workshop on Parallel and Distributed Simulation (PADS 2000)*. May 28-31, 2000. Bologna, Italy.
8. Brian White, Jay Lepreau, Leigh Stoller, Robert Ricci, Shashi Guruprasad, Mac Newbold, Mike Hibler, Chad Barb, and Abhijeet Joglekar. *An Integrated Experimental Environment for Distributed Systems and Networks*. in *Proceedings of 5th Symposium on Operating Systems Design and Implementation (OSDI)*. 2002.
9. Amin Vahdat, Ken Yocum, Kevin Walsh, Priya Mahadevan, Dejan Kostic, Jeff Chase, and David Becker. *Scalability and Accuracy in a Large-Scale Network Emulator*. in *Proceedings of 5th Symposium on Operating Systems Design and Implementation (OSDI)*. December 2002.
10. Xin Liu and Andrew Chien. *Traffic-based Load Balance for Scalable Network Emulation*. in *SuperComputing 2003*. Noverber 2003. Phoenix, Arizona: the Proceedings of the ACM Conference on High Performance Computing and Networking.

11. Xin Liu, Huaxia Xia, and Andrew Chien, *Validating and Scaling the MicroGrid: A Scientific Instrument for Grid Dynamics*. Journal of Grid Computing, 2003.
12. C. Walshaw, M. Cross, S. Johnson, and M. Everett. *JOSTLE: Partitioning of Unstructured Meshes for Massively Parallel Machines*. in *Parallel CFD'94*. 1994. Tyoto, Japan.
13. F. Pellegrini and J. Roman. *SCOTCH: a software package for static mapping by dual recursive bipartitioning of process and architecture graphs*. in *High-performance Computing and Networking, Proc. HPCN'96*. 1996. Springer, Berlin.
14. Preis R. and Diekmann R, *PARTY - A Software Library for Graph Partitioning.* Advances in Computational Mechanics with Parallel and Distributed Processing, 1997: p. 63-71.
15. B. Hendrickson, *Graph Partitioning Models for Parallel Computing*. Parallel Computing Journal, 2000. **26**(12): p. 1519--1534.
16. H. Song, X. Liu, D. Jakobsen, R. Bhagwan, X. Zhang, K. Taura, and A. Chien. *The MicroGrid: a Scientific Tool for Modeling Computational Grids*. in *IEEE Supercomputing (SC 2000)*. 2000. Dallas, USA.
17. Jason Liu and David M. Nicol. *Learning Not to Share*. in *Proceedings of the 15th Workshop on Parallel and Distributed Simulation (PADS 2001)*. 2001. Lake Arrowhead, CA.
18. James Cowie, Hongbo Liu, Jason Liu, David Nicol, and Andy Ogielski. *Towards Realistic Million-Node Internet Simulations*. in *Proceedings of the 1999 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'99)*. June 28 - July 1, 1999. Las Vegas, Nevada.
19. Sam Halabi, *Internet Routing Architectures Second Edition*. 2001: Cisco Press.
20. Lakshminarayanan Subramanian, Sharad Agarwal, Jennifer Rexford, and Randy H. Katz. *Characterizing the Internet Hierarchy from Multiple Vantage Points*. in *IEEE Infocom*. 2002.
21. Shashi Guruprasad, Leigh Stoller, Mike Hibler, and Jay Lepreau. *Scaling Network Emulation with Multiplexed Virtual Resources*. in *SIGCOMM 2003 Poster Abstract*. 2003.
22. V. Kumar, A. Grama, A. Gupta, and G. Karypis, *Introduction to Parallel Computing - Design and Analysis of Algorithms*. 1994: The Benjamin/Cummings Publishing Company.
23. Anukool Lakhina Alberto Medina, Ibrahim Matta, and John Byers. *BRITE: An Approach to Universal Topology Generation*. in *In Proceedings of the International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems- MASCOTS '01*. 2001. Cincinnati, Ohio.
24. Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. *On Power-Law Relationships of the Internet Topology*. in *SIGCOMM*. 1999.
25. Neil Spring, Ratul Mahajan, and David Wetherall. *Measuring ISP Topologies with Rocketfuel*. in *ACM SIGCOMM*. 2002.
26. A.Petitet, S.Blackford, J.Dongarra, B.Ellis, G.Fagg, K.Roche, and S.Vadhiyar. *Numerical Libraries and the Grid: The GrADS Experiment with ScaLAPACK*. in *International Journal of High Performance Computing Applications*. 2001.
27. Rob F Van Der Wijngaart and Michael Frumkin, *NAS Grid Benchmarks Version 1.0*. 2002, NASA Ames Research Center.NAS-02-005
28. Alberto Medina, Anukool Lakhina, Ibrahim Matta, and John Byers. *BRITE: An Approach to Universal Topology Generation*. in *In Proceedings of the International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems- MASCOTS '01*. 2001. Cincinnati, Ohio.
29. J. Winick and S. Jamin, *Inet-3.0: Internet topology generator*. 2002, University of Michigan Ann Arbor.CSE-TR-456-02
30. K.L. Calvert, M.B. Doar, and E.W. Zegura, *Modeling Internet Topology*. IEEE Communications Magazine, June 1997. **36**(6): p. 160-168.
31. DaSSFNet Homepage.http://www.cs.dartmouth.edu/~ghyan/dassfnet/overview.htm
32. Lee Breslau, Deborah Estrin, Kevin Fall, Sally Floyd, John Heidemann, Ahmed Helmy, Polly Huang, Steven McCanne, Kannan Varadhan, Ya Xu, and Haobo Yu, *Advances in Network Simulation*. IEEE Computer, May, 2000. **33**(5): p. 59-67.
33. SSFNet Webpage.http://www.ssfnet.org
34. F. Wang and L. Gao. *Inferring and Characterizing Internet Routing Policies*. in *ACM SIGCOMM Internet Measurement Conference*. 2003.
35. T. Kielmann, H. Bal, J. Maassen, R. van Nieuwpoort, L. Eyraud, R. Hofman, and K. Verstoep, *Environments for High-Performance Grid Computing: the Albatross Project*. Future Generation Computer Systems, 2002. **18**(8).
36. L. Rizzo. *Dummynet and Forward Error Correction*. in *Proc. of the 1998 USENIX Anuual Technical Conf.* June 1998. New Orleans, LA: USENIX Association.
37. Rob Simmonds, Russell Bradford, and Brian Unger. *Applying parallel discrete event simulation to network emulation*. in *14th Workshop on Parallel and Distributed Simulation (PADS 2000)*. May 28-31, 2000. Bologna, Italy.
38. Ken Yocum, Ethan Eade, Julius Degesys, David Becker, Jeff Chase, and Amin Vahdat. *Toward Scaling Network Emulation using Topology Partitioning*. in *Proceedings of the International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*. 2003.
39. W. Aiello, F. Chung, and L. Lu. *A random graph model for massive graphs*. in *ACM Symposium on Theory of Computing*. 2000.
40. Dong Lu and Peter A. Dinda. *Synthesizing Realistic Computational Grids*. in *SuperComputig 2003*. November 2003. Phoenix, Arizona.

41. L. Gao, *On Inferring Automonous System Relationships in the Internet*. IEEE/ACM Transactions on Networking, 2000.
42. Z. Morley Mao, Randy Bushy, Timothy G. Griffinz, and Matthew Roughan. *BGP Beacons*. in *IMC'03*. October 2003. Miami Beach, Florida.