Modeling Large-Scale Platforms for the Analysis and the Simulation of Scheduling Strategies

Henri Casanova
San Diego Supercomputer Center,
Dept. of Computer Science and Engineering,
University of California, San Diego
9500 Gilman Dr., La Jolla, CA 92093-0114, U.S.A.
casanova@cs.ucsd.edu

Abstract

An important trend in scientific computing is the establishment of computing platforms that span multiple institutions to support applications at unprecedented scales and levels of performance. A key issue for achieving high performance is the scheduling of application components onto available resources, which has been an active area of research for several decades. However, most of the platform models traditionally used in scheduling research, and in particular the network models, break down for platforms spanning multiple (wide-area) networks. In this paper we examine modeling issues for large-scale platforms. More specifically, we discuss network latency, bandwidth sharing, and network topology. Our discussion is from the perspective of scheduling research and the main challenge we address is to develop models that are sophisticated enough to be realistic, but simple enough that they are amenable to analysis. Finally, while the models we propose can be used to study scheduling problems directly, they also form a good basis for realistic simulation, which is typically the method of choice for comparing scheduling strategies.

This material is based upon work supported by the National Science Foundation under Grant ACI-0204007.

1. Introduction

Parallel and distributed application scheduling, i.e. the decision process by which components of an application are assigned to resources that are distributed over a network, has been an exceptionally active research area for several decades. A scheduling problem is generally defined by three components: (i) an *application model* that specifies the application's structure and requirements; (ii) a *platform model* that specifies the nature of the available resources and of the network by which they are interconnected; and (iii) an *objective* that must be achieved, such as minimizing application execution time, minimizing cost, maximizing execution time predictability, etc.

A trend in high performance computing (HPC) is to establish computing platforms that span large networks. The goal is often to support applications at scales that are beyond what can be achieved at a single site or institution. This *Grid Computing* [11] approach has been made possible through the development of appropriate middleware services [13, 12] and a number of Grid platforms have been put in production [32, 27, 8]. In this paper we discuss issues pertaining to the modeling of large-scale platforms from the perspective of both analysis and simulation for developing and evaluating scheduling strategies.

One key difference between these platforms and more traditional parallel computing platforms (e.g., clusters, MPPs) is the nature of the network.

The properties of wide-area networks are radically different than those of, say, a switch within an MPP, and must be well understood to instantiate realistic platform models. There has traditionally be an intellectual disconnect between the wide-area networking community and the scheduling community: while the former focuses on the effect of network protocols on global Internet traffic and advanced networking functionality provided by new protocols, the latter is generally concerned with the performance of a single application and researcher typically use simplistic network models. Some of these models, while perhaps appropriate for traditional parallel computing platforms, become radically inadequate for wide-area networks. In this paper we propose new models that capture some of the properties of wide-area networks that are essential for designing scheduling strategies relevant to practice.

The models discussed in the paper can be used directly as the basis for developing and analyzing new scheduling algorithms. However, it is often impossible to obtain analytical results concerning the efficacy of different scheduling algorithms. One possible approach would be to perform experiments on real-world platforms, but it is difficult to perform reproducible and diverse experiments on wide-area computing platforms. Consequently, simulation has long been the method of choice for evaluating the relative merit of competing scheduling strategies. Therefore, in the face of new emerging computing platforms, there is a need for a simulation framework for scheduling research that strikes a sound balance between realistic and fast simulations. In this view, we have implemented most of the models discussed in this paper as part of the SimGrid simulation toolkit [1, 19].

2. Network Modeling

In this section we discuss three aspects of networks that must be taken into account in platform models: (i) network latencies; (ii) bandwidth sharing; and (iii) network topology. For each we describe the issue, briefly review and evaluate the traditional approach in traditional scheduling work, and propose new models when necessary.

2.1. Network Latencies

It is well-known that a reasonable approximation of the time required to send x bytes of data over a network link is affine of the form $\alpha + x/\beta$, where α is the *latency* (i.e., the time required for a zero byte message to travel form the source to the destination), and β is the data transfer rate (in fact, models such as LogP [7] proposed more sophisticated models as early as ten years ago). Nevertheless, many scheduling works have assumed linear transfer times x/β, for instance in the divisible load scheduling area [4, 2, 21]. Most of these works were actually developed after the publication of the LogP model, but assuming a linear transfer time makes it possible to obtain elegant solutions to certain scheduling problems. However, ignoring latencies may lead to flawed solutions as there is no prohibitive cost to sending large numbers of very small messages. For instance, in [2], which develops a multi-round divisible load scheduling algorithm, it is noted that the linear model implies an infinite number of rounds where an infinitesimal amount of work is sent out at each round. This is clearly impractical and the authors point out that a "reasonable" number of rounds should be selected. Without a model that takes latencies explicitly into account it is difficult to quantify what may be reasonable. For instance, one could decide on a minimal message size that can be scheduled, but it is not clear how to choose the best such minimal size. Typically, as seen for instance in [33], taking latencies into account adds a significant amount of complexity to the scheduling problem.

One may wonder whether the fixed part α of the transfer time is actually significant when compared to the proportional part x/β. While this of course depends on the message size x, the current trend indicates that while latencies are bounded below by the speed of light, network link bandwidth increases at an exponential rate. For instance, the TeraGrid platform [32] has established a 40GBit/sec dedicated link between the San Diego Supercomputer Center (SDSC) and the National Center for Supercomputing Applications (NCSA). The expected network latency is in the

100ms range. A back of-the-envelope calculation assuming that a data transfer could use the full bandwidth (e.g., with parallel TCP connections, tuned congestion windows) gives that one third of the time to transfer 1 GByte of data is due to the network latency on the TeraGrid. This is only a coarse estimate, but it is indicative of the trend for these types of networks. Therefore, latencies can indeed be significant and even dominate communication costs and, for some applications, it is thus imperative that scheduling strategies take these latencies into account.

. It is interesting to note that latencies can also be experienced for computation. Indeed, some applications require that many (computational) processes be initiated on remote resources, which generally involves calls to middleware services to perform authentication, resource acquisition, process creation, etc. This overhead can be significant. For instance, data obtained as part of the GRASP project [6] shows that using the Globus Toolkit version 2.0 [15] to launch a no-op job on a remote compute resource could require up to 25 seconds. This suggests that an affine model for computation times applies even, for instance, when the number of cycles required is linear in the input data size. We have used an affine model for computation in some of our recent work [35, 34, 33]. .

Accordingly, the SimGrid toolkit provides ways to define affine models both for communication and computation times.

2.2. Bandwidth Sharing

A key difference between local-area and wide-area networks is their *bandwidth sharing* behaviors. Let us first review briefly the traditional models used in the scheduling literature concerning the sharing of network resources. The typical assumption is that processors are interconnect in a point-to-point fashion by network links. In the "one-port" model a processor can only send data to one other processor at a time, while in the "multi-port" model a processor can send data to multiple processors simultaneously. In both cases, it is assumed that a single data transfer may occur on a given network link at a time. With these assumptions, a fully connected topology can represent a switch within a cluster, but does not model a shared communication medium such as an Ethernet network for instance. Arguably, a schedule could enforce that only one single communication happens at a time on a given network link, thereby hiding the fact that network links can be shared. This however has two major limitations.

The first limitation stems from the fact that distinct logical network links between hosts often correspond to shared physical links. Therefore, although a simultaneous communications between processor A and B, and between C and D, may contend for one or more networking resource and thereby make the non-shared model invalid in spite of the constraints imposed by the schedule. This in fact alludes the question of the topology of the network, which we discuss in Section 2.3.

The second limitation is that allowing network sharing can in fact be beneficial to an application, which we illustrate here with a simple example. Consider an on-line scheduling problem in which requests for computation arrive at a server and must be dispatched to any of several identical worker processors over a network link. Say that a request arrives at time t for a job that requires 10 minutes of data transfer for 10 minutes of computation, and that a request arrives at time $t + 5$ minutes for a job that requires 1 minute of data transfer time for 19 minutes of computation. Finally, say that the performance metric is the average slowdown over all submitted jobs, where the slowdown is the ratio of a job's effective turn-around time and of the turn-around time that could be achieved if the platform where dedicated to that job. This metric is commonly used for on-line scheduling problems and should be minimized. Three possible scheduling strategies are: (i) first come first serve; (ii) wait until both jobs have arrived and schedule the second job first; and (iii) start jobs exactly when they arrive allowing simultaneous communication on the network link and assuming that each communication gets half of the available bandwidth. Other strategies are possible but can easily be shown to reduce to or be outperformed by one of the above. In this example, the average slowdown obtained for

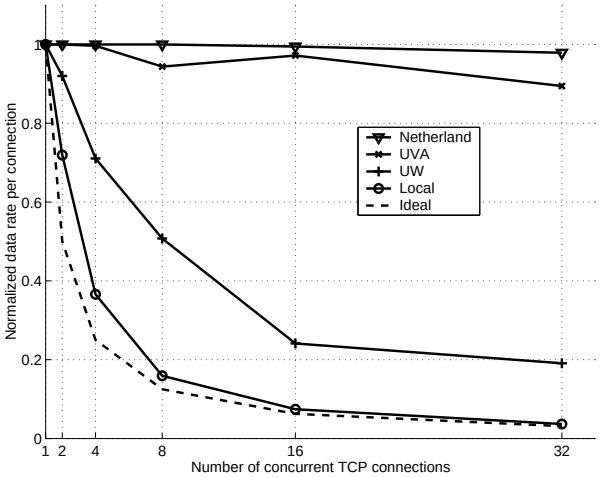

Figure 1. Bandwidth-sharing experiments between a host at UCSD and one in the Netherlands, UVA, UW, and the local-area network.

the three strategies are respectively 1.125, 1.175, and 1.050, showing that allowing the link to be shared is the best option. Note that another possibility would be to allow for interruptible communications by which any communication can be stopped and resumed at will, which can theoretically achieve the same average slowdown as strategy (ii) if done appropriately but poses several implementation problems and incurs overhead.

2.2.1. A Simple Bandwidth Sharing Experiment

The (admittedly contrived) example above uses the simple assumption that when x connections occur simultaneously on a network link with available bandwidth B (in Mbit/sec) then each connection proceeds at B/x Mbit/sec. This model has been used in most previous work and is representative of what can be observed, for example, on a non-switched Ethernet local-area network. This model however is not adequate for wide-area networks, mostly due to the implementation of TCP. Figure 1 shows the results from experiments in which we performed data transfers of 100MB files over TCP between a host in our lab at the University of California, San Diego, and hosts in that same lab ("Local"), at the Delft Technical University, Netherlands ("Netherlands") at the University of Virginia ("UVA"), and at the University of Washington ("UW"). For each of these destinations we initiated from 1 to 32 simultaneous TCP connections and the graph in Figure 1 plots the achieved data transfer rate **per connection** versus the number of connections. The data transfer rate is normalized that that achieved when only one connection is used. In addition the graph plots the ideal bandwidth-sharing model described above (dashed line).

We can make the following observations. First, the ideal bandwidth-sharing model, while accurate for the "Local" experiment, is widely inappropriate for any of the wide-area transfers. In fact, the curves for the two most distant sites, Netherlands and UVA, are roughly flat, meaning that connections beyond the first one get bandwidth "for free". The behavior for the wide-area transfers is due to multiple factors. For instance, no matter how much bandwidth may be available on an Internet backbone the portion of that bandwidth used by a TCP connection is limited by the sender's congestion window. Also as backbone links typically support very large numbers of connections the contention among the connections of a single application is often effectively negligible, at least up to a point.

2.2.2. An Empirical Model for Bandwidth Sharing

Based on these results, we conclude that the simple bandwidth sharing model is not applicable to wide-area links. Furthermore, a reasonable model for these links is that each concurrent connection of an application gets assigned the same amount of bandwidth (we will refine this statement in Section 2.3). The SimGrid simulator makes it possible to instantiate network links that follow either the simple bandwidth sharing model, or the backbone model. This provides the foundation for differentiating the contention among data transfers of an application over both local-area and wide-are networks. Note that the curve corresponding to the UW experiment in Figure 1 follows neither model but exhibits somewhat of a hybrid behavior. We have also observed this behavior for experiments between UCSD and the

University of California, Santa Barbara. This suggests that a more appropriate model could be one that allows some type of exponential decay of the bandwidth per connection, which we will investigate in future work. Finally, note that our models ignore TCP "slow-start" behavior and models only bandwidth sharing in steady state, which may not be accurate for short messages.

2.3. Network Topology

The backbone model in the previous section suggests that an infinite number of connections could be established "for free" over a backbone link, which is clearly not the case in practice. In fact, a host that participates in a distributed computation is never attached directly to a backbone link, but has a network card with some limited capacity, which is itself connected to possibly multiple local-area links via a number of routers and eventually to the backbone. Therefore, when many connections are opened from a host to remote hosts, the network bottleneck may be the host's network card for instance, which limits the number of connections that can be used effectively. This effect was not seen in Figure 1 given the moderate number of connections and the relative speeds of the backbone link and the local links. Furthermore simultaneous communication from a single host to different sites (say from UCSD to both UCSB and UW) share network resources, in this case the network card and probably network links all the way to the first backbone link, and maybe other backbone links beyond the first one. Similarly, communication emanating from different hosts within a site most likely share local-area network resources on their way to the Internet backbone.

Therefore, it is clear that a reasonable network model must consider a sequence of links, or a *path*, rather than only single-link connections. Note that these links can be logical links that each model a set of physical links, as further discussed in Section 3. The question then arises of how to model data transfers over network paths. One possibility could be to use a store-and-forward model by which a message is sent in its entirety through each link in sequence. However, this is extremely unrealistic because in fact messages are split into packets and packet transfers are pipelined over the network links. One possibility would then be to model the network at the packet level. While this can be done for simulation (e.g., as in NS [28]), the resulting complexity precludes the analysis necessary for the development of scheduling algorithms.

2.3.1. A Simple Macroscopic Model of TCP

The key observation here is that, due to the aforementioned pipelining of messages, the general behavior is that a data transfer over a network path goes at the speed of the slowest link, i.e. the bottleneck link, on that path. This makes it possible to develop macroscopic models of bandwidth-sharing. Such models have been derived via an analogy between network connections, or *flows*, and fluids in pipes while ignoring the packet granularity [23] and several authors have proposed theoretical models for TCP bandwidth allocation among flows [5, 10, 24]. These works show that bandwidth sharing among TCP flows is not fair but in fact depends on the network latencies experienced by the different flows. Simply put, two flows competing over a same bottleneck link receive bandwidth approximately inversely proportional to their round trip times, which is proportional to the link's latency.

Building on these considerations we have designed an efficient algorithm for simulating TCP flows competing over multi-path routes. Due to lack of space we only give here the main ideas and we refer the reader to [3] for all details. The algorithm first considers all links and determines bottleneck links for some flows. These flows are assigned bandwidth on these links inversely proportionally to their round-trip times (remember that as seen in Section 2.1 network link latencies are a fundamental part of the network model). These flows consume this bandwidth end-to-end, and thus our algorithm reduces the bandwidth capacity of links traversed by all these flows in the network accordingly. This process is repeated until bandwidth has been allocated to all flows and considers backbone links and local-area links as

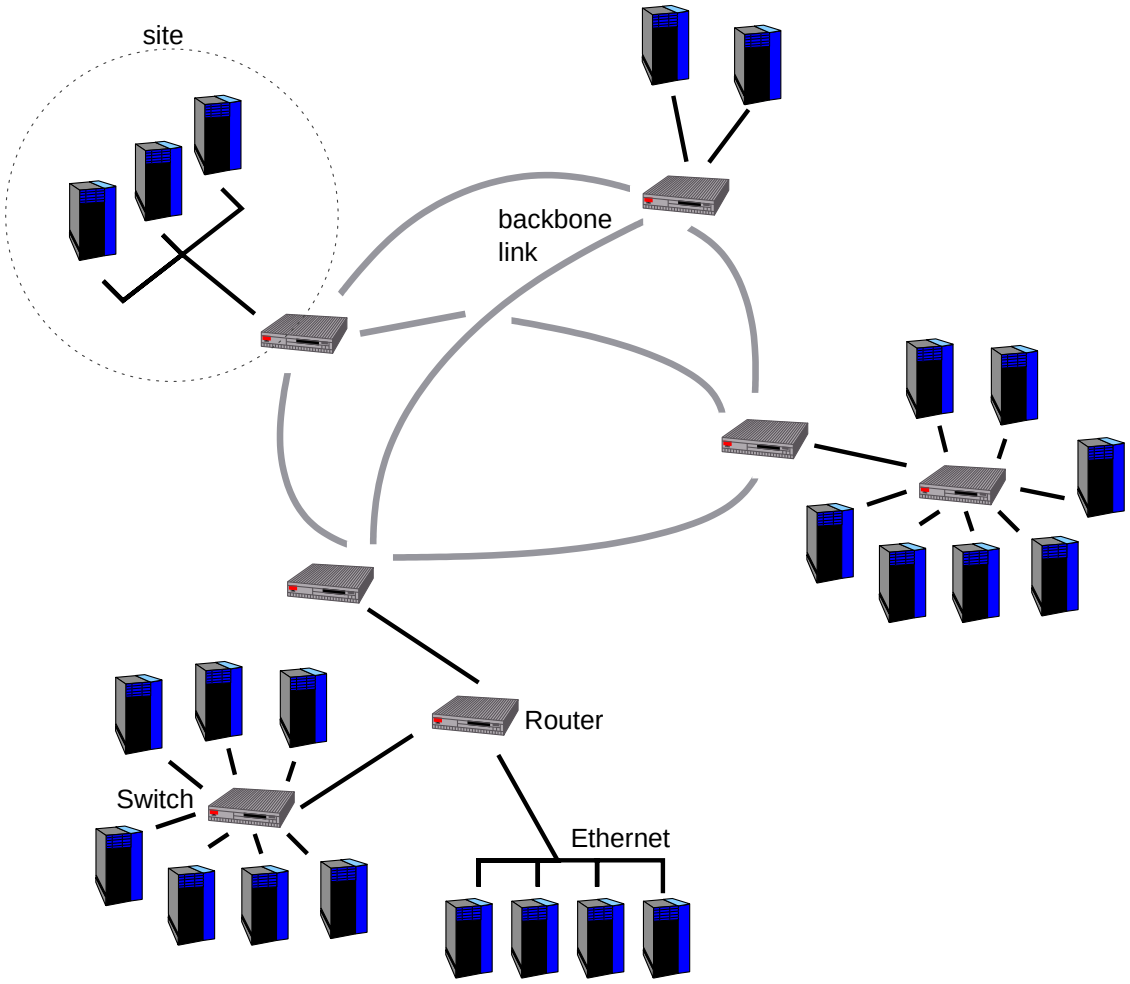

Figure 2. Sample Grid Model

discussed in Section **??**. We have proved the correctness of our algorithm and validated it with the Network Simulator (NS) [28]. The critical point is that the bandwidth sharing model that our algorithm implements is simple enough that it can be used directly for analyzing and developing scheduling strategies that take bandwidth sharing into account (we are currently engaged in doing so for divisible load scheduling problems). SimGrid implements our algorithm and thus makes TCP bandwidth-sharing completely transparent to the SimGrid user. Note that network protocols more evolved than TCP are being investigated for supporting Grid computing applications, for instance ones by which bandwidth sharing can be controlled better at the application level or ones that exploit optical network technology. However, for the time being, TCP is the de-facto standard that most Grid platforms use, and is thus the focu of our model and of SimGrid.

3. Putting it Together

3.1. A Simple Grid model

A key question is to decide which platform model should be instantiated for the purpose of scheduling research. The challenge is to instantiate a model that is simple enough that it can be used for analytical purposes and for fast simulations, and complex enough that it captures the relevant characteristics of real-world platforms. It is almost impossible to precisely quantify this trade-

off, but the discussion in the two previous sections points to the following requirements: network latencies should be modeled, wide-area backbone links should be modeled differently than local-area links, a macroscopic model of TCP bandwidth-sharing should be used. But the question of which network topology to model is still open.

Several Internet topology generators have been developed [17, 25, 31, 22] that create a network consisting of links and router that has properties similar to that of real Internet topologies (power laws, structure, etc.). These generators typically do not model link characteristics and/or traffic and it is the responsibility of the user to annotate the generated topology appropriately, which is difficult. The generated topologies has been traditionally used for simulation, notably with NS [28] in the network research area, and in fact SimGrid makes it possible to import and annotate topologies generated by BRITE [25]. However, there are problems with using such complex topologies for the purpose of designing scheduling algorithms. The most significant one is that since these algorithms are implemented at the application level, only very coarse information about topology and routing is available to them in practice. So while using complex topologies may be appropriate for simulation, more synthetic models that only use information available at the application level are needed for designing new scheduling algorithms.

One possibility is to abstract away the wide-are networking infrastructure as a fully connected network among the "sites" that form the Grid platform, where a site typically corresponds to an institution in a single geographical location. This simplifying model implies that there is no contention between application transfers on the Internet when the source and/or destination sites of these transfers are different. For instance, transfers from UCSD to UVA do no interfere with transfers from UCSD to UW on the Internet, but of course can interfere with each other within the UCSD institution, i.e., before the connection to an Internet backbone. We have seen in Section 2.3 that backbone links can often be modeled as ones on which transfers of the applications do not contend for bandwidth among themselves. If this is the case, then this model implies no interference between any transfer on wide-area links. Within each institution it is reasonable to think that some knowledge of the network topology is directly available or that it can be easily discovered. For instance, the work in [18] describes an extension to the Effective Network View (ENV) software [30] by which effective layer 3 topologies can be discovered with application-level measurements and contention tests (we focus on layer 3 topology information as this is information that can be obtained at the application level). See [20] for an example of the type of topologies that can be discovered with this tool. Note that ENV also annotates the (logical) network links with effective bandwidths. The bandwidth and bandwidth-sharing behavior of wide-area links can be identified with experiments similar to the one used to compute the data in Figure 1. Other layer 3 topology discovery approaches are of course also available and may be used if deployed [9, 29, 14, 26, 16].

Figure 2 depicts a sample Grid model with four sites. Backbone links are shown as grey lines, while local-area links are shown in black. Even with such a seemingly complex topology, it is straightforward to exploit the bandwidth sharing model described in Section 2.3 to develop new scheduling algorithms analytically and we are currently engaged in such efforts. Note that the ENV approach could also be used to try to infer link sharing over the wide area and move beyond the fully connected assumption if needed.

We claim that this topology model together with our bandwidth sharing model strikes a good balance by being simple enough to be amenable to the analysis and development of novel scheduling algorithms and yet significantly more realistic than models traditionally used in the scheduling literature. We have performed component-wise validation of our model (see Section **??** and 2.3) and the SimGrid software makes it straightforward to instantiate model simulations.

3.2. The SimGrid Framework

4. Conclusion

In this paper we have discussed a number of issues pertaining to the modeling of large-scale computational platforms for the specific purpose of scheduling research. In particular we have focused on network models and have highlighted the shortcomings of modeling approaches used traditionally in the parallel computing area when applied to wide-area networks. We have made a case for the modeling of network latencies, outlined new models for bandwidth sharing that better reflect the behavior of TCP on real-world networks, and proposed a streamlined model for Grid computing platforms that is more realistic that used previously and yet amenable to analysis. We have also highlighted those models that we have implemented as part of our own simulation framework for evaluating scheduling algorithms, SimGrid.

References

[1] The SimGrid Project. `http://grail.sdsc.edu/projects/simgrid`.

[2] Bharadwaj, V., Ghose, D., and Mani, V. Multi-Installment Load Distribution in Tree Networks With Delays,. *IEEE Trans. on Aerospace and Electronc Systems 31*, 2 (1995), 555–567.

[3] Casanova, H., and Marchal, L. A Network Model for Simulation of Grid Application. Tech. Rep. 2002-40, Laboratoire d'Informatique du Parallélisme, ENS-Lyon, France, October 2002.

[4] Cheng, Y.-C., and Robertazzi, T. Distributed Computation for a Tree-Network With Communication Delay. *IEEE transactions on aerospace and electronic systems 26*, 3 (1990).

[5] Chiu, D. Some Observations on Fairness of Bandwidth Sharing. Tech. rep., Sun Microsystems, 1999.

[6] Chun, G., Dail, H., Casanova, H., and Snavely, A. Benchmark Probes for Grid Assessment. Tech. Rep. CS2003-0760, Dept. of Computer Science and Engineering, University of California, San Diego, August 2003.

[7] Culler, D., Karp, R., Patterson, D., Sahay, A., Schauser, K., Santos, E., Subramonian, R., and Von Eicken, T. LogP: Towards a Realistic Model of Parallel Computation. In *Proceedings of the ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming* (1993).

[8] DOE Science Grid. `http://www.doesciencegrid.org/`.

[9] Downey, A. Using Pathchar to Estimate Internet Link Characteristics. *Measurement and Modeling of Computer Systems* (1999), 222–223.

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

[11] Foster, I., and Kesselman, C., Eds. *Computational Grids: Blueprint for a New Computing Infrastructure*, 2nd ed. M. Kaufman Publishers, Inc., 2003.

[12] Foster, I., Kesselman, C., Nick, J., and Tuecke, S. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. Open Grid Service Infrastructure WG document, Global Grid Forum, June 2002.

[13] Foster, I., Kesselman, C., and Tuecke, S. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International Journal of High Performance Computing Applications 15*, 3 (2001).

[14] Francis, P., Jamin, S., Jin, C., Jin, Y., Raz, D., Shavitt, Y., and Zhang, L.

IDMaps: A Global Internet Host Distance Estimation Service. *IEEE/ACM Transactions on Networking* (October 2001).

[15] The Globus Toolkit. `http://www.globus.org`.

[16] Govindan, R., and Tangmunarunkit, H. Heuristics for Internet Map Discovery. In *Proceedings of INFOCOM* (March 2000), pp. 1371–1380.

[17] GT-ITM: Georgia Tech Internetwork Topology Models. `http://www.cc.gatech.edu/projects/gtitm/`.

[18] Legrand, A., and Lerouge, J. MetaSimGrid : Towards realistic scheduling simulation of distributed applications. Tech. Rep. 2002-28, Laboratoire d'Informatique du Parallélisme, ENS-Lyon, France, July 2002.

[19] Legrand, A., Marchal, L., and Casanova, H. 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).

[20] Legrand, A., Marchal, L., and Casanova, H. 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).

[21] Li, K. Parallel Processing of Divisible Loads on Partitionable Static Interconnection Networks. *Cluster Computing 6*, 1 (2003), 47–55.

[22] Lu, D., and Dinda, P. GridG: Generating Realistic Computational Grids. *Performance Evaluation Review 30*, 4 (March 2003).

[23] Massoulié, L., and Roberts, J. Bandwidth sharing: Objectives and algorithms. In *Proceedins of INFOCOM* (1999), pp. 1395–1403.

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

[25] Medina, A., Lakhina, A., Matta, I., and Byers, J. BRITE: Universal Topology Generation from a User's Perspective. Tech. Rep. 2001-003, Computer Science Department, Boston University, 1 2001.

[26] Ng, T. E., and Zhang, H. Predicting Internet Network Distance with Coordinates-based Approaches. In *Proceedings of IEEE INFOCOM'02, New York, NY* (June 2002).

[27] The NPACI Grid. `http://npacigrid.npaci.edu/`.

[28] The network simulator - ns-2. `http://www.isi.edu/nsnam/ns`.

[29] Seshan, S., Stemm, M., and Katz, R. SPAND: Shared Passive Network Performance Discovery. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems* (1997).

[30] Shao, G., Berman, F., and Wolski, R. Using effective network views to promote distributed application performance. In *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications* (June 1999).

[31] Sugih, J. Inet-3.0: Internet Topology Generator. Tech. Rep. 456-02, Computer Science Department, University of Michigan, 2002.

[32] TeraGrid. `http://www.teragrid.org/`.

[33] Yang, Y., and Casanova, H. Extensions to the Multi-Installment Algorithm: Affine Costs and Output Data Transfers. Tech. Rep. CS2003-0754, Dept. of Computer Science and Engineering, University of California, San Diego, July 2003.

[34] Yang, Y., and Casanova, H. RUMR: Robust Scheduling for Divisible Workloads. In *Proceedings of the 12th IEEE Symposium on High Performance and Distributed Computing (HPDC-12)* (June 2003).

[35] Yang, Y., and Casanova, H. UMR: A Multi-Round Algorithm for Scheduling Divisible Workloads. In *Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS'03)* (April 2003).