Laboratoire de l'Informatique du Parallélisme

École Normale Supérieure de Lyon
Unité Mixte de Recherche CNRS-INRIA-ENS LYON
n° 5668

A Network Model for Simulation of Grid Application

Henri Casanova,
Loris Marchal

October 2002

Research Report N° 2002-40

École Normale Supérieure de Lyon
46 Allée d'Italie, 69364 Lyon Cedex 07, France
Téléphone : +33(0)4.72.72.80.37
Télécopieur : +33(0)4.72.72.80.80
Adresse électronique : `lip@ens-lyon.fr`

A Network Model for Simulation of Grid Application

Henri Casanova, Loris Marchal

October 2002

Abstract

In this work we investigate network models that can be potentially employed in the simulation of scheduling algorithms for distributed computing applications. We seek to develop a model of TCP communication which is both high-level and realistic. Previous research works show that accurate and global modeling of wide-area networks, such as the Internet, faces a number of challenging issues. However, some global models of fairness and bandwidth-sharing exist, and can be link with the behavior of TCP. Using both previous results and simulation (with NS), we attempt to understand the macroscopic behavior of TCP communications. We then propose a global model of the network for the Grid platform. We perform partial validation of this model in simulation. The model leads to an algorithm for computing bandwidth-sharing. This algorithm can then be implemented as part of Grid application simulations. We provide such an implementation for the SimGrid simulation toolkit.

Keywords: grid application, simulation, TCP modeling, bandwith-sharing, fairness, SimGrid

Résumé

Nous nous intéressons ici aux modèles de réseaux disponibles pour la simulation d'algorithmes d'ordonnancement dans le cadre du calcul distribué. Nous cherchons une modélisation du protocole TCP qui soit à la fois globale (approche de haut niveau) et réaliste. De précedents travaux montrent la simulation présise et globale un réseau à échelle mondiale (tel que Internet) se heurte à de nombreuses difficultés, mais qu'on peut rapprocher TCP de plusieurs modèles globaux de partage équitable de bande passante. En utilisant la simulation de TCP (avec NS) et à l'aide des études précedentes, nous essayons d'apréhender le comportement macroscopique de TCP sur une topologie locale. Nous en déduisons ensuite un modèle global pour la plateforme Grid. Nous validons ce modèle partiellement en simulation. Le modèle conduit à un algorithme de partage de bande passante qui peut etre implementé dans les simulateurs d'applications distribuées, ce que nous avons fait pour SimGrid.

Mots-clés: applications disribuées, simulation, modélisation réseau, partage de bande passante, SimGrid

Introduction

Advances in hardware and software technologies have made it possible to deploy parallel applications over increasingly large set of distributed resources. In particular, due to improvements in interconnected technologies, it has become practical to distribute applications on increasingly widely distributed sets of resources: the computation which used to be confined to a single system can now be distributed over what has been termed as the *Computational Grid* [9, 10].

Consequently, new research works have focused on developing scheduling algorithms for Grid applications. These algorithms have the following objective: finding the best assignment of a set of *tasks* onto a set of *resources*, with respect to a given metric. For instance, a typical goal is to minimize application makespan, that is the time between the beginning of the first task and the end of the last task. In this context, a *task* can either be a computation or a transfer, and a *resource* is either a CPU or a network path. For instance tasks can have *dependencies* and applications are thus often described as Direct Acyclic Graph (DAG). Scheduling a DAG on a set of distributed resources is known to be NP-hard. In fact, most "relevant" scheduling problems are NP-hard. As a result, most scheduling algorithms employ *heuristics* to approximate the optimal schedule (e.g. see [12, 3]). These heuristics have low complexity and can be used in practice. However, it is almost impossible to obtain any analytical result concerning the generated schedules. The comparison between different heuristics must then be made via experiments either on a real platform or in simulation.

There are many advantages to use simulation rather than tests in real environments. First the time needed by experiments on a real environment prevents us from conducting a significant number of experiments. Second, using real resources makes it difficult to explore a wide variety of resource configurations. Third, simulations allow for repeatability which is almost never possible in real distributed computing environments. These considerations motivated the establishment of the SimGrid project. SimGrid provides a toolkit for the simulation of Grid applications in a view to evaluate scheduling algorithms.

A critical component of a simulator for Grid computing is the *network model*. Currently SimGrid only provides a simple network model which is limited in that it does not allow for the realistic simulation of complex networks such as available in Grid computing platforms. Our objective in this work is to develop, validate, and implement a new SimGrid network model which addresses these limitations.

This report is structured as follows. In Section 1, we introduce SimGrid and describe its current network model. We highlight specifics limitations and thereby motivate our work. In Section 2, we review relevant results available in the networking literature. Based on these results, we develop a global network model for the Grid platform assuming that the network protocol in use is TCP. In Section 3 we present experimental results conducted with the NETWORK SIMULATOR [15]. These results are key to both the instantiation and the validation of our new network model. In particular, we focus of *bandwidth-sharing* properties of the network. Section 4 presents an efficient algorithm which implements the appropriate bandwidth-sharing properties as dictated by our model. We also describe how this algorithm is integrated with the SIMGRID software. Finally, Section 5 concludes this report with a summary of accomplishments and perspectives for future work.

1 SimGrid

SIMGRID is a toolkit for simulation of application scheduling on the Grid environment ([5]). It is useful to study the behavior of a scheduling algorithm without writing its own simulator. It provides all the tools necessary to schedule tasks on hosts (for computational tasks) or links (for transfers), set dependencies between tasks, and simulate the application.

1.1 Extant Network Model

SIMGRID models networks as ensemble of *network links*. Each network link is characterized by a *Latency* (in seconds) and a *Bandwidth* (in bytes per second). Assuming a single transfer of S bytes, the duration of that transfer on a network link, T, is:

$$T = \text{Latency} + \frac{S}{\text{Bandwidth}}$$

Latency and bandwidth values for each link can be constant, or time-varying according to *traces*. Traces are either generated artificially or measured on real networks (e.g. NWS [19, 16] traces).

SIMGRID provides three modes for multiple transfers to co-exist a network link: *in order*, *out of order*, and *time sliced*. The first two modes are sequential, i.e. each transfer uses the link exclusively from beginning to end. In the first mode, transfers occur in the order they were scheduled on the link (FIFO). In the second mode, a transfer can start execution if it is ready and if all transfers previously scheduled before it are not ready (e.g. due to dependencies on tasks that have not yet completed). The third mode allows transfers to share the bandwidth of the link: bandwidth is allocated fairly among all the transfers. In other words, if N transfers are executing on a link of bandwidth $Bandwidth$, each transfer experiences a data transmission rate equal to $Bandwidth/N$.

1.2 Limitations of the Model

The network model currently used in SimGrid is straightforward and has the advantage of requiring little computation, enable fast simulations. However, it has several shortcomings both in terms of usability and of realism, which we review in this section.

In its current incarnation, SimGrid does not provide an abstraction for *routing*. In other words, there is no built-in features to specify that a transfer need to follow a sequence of network links. Grid applications usually utilize ensembles of resources that span several sub-networks (e.g. institutions), all sub-networks being connected via wide-area networks. I such a setting, most data transfers occur over multiple network links. Therefore, it is necessary to simulate the routing of transfers for increased realism.

In the current SimGrid implementation, routing can be implemented at the user level as a chain of dependent transfers that are each scheduled on a network link in a route. However, this places a heavy burden on the SimGrid user. Furthermore, it implements a *store-and-forward* policy as each transfer in the chain can start only when the previous transfer as completed. Clearly this is extremely unrealistic when compared to real routers and networks.

One approach could be for the user to segment each transfer into *packets* (thereby creating many individual but inter dependent transfers) in order to allow pipelining of communications. This places even more burden on the user, and requires dramatically more computations and memory, rendering the simulation time prohibitive. Although this could be done within SimGrid, it is rather contrary to the project's philosophy: the goal is to keep simulations fast and simple for the purpose of simulation Grid computing applications. If packet-level simulation is required, then tools like the Network Simulator (NS) [15] should be used.

Another limitation of the SimGrid network model is that it provides only naive bandwidth-sharing capabilities. As described in Section 1.1, SimGrid can implement fair sharing of bandwidth among transfers on a link. This can be justified for a single local-area link, but is known to be unrealistic when multiple links are used by multiple transfers. A reason is that a transfer that uses multiple links achieves an overall transfer rate equal to that on its *bottlenecks link*. The current SimGrid implementation does not make it possible to simulate such behaviors.

In summary, simulating the routing of transfers over multiple links is necessary for realistic Grid application simulation. It requires realistic simulation of a transfer occurring on multiple links, and simulation of non-trivial bandwidth sharing behavior among different transfers. The current SimGrid implementation falls short in that it does not provide appropriate ways to specify transfer routing, does not model multiple-link transfers adequately, and implements only simplistic and unrealistic bandwidth-sharing behaviors.

1.3 Objectives

The objective of this work is to address the limitations identified in Section 1.2. We achieve this by developing a radically new SimGrid network model and by extending the SimGridimplementation and API. More specifically, this work provides the following capabilities:

1. Addition of an abstraction for routing of transfers on multiple links;
2. Modeling of the TCP protocol (as it is the most commonly used for transfer of data in Grid applications);
3. Implementation of realistic transfer behavior of an individual transfer across multiple links (e.g. taking into account bottleneck links);
4. Implementation of realistic bandwidth-sharing properties among transfers that compete for network links;
5. Low computation cost: our goal is to keep the same ratio of simulation time to simulated time as that of the current SimGrid software.

In order to achieve these goals, we are seeking a model that allow us to compute the bandwidth usage of all simulated transfers over arbitrary network topology at the *global* level, that is without simulating the individual behavior of each transfer and link in the system. The only information that we can use are: (i) the topology of the network; (ii) the characteristics of each link (i.e. bandwidth and latency); (iii) the set of pending transfers and for each transfers the amount of data and the route to be used.

2 Modeling TCP Networks

We focus on the TCP protocol because it is most commonly used for communication in current Grid applications. Works in the networking literature that try to understand and model the behavior of TCP fall into two categories:

1. A number of efforts propose models for the global (macroscopic) shape of TCP traffic and load on the Internet, and try to understand how that traffic is impacted by Internet applications (e.g. Web traffic);
2. At a lower level, many authors have proposed models for the behavior of TCP connections (microscopic behavior). These models typically use many variables (e.g. packet loss, congestion window size), are very sophisticated. They make it possible to make a connection between TCP behavior and theoretical models of resource allocation *fairness*.

Both approaches provide us with precious insights that we use for developing a simple and high-level model of TCP connections for the simulation of Grid applications.

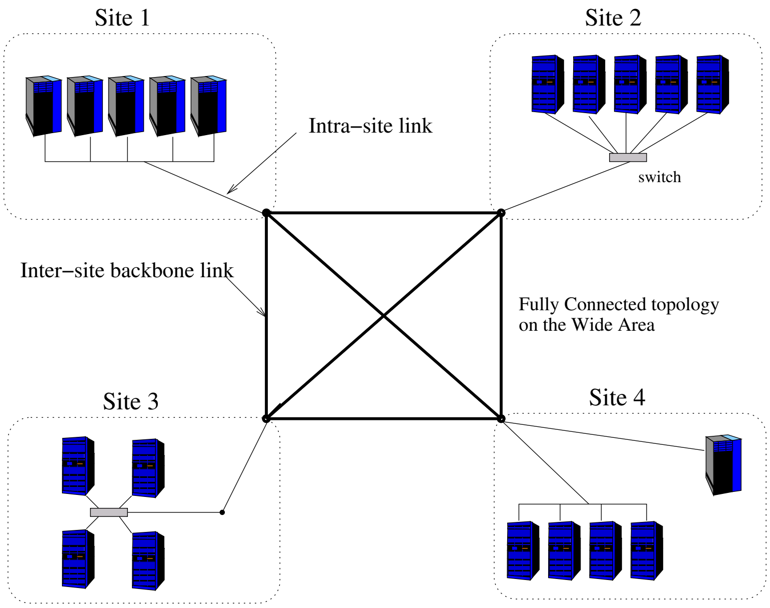

Figure 1: Typical model for a Grid topology.

2.1 Network Topology and Heterogeneity

Simulating Internet traffic is known to be an extremely challenging problem that will probably never be entirely be solved [8]. First of all, Internet topology is very difficult to study because (i) we lack precise information about it, and (ii) it is in constant evolution. Indeed, Internet traffic grows exponentially, and the Internet is engineered by competing entities who are often unwilling to provide topology information. Significant advances have been made in developing topologies for Internet simulation [4], but they do not provide a full understanding of the global Internet behavior.

Other difficult issues are multi-pathing and asymmetric routing: different packets of a same TCP connection can take different paths to reach a same destination, and route from A to B is not necessary the same as route from B to A. Therefore, acknowledgement packets may not take the same route as the data packets.

Also, many different protocols are used in the Internet, including several different versions of TCP. The Reno version is the most common. Newest versions, such as New-Reno and SACK, may be used, and older versions (Tahoe, Vegas) still exist. A model should not depend on the idiosyncrasies of a particular TCP version.

Given these difficulties, and our goal to provide fast, high-level simulation for the purpose of studying Grid applications, we focus on a simple model of the Internet. Grid computing platforms consist of many network links which are heterogeneous in terms of physical characteristics and load. We claim that a reasonable model of a Grid computing platform is one that consists of several *sites* that are interconnected over the Internet, as depicted in Figure 1 for 4 sites. We ignore features of the Internet topology and we assume that sites are connected over the wide-area in a fully-connected topology. We argue that this is a reasonable assumption for the time being given the state-of-the-art of Internet modeling, and the specific goals of the SimGrid project.

As a result, we model only two types of network links:

1. Local-area intra-site links sites. The only load on those links comes from transfers simulated by SimGrid. Bandwidth is shared among all these transfers. Note that on those links, "background" load must be explicitly generated by the SimGrid user.

2. Internet backbone inter-site links, with large physical bandwidth and thousands of pending transfers. On these links, the transfers of a Grid application are only few of many other transfers. Therefore, there is no sharing of bandwidth among the transfers we simulate: each SimGrid transfer gets the same (small) fraction of the overall bandwidth. Background load can be simulated by using bandwidth traces such as the ones provided by NWS [19].

2.2 Modeling Bandwidth-Sharing

As we mentioned in Section 1.3, one of the goals of this work is to implement more realistic bandwidth-sharing than that currently available in SimGrid. This is important in order to be able to model multiple flows sharing local-area links (recall that we assume no bandwidth-sharing among SimGrid transfers on Internet backbone links).

A number of recent efforts in network research tackle the problem of finding theoretical models of communication protocols, and in particular to understand the effect of bandwidth-sharing on global network performance [18, 13, 2]. Most works use a network model that makes an analogies with fluids in pipes: they assume that transfers, or *flows*, are perfectly fluid and they ignore the granularity which is due to packet size. Also, they only consider *steady-state* behavior: the time-scale of the *flow level* is supposed to be much longer than the time-scale of the *packet level*. Therefore, the time of convergence to the steady state can be neglected. This is a valid assumptions for large data transfers but is questionable for short-lived connections (due to the TCP slow-start for instance). Given that we target the simulation of Grid applications, we assume that data transfers are relatively large (e.g. due to scientific data) and that it is reasonable to assume steady-state to compute bandwidth-sharing in SimGrid.

We consider a network consisting of a set of links $\mathcal{L}$, where each link $l \in \mathcal{L}$ has a capacity $C_l > 0$. The flows that compete for access to these links use *routes*. A route is a sequence of links defined as a subset of $\mathcal{L}$. In the discussion of our model, we use the terms *route* and *flow* interchangeably. We assume that there is one flow for each route. Therefore, we we say "the bandwidth allocated to route r" it means "the bandwidth allocated to the flow going through route r". Note that there can be several identical routes in the network, meaning that several flows go over the same sequence of links. When route r goes through link l, we have $l \in r$, and $\mathcal{R}$ denote the set of routes. Let λ_r denote the data transfer rate for route r due to link capacity allocation. A *feasible* bandwidth allocation must satisfy the following constraints:

$$\forall l \in \mathcal{L}, \quad \sum_{r \ni l} \lambda_r \leq C_l,$$

which states that links cannot deliver more bandwidth than their capacities.

We now discuss three well-known models for bandwidth-sharing. We illustrate them on the classical example of a linear network, which is depicted in Figure 2.

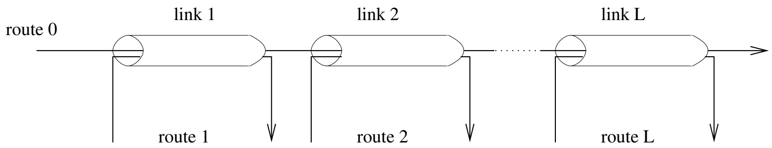

Figure 2: Linear network with identical links of capacity C.

MaxMin Fairness This is the traditional bandwidth-sharing principle, discussed for example in [1]. The objective is to maximize the minimum of $\{\lambda_r\}$, or more specifically: to find a feasible allocation such that an increase of any λ_r within the domain of feasible allocations must be at the cost of a decrease of some $\lambda_{r'}$ such that $\lambda_{r'} < \lambda_r$. This leads to the following formula:

$$\forall r \in \mathcal{R}, \quad \exists\, l \in r, \qquad \sum_{r' \ni l} \lambda_{r'} = C_l \text{ and } \lambda_r = \max\{\lambda_{r'}, r' \ni l\}$$

In the linear network of Figure 2, each flow is assigned half of the capacity C under MaxMin fairness.

Proportional Fairness This notion has been introduced by Kelly [11]. He initially questioned the appropriateness of MaxMin fairness as it allocates more overall resources to long flows than to short flows. This led to proportional fairness principle as described in [13, 18]. The goal is to find a feasible allocation which maximize $\sum_{\mathcal{R}} \lambda_r log(\lambda_r)$. One can show that the proportionally fair allocation, $(\lambda_r)_{r\in\mathcal{R}}$, is unique and such that:

$$\text{for any other feasible allocation}(\lambda'_r)_{r\in\mathcal{R}}, \quad \sum_{r\in\mathcal{R}} \frac{\lambda'_r - \lambda_r}{\lambda_r} \leq 0.$$

In the linear network of Figure 2, proportional fairness leads to the following allocation: $C/(L+1)$ for flow 0, and $C(L-1)/(L+1)$ for all other flows.

Potential Delay Minimization An other idea for sharing bandwidth between flows is to minimize the time delay needed to complete all transfers. The assumption is that flows are used to transfer documents of fixed size. One must then minimize the potential delay $1/\lambda_r$ for each flow. Therefore, the goal is to find an allocation that minimizes $\sum_{\mathcal{R}} 1/\lambda_r$. In the case of Figure 2, this will lead to allocate $C/(1+\sqrt{L})$ to flow 0, and $C\sqrt{L}/(1+\sqrt{L})$ to the other flows.

The question is then: *which bandwidth-sharing principle holds for TCP connections on the Internet?* The consensus is that TCP protocol is "close" to proportional fairness since it favors short flows. However, Chiu shows in [6] that TCP does not implement proportional fairness exactly. Indeed, TCP transfer rate for a flow can be roughly approximated by:

$$x = \frac{c}{RTT\sqrt{q}}, \tag{1}$$

where RTT is the flow's round-trip time, q is the fraction of packets lost, and c is a constant. Assuming that q is approximately the same for all flows, then the transfer rate of a flow going through a bottleneck link is inversely proportional to its round-trip time (RTT). Based on this model, Chiu present a counter example in which the proportional fairness bandwidth allocation and the TCP bandwidth allocation are different.

A number of advances have been made in TCP modeling. For example, [17] presents a a stochastic study of TCP's "low-level" behavior. The work takes into account the congestion avoidance mechanism with acknowledgement behavior, and the packet loss detected either by duplicate acknowledgement or by timeout. The study synthesizes a formula for the transfer rate which depends on the RTT, the number of acknowledged packets, and the fraction of packets lost. In our simulations, we do not know the value of some of these parameters, but if we assume that everything is constant but the RTT, we obtain a transfer rate inversely proportional to the RTT. This property is commonly accepted an appears in other modeling work [7, 14].

2.3 A New Network Model for SimGrid

Based on the above considerations, we make the following choices for designing a new network model for intra-site links in SimGrid. The model considers transfers as a fluid flows because Grid applications usually exchange large amounts of data. Therefore, we can restrict the model to capture steady-state behavior. The model should implement a variation of proportional bandwidth-sharing. The weights for the proportional sharing should be inversely proportional to the round-trip time of flows. In other words, the fractions of a link capacity allocated to flows for which that link is a bottleneck link are inversely proportional to the flows' RTTs. We ignore all other parameters of TPC modeling (e.g. packet loss rate, size of congestion windows). Finally, our model contains intra-site and inter-site links as presented in Section 2.1. In the next section we present experimental results in a view to validating and refining the model.

3 Experiments

In order to validate the network model described in Section 2.3, we performed a number of simulation experiments. We have chosen the NS simulator [15] because it is widely accepted and complete simulator (e.g. it implements most versions of the TCP protocol). More specifically, we performed 4 sets of experiments: (i) experiments with a single TCP link; (ii) experiments to study the relationship between link latency and flow RTT; (iii) experiments to validate our bandwidth-sharing principle on a generalized "dogbone" topology with many flows on links of various capacities and latencies; (iv) experiments to validate our inter-site backbone link model.

3.1 Single TCP Link

Before we can study the behavior of TCP on a network with different links and load, we must understand its behavior on a single link with a single flow. More specifically, we wish to discover relationships between the physical characteristics of a link and the characteristics that are experienced and measured at the application level (e.g. by NWS [19]). The former are declared in NS. The latter are to be used in SimGridas they are known to the user who instantiate a simulation. We denote the physical latency and bandwidth of the link as Lat_{NS} and BW_{NS}, and the measured ones as Lat_{exp} and BW_{exp}. Considering a transfer of size S, we express the transfer time, T, with the traditional model:

$$T = \mathrm{Lat}_{exp} + \frac{S}{\mathrm{BW}_{exp}}.$$

A first question is: are the experimental values for latency and bandwidth identical to their physical counterparts? Figures 3 and 4 show the experimental values as functions of the physical parameters.

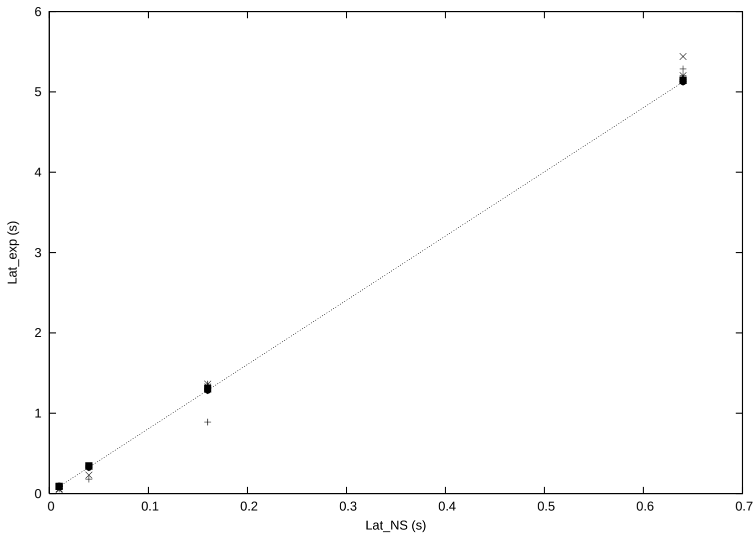

Figure 3: Experimental latency as a function of the physical latency.

The experimental latency is well approximated by a linear function of the physical latency as:

$$\mathrm{Lat}_{exp} \quad = \quad \alpha \times \mathrm{Lat}_{NS}, \tag{2}$$

where α can be determined from the simulation data.

By contrast, the experimental bandwidth is not a simple function of the physical bandwidth. In fact, it depends on the physical latency, as seen in Figure 4. In the figure, the circles represent the experimental data, and the surface plots the following function:

$$\mathrm{BW}_{exp} \quad = \quad \min(\lambda \times \mathrm{BW}_{NS}, \frac{\gamma}{\mathrm{Lat}_{NS}}) \tag{3}$$

where λ and γ can be determined from the simulation data.
The above function fits the experimental data point extremely well (for extended results, see Appendix A). This suggests that, even in a non-congested context, the bandwidth is limited by a quantity that is inversely proportional to the latency. One can then find an equation for experimental values:

$$\mathrm{BW}_{exp} = \min(\mathrm{BW}_{max}, \frac{\gamma \times \alpha}{\mathrm{Lat}_{exp}}).$$

Where $\mathrm{BW}_{max} = \lambda \times \mathrm{BW}_{NS}$ is the maximum bandwidth achievable on the link by a TCP connection when the transfer rate is not impacted by the

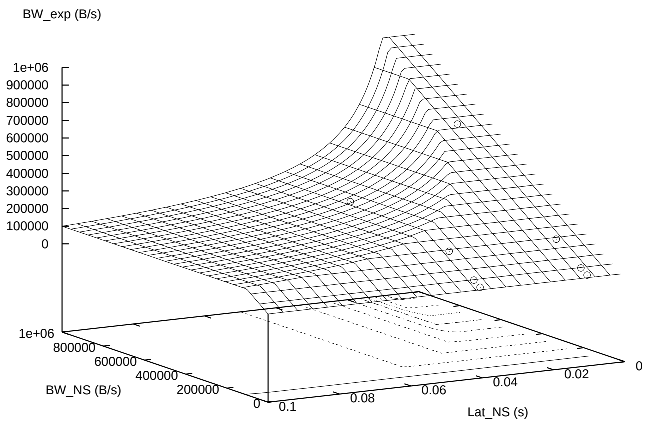

Figure 4: Experimental bandwidth as a function of physical latency and bandwidth values.

latency. In practice, for a local-area link, this value could be computed as the maximum observed data rate on the link (e.g. maximum value ever seen in an NWS trace over time). The upper bound of the bandwidth (γ/RTT) can be found in one of the formulas developed in [17] as W_{max}/RTT, where W_{max} is the higher size of the congestion window. This phenomenon can be explain by the "*round*" mechanism: in the begin of the round, the W packets of the congestion window are sent without any need of acknowledge packet. Then no other packets can be sent until the first acknowledgement is received for one of these packets. This acknowledgement reception marks the beginning of a next round. In this model, the duration of a round is assumed to be the round-trip time, and we assume this RTT does not depend of the window size. This hypothesis is usually adopted [17] and seems justified in the case of long RTTs, which is the case where we can notice the effect of this upper bound.

3.2 Latency and Round-Trip Time

Round-Trip Time (RTT) is a fundamental component of most models of TCP transfers. The RTT is defined as the time between sending a packet and receiving its acknowledgement. Intuitively, the RTT should be two times the latency of the link, for a route composed of a single link, and two times the sum of all link latencies for a route going through several links. The NS simulator allow us to measure the RTT of a transfer, and we plot simulation results in Figure 5. Both graphs plot the ratio of RTT to physical latency as a function of link bandwidth. The leftmost graph corresponds to a single network link, whereas the rightmost one corresponds to a chain of several (up to 12) network links. In both cases, the RTT/latency ratio is constant for physical bandwidth values over 200KB/s. The data leads to the following model for RTT_t, the RTT for route r:

$$\mathrm{RTT}_r = \beta \times 2 \times \sum_{l \in r} \mathrm{Lat}_l$$

where $\beta = 0.1$ is a multiplicative factor which we attribute to NS and have yet to explain. However, this multiplicative factor does not cause any problem in our model. Indeed, our bandwidth sharing principle uses weights that are proportional to the RTT (i.e. the β factor cancels out).

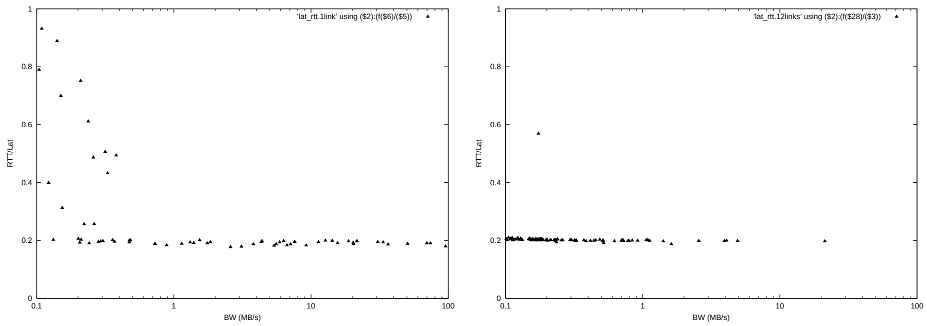

Figure 5: Ratio of RTT over physical latency as a function of the bandwidth in two scenarios: (i) one single link; (ii) a chain of several (up to 12) links.

The leftmost graph in Figure 5 is shows data for 100 experiments with a single link, whose characteristics are randomly chosen between 10ms and 100ms for the latency, and between 0.1MB/s and 100MB/s for the bandwidth. The rightmost graph on the same figure shows data for 100 experiments on a route going through 2 to 12 links (the length is randomly chosen). The link characteristics are sampled similarly to the single link case.

As seen in Figure 5, the equation above fits the NS simulation better for several links (up to 12, rightmost graph) than for one single link (leftmost graph). This may seem surprising at first. However, these simulations are conducted without any load on the links: congestion never happens. For one link, if the latency is very small, effects such as time spent in the network queues are noticeable. In these simulations we sampled link latencies at random. Therefore it is very unlikely that the sum of several links latencies be so small, and network queue delays are almost never significant in the second experiment.

We now consider a scenario in which there is congestion. In this case, the RTT can be much higher that two times the sum of latencies, due to the

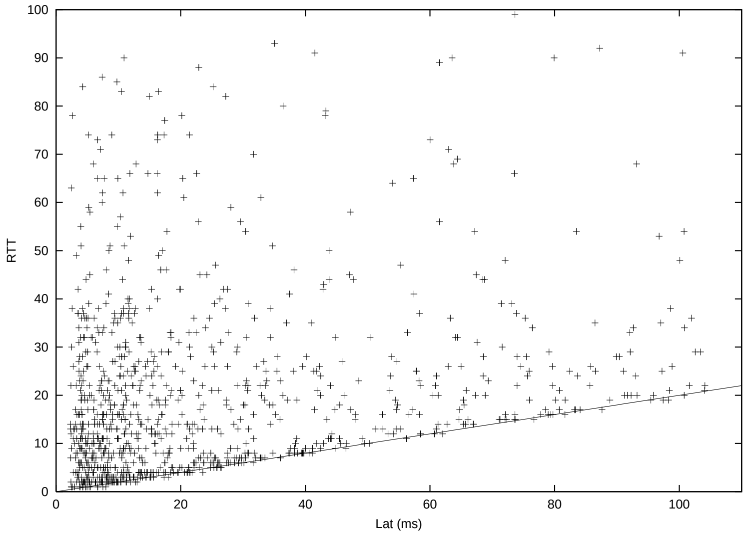

Figure 6: RTT versus latency in a congested network.

time spent in the network queues. For example, Figure 6 shows the RTT versus the latency for the topology depicted in Figure 7, with $n = 5$ routes going through 3 links (where one is a shared link). This figure plots 1000 experimental data points for bandwidth and latency sampled at random between 50KB/s and 50MB/s, and between 1ms and 100ms. The topology will be detailed later in this document, but the important point here is that many flows compete over a single bottleneck link. The solid line represents the previous equation ($y = \beta \times 2 \times x$).

The differences between the model and the experimental RTT/latency ratio can be a problem for the validation of our model. The fundamental question is: should we use latencies or real RTTs when weighting flows for bandwidth-sharing? RTTs would probably give better results than latencies since Eq. 1 uses the RTT. However, the RTT of each route is not readily available to the SimGrid user. The SimGriduser will use two different kinds of data for declaring link "latencies" for the two types of links in the network.

Intra-site links: Latency and bandwidth are fixed values. In that case the model will have to use the sum of the latency as an approximation of the RTT;

Inter-site backbone links: Latency and bandwidth are taken from traces measured on a real. These measures take into account the time spent in the network queues, which approximates the RTT. In this case, the sum of the latencies is a more accurate measure of the RTT.

In both cases, our model will use the sum of the "latencies" declared by the SimGrid user as a more or less accurate approximation of the RTT.

3.3 Bandwidth-Sharing Model

We have seen in Section 3.1 that the bandwidth experienced by a flow on a single link is limited by the RTT. This phenomenon should be observable for multiple-link connections as well. In this section we extend our single link model to a multi-link model with many flows. We have seen in Section 2.2 that the bandwidth allocated to flows competing over a bottleneck link should be inversely proportional to RTTs.

A link is a bottleneck if the sum of the bandwidths allocated to the routes going through this link equals the total bandwidth of the link. We say that the bandwidth BW of a bottleneck is shared between the routes $r_1, \ldots, r_n$ with weights $w_1, \ldots, w_n$ if route r_i is allocated a bandwidth equal to $\text{BW} \times w_i / \sum w_j$. More formal definitions of bottlenecks and sharing are presented in Section 4.1.

3.3.1 Five Candidate Models

Given this definition, we test 5 different models of bandwidth-sharing. In all models we assume that the bandwidth allocated to a flow is bounded by that allocated on the flow's bottleneck link. Each model uses a different way to weight the allocations of flows that have a *common* bottleneck link.

INV-LAT: Each flow r_i is allocated a share of the bandwidth on the bottleneck link with a weight w_i such that:

$$w_i = 1/\sum_{l \in r_i} \text{Lat}_l.$$

INV-LAT-BOUNDED: Same as INV-LAT, with the additional constraint that the bandwidth allocated to flow r_i is bounded by the value obtained in Section 3.1:

$$\text{BW}_{max\, r_i} = \gamma / \sum_{l \in r_i} \text{Lat}_l.$$

INV-RTT: Same as INV-LAT but $\sum_{l \in r} \text{Lat}_l$ is replaced by RTT_{r_i}.

INV-RTT-BOUNDED: Same as INV-LAT-BOUNDED but $\sum_{l \in r_i} \text{Lat}_l$ is replaced by RTT_{r_i}.

MAX-MIN: This is the traditional MaxMin fairness. All flows get an equal share of the bottleneck: $w_i = 1$.

The first 4 models are plausible due to results available in the networking literature (see Section 2.2) and given the simulation results presented in previous sections. We include MAX-MIN fairness for comparison purposes even previous work indicates that TCP does not achieve MaxMin fairness. The question we want to answer is: Which of INV-LAT, INV-LAT-BOUNDED, INV-RTT, and INV-RTT-BOUNDED is the most accurate model of bandwidth-sharing for TCP?

3.3.2 Validation on a Generalized "Dogbone" Topology

We choose to use a classic test-case network topology for our initial validation experiments: the "Dogbone topology". However, we extend the typical 5-link topology to contains an arbitrary number of links as depicted in Figure 7. This topology is sufficiently simple to be interpreted easily, and yet sufficiently sophisticated to lead to interesting congestion scenarios.

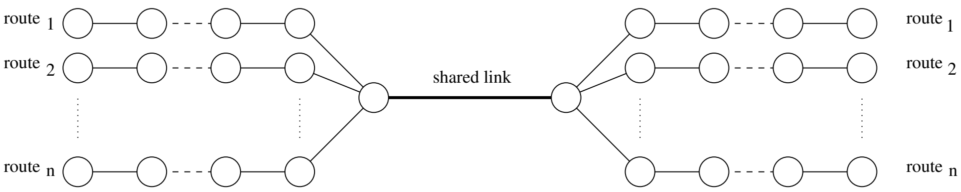

Figure 7: Dogbone topology

We performed 1,000 experiments for 4 instantiations of the topology in Figure 7: (i) 2 routes with 3 links; (ii) 2 routes with 7 links; (iii) 5 routes with 3 links; (iv) 5 routes with 7 links. For each experiment, physical link latencies and bandwidths are sampled between 1ms and 100ms, and 50KB/s and 50MB/s respectively, with log-normal distributions. For each experiment we run an NS simulation and measure the bandwidth allocated to each flow. We also use the 5 bandwidth-sharing models presented in Section 3.3.1 to compute the bandwidth allocated to each flow. For each flow we the error with respect to the bandwidth allocated in the NS simulation as:

$$error = \log_2(bw_{\mathrm{model}}) - \log_2(bw_{\mathrm{NS}}),$$

where bw_{model} is the bandwidth the is allocated to the flow according to each model, and bw_{NS} is the bandwidth allocated to the flow in the NS simulation.

The graphs in Figure 8 show histograms of those errors for all 1,000 experiments, all flows, and 5 models, for the 4 configurations of our generalized dogbone topology. When $|error|$ is close to 0, the error between the modeled bandwidth and the simulated bandwidth is small; when $|error|$ is high, the error is large. For example when $error = 2$, the simulated bandwidth is four times as big as the measured bandwidth. In order to make the histograms readable we use a logarithmic scale on the y-axis, which might hide the fact that the frequency of $error = 0$ (perfect result) is much higher than for other values. The “narrower” the distribution, the better the model.

A summary of those results is presented in Table 3.3.2 and 3.3.2 where we show the mean and variance of the 5 models for all 4 test topologies. We can see that the INV-RTT-BOUNDED model has a mean close to 0, and the lowest variance, for all 4 topologies.

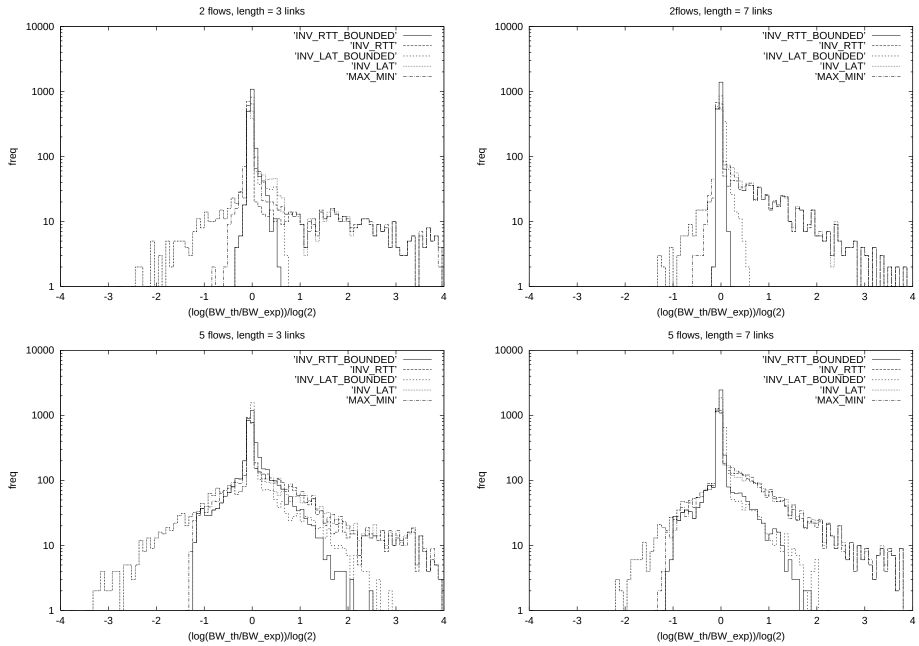

Figure 8: Histograms of model errors of the 5 bandwidth-sharing models for the 4 test-case topologies.

Model	topology #1 (2 flows, 3 links)	topology #2 (2 flows, 7 links)	topology #3 (5 flows, 3 links)	topology #4 (5 flows, 7 links)
INV-RTT-BOUNDED	0.04	0.02	0.09	0.04
INV-RTT	0.53	0.38	0.44	0.37
INV-LAT-BOUNDED	-0.04	0.017	-0.06	0.01
INV-LAT	0.48	0.37	0.32	0.33
MAX-MIN	0.54	0.38	0.48	0.38

Table 1: Mean error of the 5 bandwidth-sharing models.

The main conclusion from these experiments is that the INV-RTT-BOUNDED

Model	topology #1 (2 flows, 3 links)	topology #2 (2 flows, 7 links)	topology #3 (5 flows, 3 links)	topology #4 (5 flows, 7 links)
INV-RTT-BOUNDED	0.09	0.03	0.43	0.29
INV-RTT	1.15	0.76	1.08	0.81
INV-LAT-BOUNDED	0.38	0.16	0.70	0.41
INV-LAT	1.24	0.79	1.26	0.87
MAX-MIN	1.15	0.77	1.13	0.83

Table 2: Variance of the error of the 5 bandwidth-sharing models.

model (solid line) is always the best one in all situations. We show more detailed graphs for more simulations in Appendix B. A number of observations follow from our experimental results:

- The "bounded" models always lead to better results and the "unbounded" ones. This validates what was said in Section 3.1 and proves that introducing the upper bound on bandwidth leads to better accuracy.
- In Figure 17 (see Appendix B) we compare results for the INV-RTT-BOUNDED model obtained in two different ways: (i) the bandwidth is allocated according to the bound γ/RTT; (ii) the bandwidth is shared, in the case when $\sum \text{BW}_{max} > \text{BW}_{link}$ for the middle link. The figure shows that the result for (i) are significantly better than for (ii). It may seem surprising that the bound has such an effect on bandwidth sharing, if we assume that the this bounding occurs only in non-congested context. But congestion leads to higher RTTs, because of time spent by the packets in the queues. Therefore, congestion leads to higher RTTs, thereby leading to lower bandwidth due to the bound (see the "BOUNDED" curve of Figure 17). This is a normal phenomenon of bandwidth-sharing, and shows that adding the γ/RTT bound is crucial for the accuracy of the model.
- In general, MAX-MIN leads to the worst results, confirming the observation made by Chiu in [6]. In fact, we expected MAX-MIN to lead to far worse results. However, in a non-congested network (e.g. when flows are limited by small physical bandwidth of "private" links on the route, not by the middle link), MAX-MIN give the same result as the other models, which is very accurate. For comparison, we plot histograms for MAX-MIN and INV-RTT-BOUNDED in congested networks only in Figures 18 and 19. These plots shows that INV-RTT-BOUNDED gives much better results that MAX-MIN.

Our main conclusion is that INV-RTT-BOUNDED leads to the best results in all situations and it is the model we choose to implement the new SimGrid network model.

3.4 Backbone Simulation

We wish to validate our model of an inter-site backbone link. We view these links as ones with large physical bandwidth and a large load which is external to the simulated Grid application. We claim that flows belonging to a single Grid applications will not share bandwidth among each others. In other words, when there is one application flow, or 10 application flows, all flows are allocated the same amount of bandwidth. To study this phenomenon, we create a NS link with a very large number of connections. The topology used for this experiments is shown in Figure 9 and has a ratio of 20 to 1 between backbone capacity and inter-site link capacity. The results are presented in Figure 10.

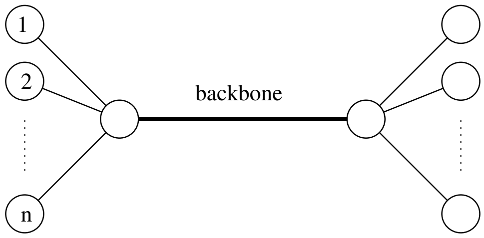

Figure 9: Topology used to study the behavior of a backbone link – The characteristics of the backbone are: BW=20MB/s and Lat=20ms, and those of all other links are: BW=1MB/s and Lat=10ms.

This experiments show that NS is unable to simulate a great number of flows: the experimental results are valid up to 75 flows, but for more 100 flows, the bandwidth allocated to a flow does not seems to depend on the load of the link. For more than 200 flows the bandwidth allocated on the backbone link is higher than the link's physical bandwidth!

Although these results are rather disturbing concerning the validity of NS simulations, researchers in our group have collected experimental data on real wide-are networks which indicate that our backbone model is valid. These results will be presented in an upcoming report.

4 Algorithm and Implementation

We give here a formal description of the bandwidth allocation problem for the INV-RTT-BOUNDED model in the context of SimGrid. A bandwidth allocation $(\lambda_r)_{r\in\mathcal{R}}$ is conform to this model if it satisfies the following rules:

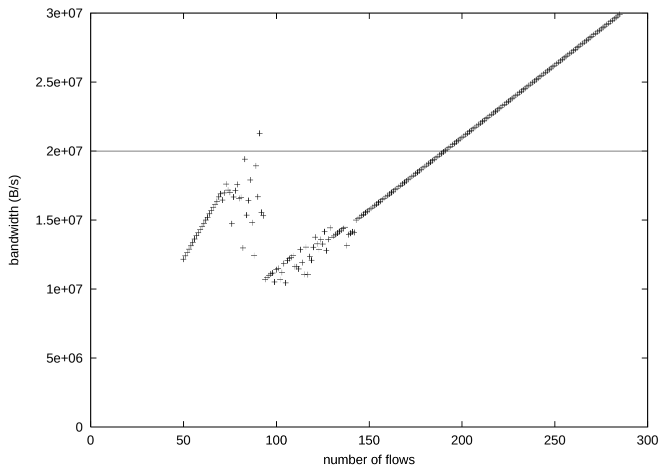

Figure 10: Results of the NS simulation of a backbone link : The x-axis is the number of flows going through the backbone link, and the y-axis is the sum of the bandwidth allocated to each flow. The solid line is the total capacity (bandwidth) of the backbone link.

(i) It is feasible:

$$\forall l \in \mathcal{L}, \quad \sum_{r \ni l} \lambda_r \leq C_l$$

(ii) The bandwidth of each route is bounded:

$$\forall r \in \mathcal{R}, \quad \lambda_r \leq \frac{\alpha \times \gamma}{\mathrm{RTT}}.$$

(iii) On a bottleneck link, the bandwidth is shared with INV-RTT weights. This means that we cannot increase the bandwidth allocated to a route r without decreasing the bandwidth allocated to a route r' such as $\lambda_{r'}\mathrm{RTT}_{r'} < \lambda_r \mathrm{RTT}_r$, or more formally:

$$\forall \lambda' = (\lambda'_r)_{r \in \mathcal{R}} \qquad \big(\exists r \in \mathcal{R}, \quad \lambda'_r > \lambda_r\big)$$
$$\Rightarrow \big(\exists r' \in \mathcal{R}, \quad \lambda'_{r'}\mathrm{RTT}_{r'} < \lambda'_r \mathrm{RTT}_r\big)\,.$$

(iv) Some links can be *backbone* links, meaning that they provide the same bandwidth to all routes without any sharing.

From now on, we say that an allocation is *fair* if it satisfies the 4 rules above.

4.1 An Algorithm to Compute the Bandwidth Allocation

Before developing an algorithm to compute a fair bandwidth allocation, we reduce the problem by removing rules (ii) and (iv). Rule (ii) can be eliminated simply by adding an artificial new link to each route. That link has the following characteristics: $bw = \gamma \times \alpha/\mathrm{RTT}_r$, $lat = 0$. With this new link, rule (i) implies rule (ii). We can also eliminate rule (iv) by creating a copy of each backbone link for each route going through the link. For example if n routes $r_1, \dots, r_n$ go through a backbone link, we replace this link by n links $l_1, \dots, l_n$ with the same characteristics, r_1 going through l_1,..., r_n going through l_n.

We have now reduced the problem to computing a fair bandwidth allocation with no backbone links and without the "bounding" rule. We now give a formal definition of bottleneck links which we will use for developing our algorithm.

Definition 1 (bottleneck) *Link l is a* bottleneck *if* $\sum_{r \ni l} \lambda_r = C_l$

Definition 2 (balanced bottleneck) *A* balanced bottleneck *is a bottleneck, l, for which the bandwidth is shared fairly:*

$$\forall r \ni l,\ \lambda_r = \frac{1/\mathrm{RTT}_r}{\sum_{r' \ni l} 1/\mathrm{RTT}_{r'}} \times C_l$$

In a network where bandwidth is shared fairly, there are both balanced and unbalanced bottlenecks. We show an example in Figure 11. In this network route 1 and route 2 will be allocated a bandwidth equal to 10 and 40 respectively. Therefore link 1 is a balanced bottleneck and link 2 is an unbalanced bottleneck.

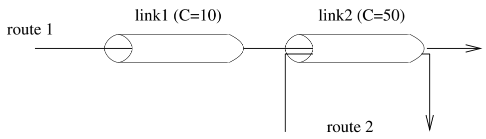

Figure 11: Example network with both balanced and unbalanced bottlenecks.

Proposition 1 (property of bottlenecks on a route) *On a route r, the links which are balanced bottlenecks have the greatest utilization, $U(l)$, where:*

$$U(l) = \frac{1}{C_l} \sum_{r \ni l} \frac{1}{\mathrm{RTT}_r}.$$

Proof Let us consider a route r that consists of links $l_1, \dots l_n$. This route has at least a bottleneck link. Let us assume that this bottleneck is balanced. We can compute the values λ_r^i, which are the bandwidth allocated to r, assuming that link l_i is a balanced bottleneck:

$$\lambda_r^i = \frac{\dfrac{1}{\mathrm{RTT}_r}}{\displaystyle\sum_{r' \ni l_i} \dfrac{1}{\mathrm{RTT}'_r}} \times C_{l_i} = \frac{\dfrac{1}{\mathrm{RTT}_r}}{U(l_i)}.$$

The bandwidth allocated to the route, λ_r, is then equal to $\min_i \lambda_r^i$. The smallest value of λ_r^i is obtained when the value of $U(l_i)$ is the greatest. Therefore, the links with the greatest $U(l_i)$ values are balanced bottlenecks on route r. □

Proposition 2 (balanced bottlenecks) *In a network, the links with the greatest $U(l)$ are balanced bottlenecks.*

Proof Let us consider a link, l, with the greatest $U(l) = M$. Assume that l is not a balanced bottleneck. Then there is at least a route r going through this link which get a bandwidth smaller than λ_r^l, meaning that route r is bandwidth-limited by another link l'. Therefore, we have: $\lambda_r^l > \lambda_r^{l'}$. Hence, $U(l) < U(l')$, which is a contradiction. □

Theorem 1 *The following algorithm computes a fair bandwidth allocation for an arbitrary network.*

While there are routes left:

1. *Compute $U(l)$ for each link,*
2. *Find the links that have the greatest $U(l)$. Note $\mathcal{L}_1$ this set of link.*
3. *For each route r going through at least one of these links, compute the value of the bandwidth allocated to r. Delete all the routes going through links in $\mathcal{L}_1$. Lower the capacity of all links used by these routes. More formally:*

$$\mathcal{R}_1 = \{r \in \mathcal{R} | r \cap \mathcal{L}_1 \neq \emptyset\}$$

$$\forall r \in \mathcal{R}_1, \quad \lambda_r = \min_{l \in \mathcal{L}_1 \cap r} \left\{ \frac{1/\mathrm{RTT}_r}{\displaystyle\sum_{r' \ni l} 1/\mathrm{RTT}'_r} C_l \right\}$$

The new link capacities are then computed as:

$$\forall l \in \mathcal{L} \quad C_l{}' = C_l - \sum_{r \ni l} \lambda_r$$

(so $\forall l \in \mathcal{L}_1$, $C_l{}' = 0$)

Proof According to Proposition 2, at each step, we consider only the links that are balanced bottlenecks. Therefore, the bandwidth allocation that we compute for these links as balanced bottlenecks complies with rules (i) and (iii). The number of routes decreases at least by one at each step, meaning that the algorithm terminates. □

4.2 Implementation in SimGrid

We implemented the algorithm in the previous section in the SimGrid library. The bandwidth allocation is recomputed when transfers start or finish, and each time link characteristics change (e.g. due to the use of NWS traces on backbone links). The SimGrid API has been extend with two new functions:

- `SG_newTCPLink()`,
- `SG_newRoute()`.

The first function is used to declare network links that are used with the TCP protocol. Those links are characterized by a latency and a bandwidth (e.g. obtained from NWS traces). Also, each link can be specified as "backbone" or "not backbone". `SG_newRoute()` creates a route that consist of a sequence of TCP links. Transfers can be scheduled on TCP routes in a way similar to what was available in the previous version of SimGrid.

Note that the notion of "route" defined by this API is different from that used in the previous sections. So far we have identified the notions of route and flow as it made the development of our model and algorithm clearer. However, our SimGrid API decouples the notion of transfer and the notion of route. This means that several transfers can use a route concurrently, which is compatible with the traditional SimGrid API. This difference is purely technical, and leads to only a few straightforward modifications for the implementation of the bandwidth allocation.

Below is an example using the extant SimGrid API:

```
link1 = SG_newLink("link1", SG_TIME_SLICED, LAT_TRACE, OFFSET, 0.0,
                   BW_TRACE, OFFSET, 0.0, NULL);
SG_scheduleTaskOnResource(transfer1, link1);
```

Here is an example with the new API:

```
links[0] = SG_newTCPLink("link0", LAT_TRACE0, OFFSET, 0.0,
                         BW_TRACE0, OFFSET, 0.0, NON_FAT_PIPE, NULL);
links[1] = SG_newTCPLink("link1", LAT_TRACE1, OFFSET, 0.0,
                         BW_TRACE1, OFFSET, 0.0, FAT_PIPE, NULL);
links[2]=NULL;
route1 = SG_newRoute("route1", links, NULL);
SG_scheduleTaskOnResource(transfer1, route1);
```

5 Conclusion

In this work, we have investigated network modeling issues for Grid computing environments. More specifically, this study focused on developing a network model which can be used in the SimGrid simulator. Note that our model is useful beyond the scope of SimGrid and should be relevant to several other Grid application simulators. We have seen that the previous SimGrid network model leads to low computational costs, but suffers from its lack of realism. Therefore, our goal was to develop a more realistic network model while preserving low computational complexity.

We have reviewed previous works in the area of network modeling and simulation (see Section 2): We have (i) highlighted the challenges for simulating large-scale networks like the Internet; (ii) examined models for *fairness* in bandwidth-sharing among network transfers; and (iii) gained insight into the macroscopic behavior of the TCP protocol. Based on these observations, we have been able to develop a network model for the Grid computing platform. This model views the Grid as a number of sites with two different types of network links: inter-site and intra-site. We used NS simulations to instantiate key parameters of our model. Once the model was instantiated, we have then performed validation experiments for intra-site networks.

We have formalized our model in Section 4. We have also developed an algorithm that computes link bandwidth allocations according to the model. This algorithm has been integrated to the SimGrid code base, thereby providing more realistic simulation capabilities to the SimGrid user.

Some experiments are currently being conducted on real networks. It will be interesting to confront these results with ones obtained with our simulator. In particular, we expect that our model for inter-site network links could be extended for better capturing the behavior of wide-area Internet connections.

Appendix

A Detailed results of the simulation for on single link

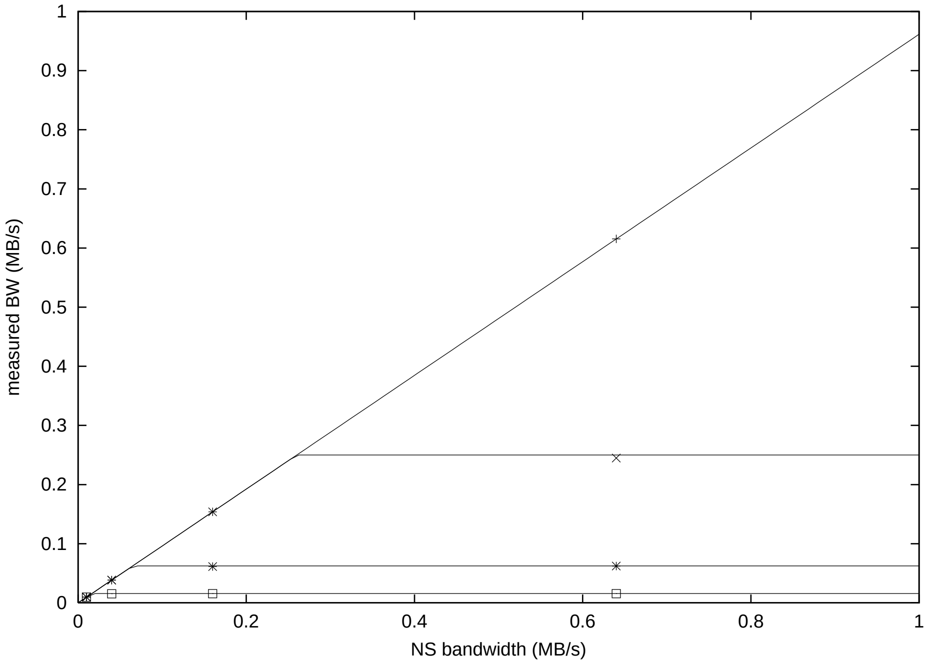

Figure 12: Measured bandwidth versus NS bandwidth – Each curve corresponds to a different link latencies: 10ms, 40ms, 160ms, 640ms.

Figure 12 is the projection of Figure 4 on the plane (NS bandwidth, measured bandwidth). It shows that the Eq. 3 is very accurate: the curves fit the experimental data almost perfectly.

B Detailed graphs of the simulations

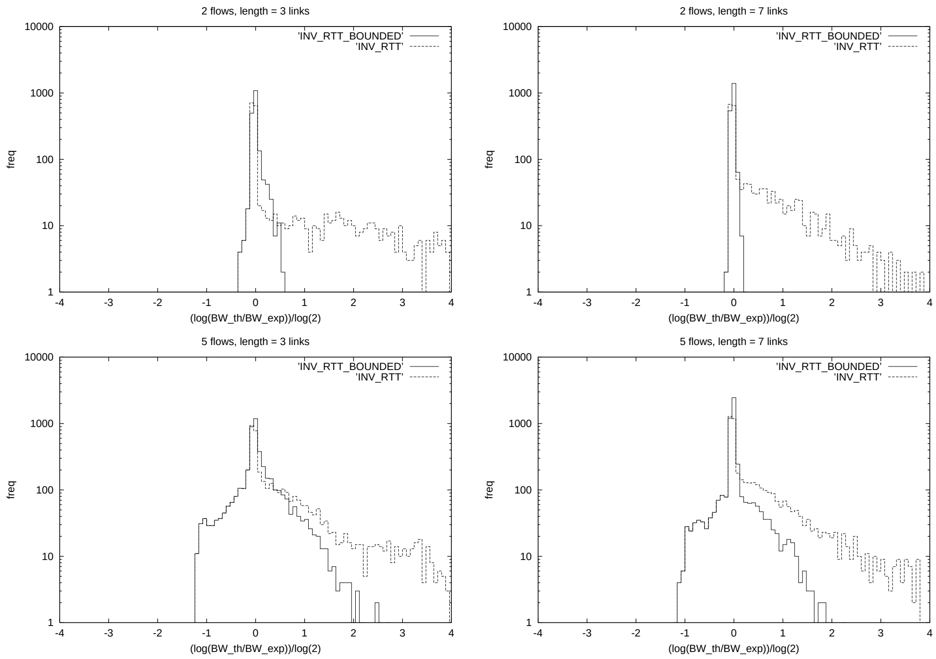

Figure 13: Comparison between bounded and non-bounded INV-RTT models.

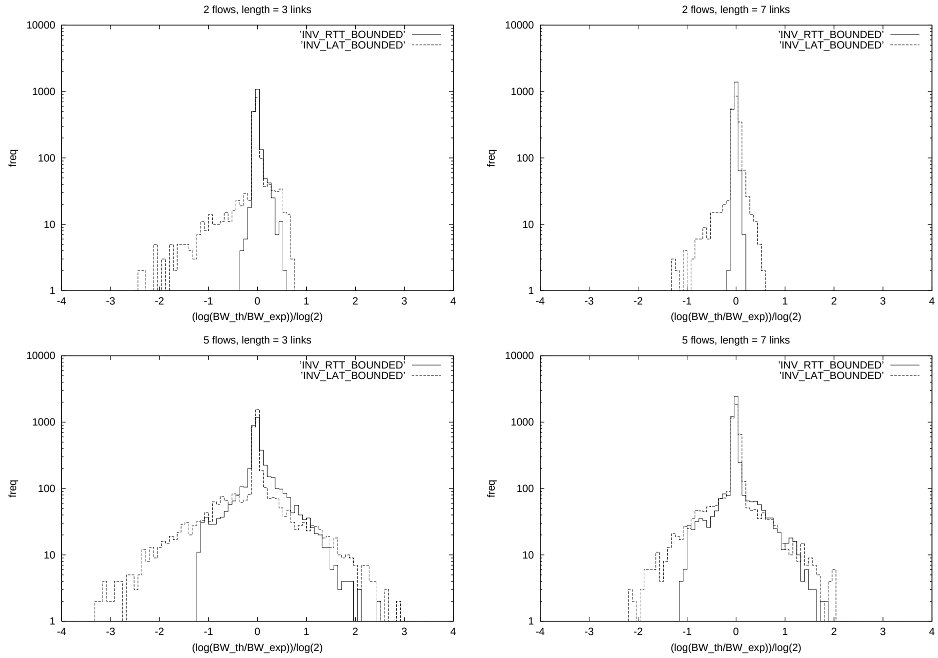

Figure 14: Comparison between using latencies or round-trip time.

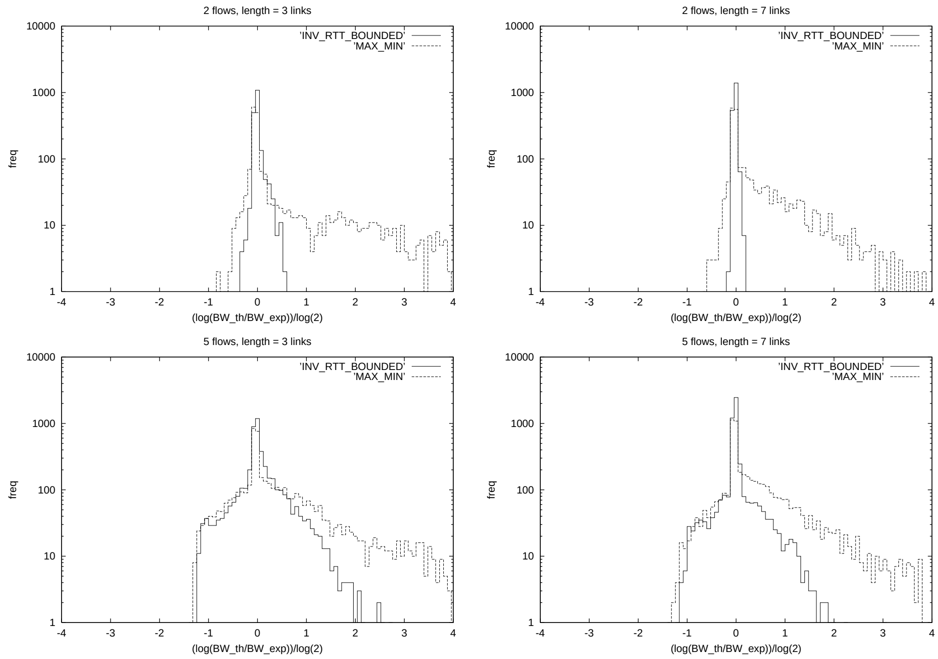

Figure 15: Comparison between INV-RTT model and MaxMin Fairness.

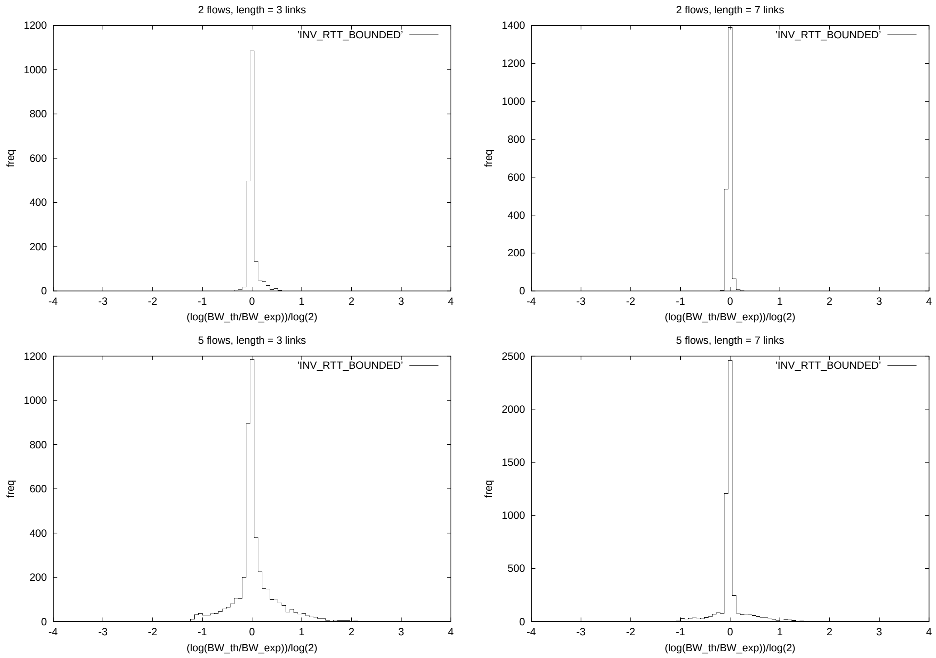

Figure 16: Results of INV-RTT-BOUNDED model.

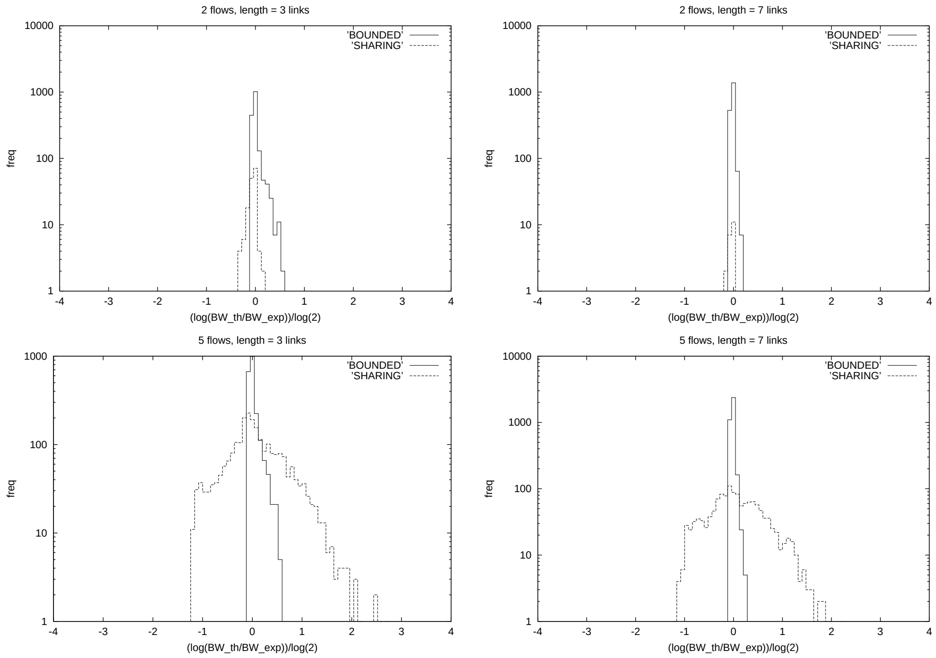

Figure 17: Comparison between bandwidth allocated because of γ/RTT bound, or because of sharing.

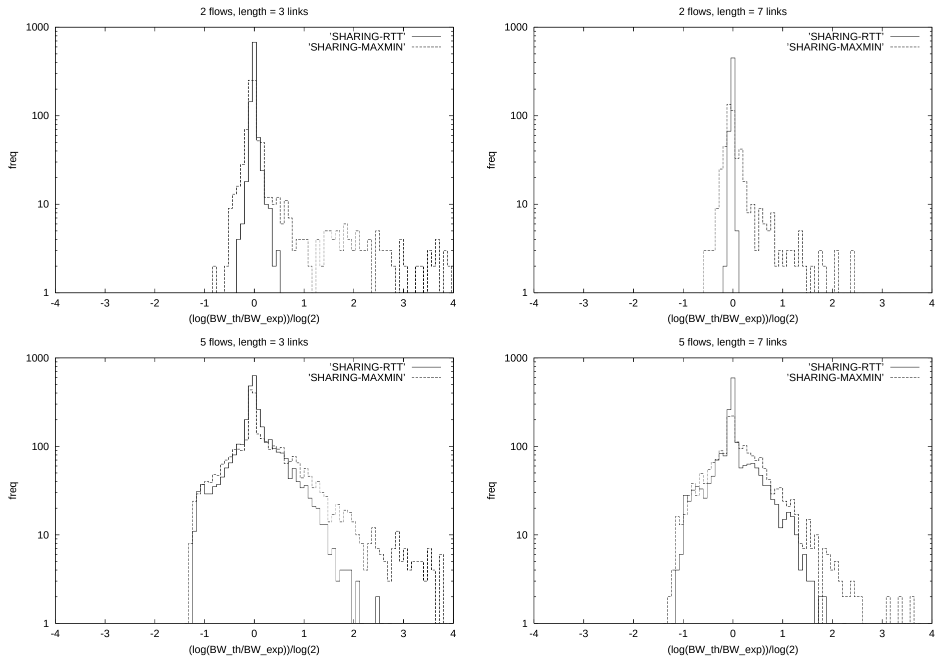

Figure 18: Comparison between INV-RTT-BOUNDED and MAX-MIN in congested networks.

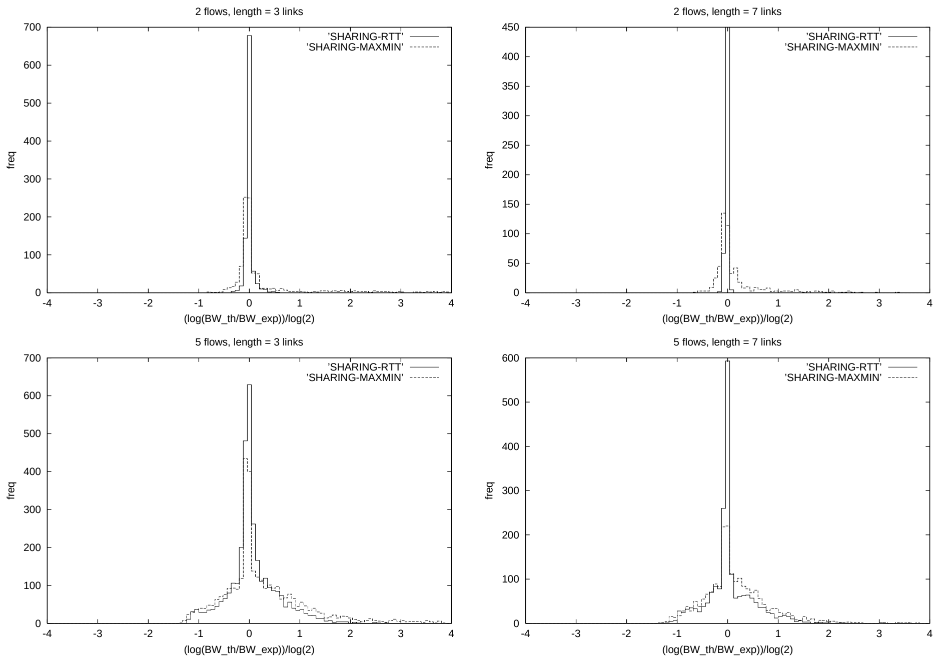

Figure 19: Comparison between INV-RTT-BOUNDED and MAX-MIN in congested networks.

References

[1] D. Bertsekas and R. Gallager, *Data Networks*, Prenctice Hall, 1987.

[2] T. Bonald and L. Massoulié, *Impact of fairness on internet performance*, in SIGMETRICS/Performance, 2001, pp. 82–91.

[3] T. Braun, H. Siegel, and N. Beck, *A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems*, Journal of Parallel and Distributed Computing, 61 (2001), pp. 810–837.

[4] K. L. Calvert, M. B. Doar, and E. W. Zegura, *Modeling internet topology*, IEEE Communications Magazine, 35 (1997), pp. 160–163.

[5] H. Casanova, *Simgrid: A toolkit for the simulation of application scheduling*, 2001.

[6] D. N. Chiu, *Some observations on fairness of bandwidth sharing*, tech. rep., Sun Microsystems, 1999.

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

[8] S. Floyd and V. Paxson, *Difficulties in simulating the internet*, 2001.

[9] I. Foster and C. Kesselman, eds., *The Grid: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann Publishers, Inc., San Francisco, USA, 1999.

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

[11] F. Kelly, *Charging and rate control for elastic traffic*, 1997.

[12] Y. Kwok and I. Ahmad, *Benchmarking and Comparison of the Task Graph Scheduling Algorithms*, Journal of Parallel and Distributed Computing, 59 (1999), pp. 381–422.

[13] L. Massoulié and J. Roberts, *Bandwidth sharing: Objectives and algorithms*, in INFOCOM (3), 1999, pp. 1395–1403.

[14] M. Mathis, J. Semke, and J. Mahdavi, *The macroscopic behavior of the TCP congestion avoidance algorithm*, Computer Communications Review, 27 (1997).

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

[16] *The network weather service.* http://nws.cs.ucsb.edu/.

[17] J. Padhye, V. Firoiu, D. Towsley, and J. Krusoe, *Modeling TCP throughput: A simple model and its empirical validation*, Proceedings of the ACM SIGCOMM '98 conference on Applications, technologies, architectures, and protocols for computer communication, (1998), pp. 303–314.

[18] J. Roberts and L. Massoulié, *Bandwidth sharing and admission control for elastic traffic*, 1998.

[19] R. Wolski, N. T. Spring, and J. Hayes, *The network weather service: a distributed resource performance forecasting service for metacomputing*, Future Generation Computer Systems, 15 (1999), pp. 757–768.