HAL
open science

On the Validity of Flow-level TCP Network Models for Grid and Cloud Simulations

Pedro Velho, Lucas Schnorr, Henri Casanova, Arnaud Legrand

To cite this version:

Pedro Velho, Lucas Schnorr, Henri Casanova, Arnaud Legrand. On the Validity of Flow-level TCP Network Models for Grid and Cloud Simulations. ACM Transactions on Modeling and Computer Simulation, 2013, 23 (4). hal-00872476

HAL Id: hal-00872476

https://inria.hal.science/hal-00872476

Submitted on 13 Oct 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Z

On the Validity of Flow-level TCP Network Models for Grid and Cloud Simulations

PEDRO VELHO, UFRGS, Institute of Informatics, Brazil
LUCAS MELLO SCHNORR, UFRGS, Institute of Informatics, Brazil
HENRI CASANOVA, Dept. of Computer and Information Sciences, University of Hawai'i at Manoa, U.S.A
ARNAUD LEGRAND, CNRS, Grenoble University, France

Researchers in the area of grid/cloud computing perform many of their experiments using simulations that must capture network behavior. In this context, *packet-level* simulations, which are widely used to study network protocols, are too costly given the typical large scales of simulated systems and applications. An alternative is to implement network simulations with less costly *flow-level* models. Several flow-level models have been proposed and implemented in grid/cloud simulators. Surprisingly, published validations of these models, if any, consist of verifications for only a few simple cases. Consequently, even when they have been used to obtain published results, the ability of these simulators to produce scientifically meaningful results is in doubt. This work evaluates these state-of-the-art flow-level network models of TCP communication via comparison to packet-level simulation. While it is straightforward to show cases in which previously proposed models lead to good results, instead we follow the *critical method*, which places model refutation at the center of the scientific activity, and we systematically seek cases that lead to invalid results. Careful analysis of these cases reveal fundamental flaws and also suggest improvements. One contribution of this work is that these improvements lead to a new model that, while far from being perfect, improves upon all previously proposed models in the context of simulation of grids or clouds. A more important contribution, perhaps, is provided by the pitfalls and unexpected behaviors encountered in this work, leading to a number of enlightening lessons. In particular, this work shows that model validation cannot be achieved solely by exhibiting (possibly many) "good cases." Confidence in the quality of a model can only be strengthened through an invalidation approach that attempts to prove the model wrong.

Categories and Subject Descriptors: I.6.4 [**Simulation and Modeling**]: Model Validation and Analysis; I.6.5 [**Simulation and Modeling**]: Model Development; I.6.8 [**Simulation and Modeling**]: Simulation Support Systems

General Terms: Experimentation

Additional Key Words and Phrases: grid and cloud computing simulation, validation, scalability, SimGrid.

ACM Reference Format:
Velho, P., Schnorr, L., Casanova, H., Legrand, A. On the Validity of Flow-level TCP Network Models for Grid and Cloud Simulations *ACM Trans. Model. Comput. Simul.* X, Y, Article Z (July 2013), 25 pages.
DOI:http://dx.doi.org/10.1145/0000000.0000000

This work has been supported by *ANR* (French National Agency for Research) through project references ANR 08 SEGI 022 (USS SimGrid) and ANR 07 JCJC 0049 (DOCCA), by *CNRS* (French National Center for Scientific Research). The authors would like to thank *CNPq* (Brazilian National Counsel of Technological and Scientific Development) for funding the PhD thesis of Pedro Velho. Experiments presented in this paper were carried out using the Grid'5000 experimental testbed, being developed under the INRIA ALADDIN development action with support from CNRS, RENATER and several Universities as well as other funding bodies (see https://www.grid5000.fr).
Author's addresses: A. Legrand, University of Grenoble, France; H. Casanova, Dept. of Computer and Information Sciences, University of Hawai'i at Manoa, U.S.A; P. Velho and L. Mello Schnorr, Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre, Brazil;
Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.
© 2013 ACM 1049-3301/2013/07-ARTZ $15.00
DOI:http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

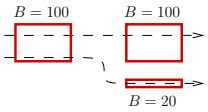
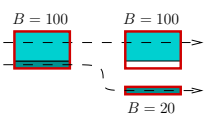
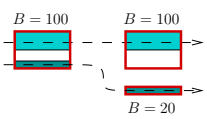
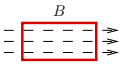
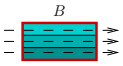
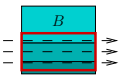
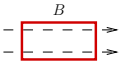
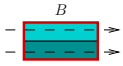
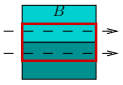
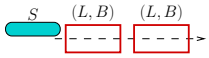
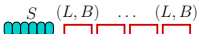
Large scale distributed computing infrastructures such as grids or clouds have become commonplace. Production Grids like EGEE span hundreds of sites with several tens of thousands of processing units interconnected by complex wide area networks. Efficiently exploiting such platforms raises several challenges, e.g., in the area of application scheduling and resource management. Scientific workflows such as the ones executed on EGEE comprise large numbers of computational tasks that require large input datasets and produce large output datasets. Optimizing the execution of such workflows on large-scale platforms is combinatorial in nature, leading to the development of many scheduling heuristics [Blythe et al. 2005; Braun et al. 2001; Topcuoglu et al. 1999]. Although it may be possible to compare scheduling heuristics analytically in some cases, such analyses rely on theoretical models that often ignore or oversimplify important characteristics of real-world deployments (e.g., the macro-dataflow model [Ramaswamy and Banerjee 1993], which assumes that there is no network contention).

Given the above, it is not surprising that most research in grid/cloud computing is empirical in nature, i.e., based on obtaining and analyzing experimental results. However, conducting experiments in large-scale distributed systems, if at all possible, is time-consuming and labor-intensive. Such systems are subject to software and hardware heterogeneity which complicates application deployment. Intermittent and non-controllable load variations and failures typically preclude reproducible experiments. Systems may not be available for the purpose of research experiments that could disrupt production use. Even if a fully controllable system is available, the energy and monetary cost of conducting experiments on such platforms is prohibitive. Finally, researchers may want to study systems that are not necessarily available (e.g., because they are being deployed, to explore "what if" scenarios, or to investigate future evolutions of the platform). For all these reasons, many published results in the area of grid/cloud computing are obtained in simulation.

The simulators that are used in the grid/cloud computing domain rely on various simulation models for compute components (e.g., servers) and network components (e.g., routers, links, network cards). Models that capture the operation of these components in detail (e.g., cycle-level simulation of a processor, packet-level simulation of a router) prove intractable when simulating applications with large computation and communication workloads on large-scale systems with many components. As a result, simulation models must trade off a higher level of detail for a higher compute speed. For example, for simulating data transfers many simulators rely on flow-level network models or on simplistic packet-level network models. Thus arises the question of model validity: to what extent do these trade-offs reduce the accuracy of obtained simulation results and can these results be used to draw scientifically valid conclusions? In this work we study this question in the context of network models for grid/cloud simulations.

Although using valid simulation models seems a prerequisite for developing a useful simulator, it turns out that many popular grid/cloud simulators use network models that have not been (sufficiently) validated. Table 1 illustrates a few simple experiments that invalidate four well-known simulators: OptorSim (2.1, 02/2010) [Bell et al. 2003], GridSim (5.2 beta, 11/2010) [Buyya and Murshed 2002], GroudSim (0.11, 06/2010) [Ostermann et al. 2010], and CloudSim (3.0.2, 11/2012) [Calheiros et al. 2011]. The version we tested were the latest at the time of writing this article. These simulators have been used to obtain results published in hundreds of research articles, and also used as building blocks to develop other simulators [Chen and Deelman 2012; Teng et al. 2011; Shi et al. 2011; Jung and Kim 2012]. Each invalidating experiment in Ta-

Table I. Invalidating experiments for four popular grid/cloud simulators. Network links are shown as boxes labeled with latencies (L) and/or bandwidth capacities (B). Network flows are dashed arrows that traverse one or more flows. Bandwidth allocations are shown as gray fills when applicable, with a different shade for each flow. The table shows both expected results and flawed results produced by each simulator.

Simulator	Setting	Expected output	Actual output	Cause
OptorSim or GroudSim (flow model)				Each link is shared fairly among all traversing flows, leading to underestimated link usage.
GridSim (flow model)				Bandwidth share is updated incorrectly. Flow #1 receives B, flow #2 receives $B/2$, flow #3 receives $B/3$.
CloudSim (flow model)				Fixes the problem in the row above, but problems still occur when flows start at different times.
GridSim (large message model)		Delay $\approx 2L + \frac{S}{B}$	Delay $= 2L + \mathbf{2}\frac{S}{B}$	A large message of size S is sent using store and forward, without any pipelining.
GridSim (small packet model)	 with B and $L \gg 1$	Delay$\approx$ $4L + \frac{S}{W_{max}/RTT}$, where W_{max} is the maximum TCP window size	Delay$\approx 4L + \frac{S}{B}$	Simple packet-level model that does not account for TCP features, such as congestion window and RTT-unfairness.

ble 1 can be devised through inspection of the simulator's source code, and some are even sometimes documented by the developers themselves. Some model validation results have been published for other simulators, but they typically consider only a few cases in which simulation models are expected to work well [Zheng et al. 2004a; Zheng et al. 2004b] (essentially merely verifying that model implementations are correct). However, it is accepted in most of the sciences that model invalidation is an important component of the research activity. Invalidation studies must be conducted that explore a wide range of scenarios for which a model's behavior is either unknown or expected to be invalid. Through these invalidation studies, the model is improved or

refuted, leading to increasingly precise knowledge of the range of situations in which the model's results are meaningful.

In this work, we focus on flow-level network models of the Transmission Control Protocol (TCP) to be used when simulating large-scale grid/cloud distributed systems. For these systems, accounting for network proximity and network topology is paramount for ensuring that simulations are valid. The goal of flow-level models is to capture complex network behavior using tractable mathematical derivations, i.e., that can be computed quickly. Several flow-level models of TCP have been proposed in the literature [Low 2003; Low et al. 2002; Low and Srikant 2004], and we ourselves have proposed such a model [Velho and Legrand 2009], which is used in the SimGrid simulation framework [Casanova et al. 2008]. Our contributions in this work are as follows:

— We obtain new insight into flow-level modeling using a systematic validation approach based on the critical method [Popper 1972].
— This approach invalidates the use of models in [Low 2003; Low et al. 2002; Low and Srikant 2004] for simulation purposes in the context of grid/cloud computing.
— Our approach also suggests several improvements to the model described in [Velho and Legrand 2009], which serves as the basis for network simulation in SimGrid.
— To the best of our knowledge, the improved model, while imperfect, is the most accurate flow-level TCP simulation model available to date for grid/cloud simulations.
— Our approach, and the difficulties encountered, provide important lessons for model development and validation.

The rest of this paper is organized as follows. Section 2 discusses and details directly relevant previous work. Section 3 presents our invalidation study and the improvements it suggests for flow-level models. Section 4 discusses the limits of flow-level modeling, as highlighted by our invalidation study. Finally, Section 5 summarizes our contributions and outlines future research directions.

2. BACKGROUND AND RELATED WORK

Network modeling and simulation can be undertaken in many contexts, thus mandating context-specific discussions of known methodologies and results. For instance, simulations for studying the fine-grain properties of the TCP protocol may have little in common with simulations for studying the scalability of some large-scale parallel computing application. In what follows we discuss works in the area of network modeling categorized by modeling approaches, highlighting the specific contexts in which each approach has been used.

2.1. Packet-Level Models

Packet-level network simulations are discrete-event simulations with events for packet emission or reception as well as network protocol events. These simulations can reproduce the movements of all network packets and the behavior of the whole TCP stack down to the IP level. Packet-level simulations have been widely used for studying fine-grained properties of network protocols. The most popular packet-level simulator is NS2 [Issariyakul and Hossain 2008], while more recent simulators include NS3 [NS3 2011], GTNetS [Riley 2003] and OMNet++ [Varga and Hornig 2008]. The TCP stack models found in simulators such as GTNets or NS2 are simplified versions of the TCP stack. More recent developments [Jansen and McGregor 2007] allow the use of real TCP stack implementations, which is slower but more realistic (i.e., real TCP stack implementations might have features/bugs that are absent from simplified versions used exclusively for simulation purposes).

Except maybe for wireless networks, where the modeling of the physical layer is challenging, packet-level simulation is generally recognized by the network commu-

nity as trustworthy, and it serves as a reference. Unfortunately, in the context of simulation of grids or clouds, it can lead to long simulation times [Fujiwara and Casanova 2007] since the life cycle of each packet is simulated through all protocol layers all the way to a simulated physical layer. Its use is thus often restricted to studying network protocols or applications that exchange small messages. An example of such application is a peer-to-peer Distributed Hash Table implementation, and peer-to-peer simulators have been developed that use standard packet-level simulators [Baumgart et al. 2009]. Nevertheless, some simulators provide the option of using realistic packet-level simulation in spite of its high overhead (e.g., the iCanCloud cloud simulator [Núñez et al. 2011]). To mitigate such overhead, simulators in the area of grid computing (e.g., GridSim [Buyya and Murshed 2002], early versions of SimGrid [Casanova 2001]), have attempted widely simplified packet-level simulation (e.g., wormhole routing, no implementation of any network protocol). Unfortunately, these simulators are easily shown to produce simulated network delays that can be far from those achieved by TCP communications in real networks (see the last row of Table 1). Furthermore, these simplified packet-level models face the same scalability issues, if not as severe, as more realistic packet-level simulators.

For the purpose of simulation, fine-grain network simulation models may be difficult to instantiate with realistic parameter values when targeting large-scale networks as some of the parameter values may be unknown or difficult to obtain (e.g., a full description of a large subset of the Internet). Fortunately, the level of detail provided by packet-level simulation is not necessary for studying large-scale applications that exchange large amounts of data.

2.2. Delay-based Models

In some contexts it is sufficient to simulate simple network delays between pairs of communicating hosts. For instance, a peer-to-peer simulator for studying overlay networks may model each communication delay as a constant delay or as a sample from a given statistical distribution [Montresor and Jelasity 2009]. These models do not account for network proximity. As a consequence, some peer-to-peer simulators [Gil et al. 2005; Montresor and Jelasity 2009; Baumgart et al. 2009] model network delays using coordinate systems. Each peer is provided with coordinates in a Euclidean metric space and the simulator simply computes the corresponding distance in this space to evaluate communication delay. Note that non-Euclidean spaces can lead to more accurate point-to-point communication delays in Wide Area networks [Dabek et al. 2004]. Coordinate-based models provide a good trade-off between compute time and space as they account for network proximity with a $\Theta(N)$ memory footprint and a $O(1)$ computation time for a network delay. Since coordinates may change over time and suffer from measurements inaccuracies [Ledlie et al. 2007], some simulators generally add noise to these coordinates [Baumgart et al. 2009]. In the context of simulating High Performance Computing systems (e.g., clusters of servers), similar models have also been proposed [Culler et al. 1993; Alexandrov et al. 1995; Kielmann et al. 2000; Ino et al. 2001; Hoefler et al. 2010]. These models account for partial asynchrony for small messages. They also include specific parameters depending on message size making it possible to account for protocol switching, which can have a large impact on overall performance. These models are extremely scalable since both description size and delay computation time are in $O(1)$.

All models above, whether for peer-to-peer application or for HPC systems, ignore network contention. In the case of peer-to-peer simulations, the rationale is that the amount of exchanged data is small and that network contention is thus not an important factor in the performance of the application. In fact, often the metric of interest is the number of exchanged messages, rather than the data transfer rates achieved. In

the case of HPC simulations, the rationale is that the target platform is provisioned with a bisection bandwidth so large that network contention is unlikely to occur. While this assumption may be reasonable for high-end systems (e.g., systems built from high-radix optical switches), it does not hold for all commodity clusters.

2.3. Flow-level Modeling

When modeling network contention is needed, one cannot use the scalable delay-based models discussed in the previous section. To achieve scalability higher than that afforded by full-fledged packet-level models, one needs an abstraction of the network topology that focuses on the part of the topology in which contention may happen. For instance, when considering the simulation of a peer-to-peer data streaming application or of a volunteer computing infrastructure in which participants donate compute cycles, a detailed model of the core of the network may not be needed as most of the contention occurs at the edges of the network (e.g., when streaming content, when downloading input data for computation). Conversely, when studying collective communications in a cluster with limited bisection bandwidth, it is necessary to model the core of the network since it is where the contention will occur. Similarly, in platforms that span wide-area networks, applications may experience network contention on some wide-area network paths.

An alternative to expensive packet-level modeling, which still makes it possible to account for network contention, is flow-level modeling. In flow-level models, which are used by simulators in various domains [Bell et al. 2003; Casanova et al. 2008; Ostermann et al. 2010; Giuli and Baker 2002; Zheng et al. 2004a], each communication, or *flow*, is simulated as a single entity rather than as a set of individual packets. The time needed to transfer a message of size S between hosts i and j is then given by:

$$T_{i,j}(S) = L_{i,j} + S/B_{i,j}, \tag{1}$$

where $L_{i,j}$ (resp. $B_{i,j}$) is the end-to-end network latency (resp. bandwidth) on the route connecting i and j. Although determining $L_{i,j}$ may be straightforward, estimating the bandwidth $B_{i,j}$ is more difficult as it depends on interactions with every other flow. This is generally done by assuming that the flow has reached *steady-state*, in which case the simulation amounts to solving a bandwidth sharing problem, i.e., determining how much bandwidth is allocated to each flow. More formally:

> *Consider a connected network that consists of a set of links $\mathcal{L}$, in which each link l has capacity B_l. Consider a set of flows $\mathcal{F}$, where each flow is a communication between two network vertices along a given path. Determine a "realistic" bandwidth allocation ϱ_f for flow f, so that:*
>
> $$\forall l \in \mathcal{L}, \quad \sum_{f \text{ going through } l} \varrho_f \leqslant B_l \ . \tag{2}$$

Given the computed bandwidth allocation (which defines all data transfer rates), and the size of the data to be transmitted by each flow, one can determine which flow will complete first. Upon completion of a flow, or upon arrival of a new flow, the bandwidth allocation can be reevaluated. Usually, this reevaluation is memoryless, meaning that it does not depend on past bandwidth allocations. This approach makes it possible to quickly step forward through (simulated) time, and thus is attractive for implementing scalable simulations of large-scale distributed systems with potentially large amounts of communicated data. However, it ignores phenomena such as protocol oscillations, slow start (even though slow start can sometimes be modeled via an additional communication latency that depends on message size [Clauss et al. 2011]), and more gen-

erally all transient phases between two steady-state operation points. Perhaps more crucially, the whole approach is preconditioned on computing a bandwidth sharing solution that corresponds to the bandwidth sharing behavior of real-world network protocols, and in particular of TCP.

2.4. Flow-Level Models of TCP

Two approaches are used to build models of a computer system, and thus of the network: in the *bottom-up* approach the model is built based on analyzing low-level phenomena; in the *top-down* approach the model is built from observations of the full system. The bottom-up approach should intuitively lead to more accurate models but does not guarantee perfect validity because analyzing low-level phenomena typically requires that approximations be made. One advantage of the top-down approach is that deriving the model is easier since understanding the low-level details of the system is not needed, but the validity of the model is more questionable.

The TCP network protocol is known for its additive increase multiplicative decrease window size policy, which causes the data transfer rate of flows to oscillate. While such oscillation complicates the modeling of the behavior of TCP, in the last decade, following the seminal work in [Kelly et al. 1998], several bottom-up flow-level models and tools for network protocol analysis have been proposed [Mo et al. 1999; Mo and Walrand 2000; Yaïche et al. 2010; Low et al. 2002; Low 2003]. The goal is to provide a quantitative understanding of the macroscopic behavior of TCP given its microscopic behavior, and in particular its window size policy. In these works, the steady-state behavior of the protocol is often characterized as the solution to an optimization problem. By making an analogy between the equations governing expected window size that follow from the specification of TCP and a distributed gradient algorithm, Low *et al.* [Low 2003] have proved that the steady-state throughputs of network flows are similar to those obtained by solving a global optimization problem under the constraints in Eq. (2). We briefly recall the main principle of this modeling approach by illustrating its use for the Reno protocol using RED [Floyd and Jacobson 1993] as a queue policy for routers.

Let $w_f(t)$ be the window size of the emitter of flow f at time t. $w_f(t)$ is thus the number of packets of f for which no ack has yet been received at time t. Let d_f be the equilibrium round trip time (propagation plus equilibrium queuing delay) of f, which is assumed to be constant. The instantaneous data transfer rate $\varrho_f(t)$ is thus equal to $w_f(t)/d_f$. Using the RED protocol, if we denote by p_l the loss probability on link l, the probability of packet loss for flow f is:

$$q_f = 1 - \prod_{l \text{ traversed by } f} (1 - p_l) \approx \sum_{l \text{ traversed by } f} p_l \quad \text{when } p_l \ll 1$$

At time t, f's emitter transmits $\varrho_f(t)$ packets per time unit, and receives (positive and negative) acknowledgments at approximately the same rate, assuming that every packet is acknowledged. On average, f's emitter receives $\varrho_f(t)(1 - q_f(t))$ positive acknowledgments per time unit and each positive acknowledgment increases the window $w_f(t)$ by $1/w_f(t)$ (additive increase). It also receives, on average, $\varrho_f(t)q_f(t)$ negative acknowledgments (marks) per time unit, halving the window for each. The net change to the window between two packet emissions, which occurs roughly every $\delta_f(t) = d_f/\varrho_f(t)$ time units, is then obtained as:

$$w_f(t + \delta_f(t)) - w_f(t) = \frac{1}{w_f(t)}(1 - q_f(t)) - \frac{w_f(t)}{2}.q_f(t) \ .$$

Hence, $\varrho_f(t)$ follows the following ODE:

$$\frac{d\varrho_f}{dt} = \frac{1 - q_f(t)}{d_f^2} - \frac{1}{2} q_f(t) \varrho_f(t)^2 \ .$$

The value $q_f(t)$ accounts for the contention along f's path and for the fact that the capacity of every link should not be exceeded (i.e., that the constraints in Eq. (2) are enforced). Associating a Lyapunov function to this ODE makes it possible to prove that the trajectories converge to a point that maximizes

$$\sum_{f \in \mathcal{F}} \frac{\sqrt{2}}{d_f} \arctan\left(\frac{d_f \varrho_f}{\sqrt{2}}\right) \text{ under constraints in Eq. (2).} \tag{3}$$

Similarly, it can be proven that TCP Vegas with DropTail achieves some form of weighted proportional fairness [Low 2003], i.e., it maximizes the weighted sum of the logarithm of the throughput of each flow:

$$\sum_{f \in \mathcal{F}} d_f \log(\varrho_f) \quad \text{under constraints in Eq. (2).} \tag{4}$$

These bottom-up models are attractive because they capture the specifics of the underlying network protocol. One drawback for using them as the basis for grid/cloud simulation, admittedly not their intended use, is that they involve parameters that may not be straightforward to instantiate by users, e.g., the equilibrium round trip time. The model may be of little use in this context if its instantiation requires in-depth knowledge of networking, or a whole set of complex experimental measurements (either of which would compel users to use unsound "guesses" as parameter values).

Top-down flow-level models have also been proposed. These models are not derived from an analysis of the specification of the TCP protocol, but from observations of the protocol's macroscopic behavior over a range of relevant network topologies. An oft mentioned bandwidth sharing objective, which leads directly to a top-down model, is Max-min fairness. This objective is reached by recursively maximizing

$$\min_{f \in \mathcal{F}} w_f \varrho_f \quad \text{under constraints in Eq. (2),} \tag{5}$$

where w_f is generally chosen as the round-trip time of flow f. There are two rationales for this objective. First, it corresponds to what one would naïvely expect from a network, i.e., be "as fair as possible" so that the least favored flows receive as much bandwidth as possible while accounting through weights w_f for the well-known RTT-unfairness of TCP [Marfia et al. 2007]. Second, there is a simple algorithm for solving the optimization problem [Bertsekas and Gallager. 1992], whereas solving non linear problems (with objectives such as the ones in Eq. (3) or Eq. (4)) requires more elaborate techniques. Previous studies have shown that Max-min fairness does not exactly correspond to bandwidth sharing under TCP but that it is a reasonable approximation in many relevant cases [Chiu 1999; Casanova and Marchal 2002].

2.5. Accuracy of Flow-level Simulation

The overarching question that we study in this work is whether flow-level simulation can provide accurate results, in particular when compared to packet-level simulation. Some simulators do implement flawed Max-min flow-level models that lead to reasonable results in some situations but also lead to unrealistic bandwidth sharing in other situations (see Table 1). Besides such plainly invalid models, there is a striking dearth of validation studies in the literature. Those works that do propose flow-level models [Low et al. 2002; Low 2003] are driven by protocol design goals [Low and Srikant

2004], and thus merely present a few test cases to illustrate the correctness of the models. More in-depth validation studies were conducted by Marchal *et al.* [Casanova and Marchal 2002; Casanova et al. 2003] and by Fujiwara and Casanova [Fujiwara and Casanova 2007]. These studies explore a moderate range of experimental scenarios, but mostly end up exhibiting instances in which the evaluated models work reasonably well. None of these works questions the validity limits of the models.

Evaluating the validity of flow-level models in complex and diverse scenarios raises practical difficulties, such as requiring that real-world networks be configured for each scenario of interest. In the context of grid or cloud computing systems, setting up many network configurations for the purpose of model validation is simply not possible. One convenient solution is to compare results obtained with flow-level models to results obtained using packet-level simulation. This approach raises the question of whether packet-level simulation is representative of real-world networks. Answering this question is out of the scope of this work. However, based on the confidence placed by the network community in its packet-level simulators, it is reasonable in this work to declare packet-level simulation the ground truth. As a result, we talk of the *error* of a flow-level model to denote the discrepancy between its results and the results obtained with packet-level simulators for the same simulated scenario. A precise error metric is defined in Section 3.1.

The work in [Fujiwara and Casanova 2007] provides an evaluation of the flow-level model implemented in the SimGrid simulation framework at that time. Conveniently, SimGrid provides an interface to the GTNetS packet-level simulator, which greatly eases the comparison of flow-level and packet-level results. GTNetS is no longer officially supported and the latest version of SimGrid also provides an interface to NS3. (Both GTNetS and NS3 implement the same TCP stack.) The current version of SimGrid implements flow-level models based either on Max-min fairness or on the model by Low *et al.* [Low 2003]. These capabilities of SimGrid make it a convenient framework for studying the validity of flow-level models. Our recent work in [Velho and Legrand 2009], on which this work builds, has taken advantage of these capabilities to evaluate and improve upon a top-down Max-min flow-level model. In that work the following advances were made:

— **Linearity:** The communication time of a message for flow f is given by $t_f = S_f/\varrho_f + L_f$ where S_f is the message size, ϱ_f is the bandwidth allotted to f, and L_f is the sum of the latencies of the links traversed by f. However, L_f and B_l (used in the computation of ϱ_f) are physical characteristics that are not directly representative of what may be achieved by flows in practice. The protocol overhead should be accounted for, which can be done by multiplying all latencies by a factor $\alpha > 1$ and all bandwidths by a factor $\beta < 1$. α accounts for TCP slow-start and stabilization, which prevent flows from instantaneously reaching steady-state. β accounts for packetization and control overheads. This simple change leads to an excellent approximation for a single flow on a single link when message size is larger than 100KB (with $\alpha = 10.2$ and $\beta = 0.92$) [Velho and Legrand 2009].

— **Flow Control Limitation:** TCP's flow control mechanism is known to prevent full bandwidth usage as flows may be limited by large latencies [Mathis et al. 1997; Floyd and Fall 1999; Jain et al. 2003]. This well-known phenomenon can be captured in a flow-level model by adding, for each flow, the constraint that:

$$\varrho_f \leqslant W_{\max}/(2L_f) \ , \tag{6}$$

where $W_{\max}$ is the configured maximum window size. The validity of this enhancement is demonstrated for a single flow going through a single link [Velho and Legrand 2009].

— **Bottleneck Sharing:** TCP protocols are known to be RTT-unfair, hence when two flows contend for bandwidth on a bottleneck link they are assigned bandwidth inversely proportional to their round trip times [Marfia et al. 2007]. On a simple dumbbell topology, but for a wide range of bandwidth and latency parameters, this round trip time is well approximated by the following formula:

$$RTT_f = \sum_{l \text{ traversed by } f} L_l + \frac{\gamma}{B_l}\,, \tag{7}$$

where γ is a fixed value for all flows. It turns out that $\gamma = 8,775$ provides a good approximation [Velho and Legrand 2009].

Although they may look *ad hoc*, these model enhancements do have sound justifications and parameter values have natural interpretations [Velho and Legrand 2009]. Improvements over basic Max-min were demonstrated on dozens of randomly generated networks with 50 to 200 nodes and 150 flows with randomly chosen sources and destinations. Yet, in spite of good average results, occasionally a flow has a throughput that is more than a factor 20 away from that obtained using packet-level simulation.

3. ON THE (IN)VALIDATION PATH

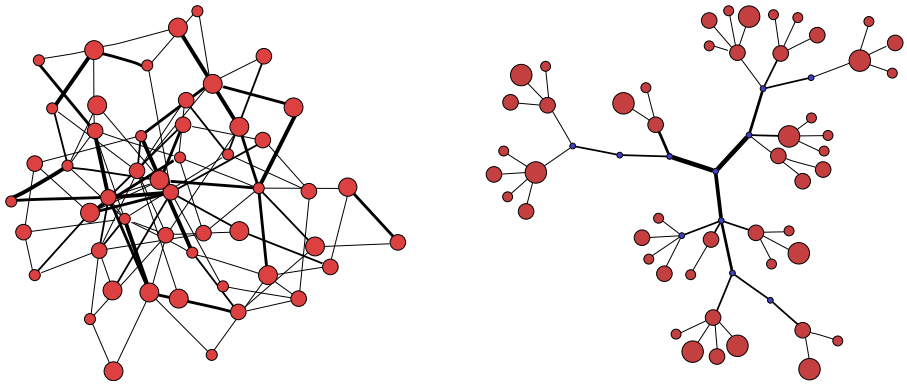
The validity of the bandwidth sharing model itself is not questioned in [Velho and Legrand 2009]. In fact, both Max-min sharing and $\arctan$-based sharing (based on Low *et al.* [Low 2003]) are shown to lead to the exact same bandwidth allocation on simple cases like dumbbell topologies, which is formally provable. Interestingly, although not reported in [Velho and Legrand 2009] because we did not know how to explain it at the time of writing, the use of $\arctan$-based sharing instead of Max-min sharing does not provide any significant improvement for more complex topologies. In other words, the main sources of error seem to be model instantiation errors rather than fundamental flaws of the bandwidth sharing model. This is surprising given that in the literature there is an unstated, but very intuitive, assumption that the bandwidth sharing model itself is of critical importance.

Our goal in this section is to understand, and hopefully resolve, the errors of the flow-level model developed in [Velho and Legrand 2009], thereby obtaining an improved model. As in [Velho and Legrand 2009], we compare the model's results with those obtained with packet-level simulators, configuring these simulators to implement TCP Vegas. We mostly report on packet-level simulation results obtained with GTNeTS but when results seem counter-intuitive we confirm them using NS3 to ensure that our observations are not due to artifacts of the packet-level simulator.

One of our contributions is the method we use for accomplishing our goal. Rather than showing that a model is valid by exhibiting a possibly large set of scenarios in which the model leads to good results, we instead strive to identify cases that "break" the model. This approach follows the *critical method* [Popper 1972], which places model refutation at the center of the scientific activity. The main implication is that inductive reasoning, by which one accumulates observations in which the model is valid, should be banned. Instead, the quality of a model should be established by searching for situations that invalidate the model. Invalidation is done via *crucial experiments* that invalidate the current model, thus motivating the need for a new model. This new model should stand the test of these crucial experiments and should also explain why the refuted model is invalid in these experiments. As a consequence, model parameters that do not have clear significance but make it possible to "fit" a model to a set of observations are to be avoided.

Note that what follows is perhaps atypical in that we repeatedly show "poor" results, while the vast majority of published works strive to show "good" results instead. Nev-

ertheless, following our method, we are able to measure and understand the inherent limitations of an entire class of network modeling techniques.



(a) A BRITE platform: scenarios based on this kind of platform have little contention.

(b) A Tiers platform: hierarchy leads to more contention.

Fig. 1. Two typical 50 nodes topologies used in our study.

3.1. Phase Effects

As described in Section 2.5, the work in [Velho and Legrand 2009] has proposed and evaluated a (top-down) flow-level model based on Max-min optimization. For completeness, we recall here the evaluation methodology therein, which we also use in this work, as well as the final results of this evaluation. Four sets of 10 random topologies were generated with the Waxman model [Waxman 1988] and the BRITE generator [Medina et al. 2001] (Figure 1(a) illustrates a typical 50-node random topology). Although there has been a long-standing debate [Chen et al. 2002; Lakhina et al. 2003; Tangmunarunkit et al. 2002] on network topology characteristics and the existence of power laws [Faloutsos et al. 1999], we use these two generators because at small scale they are probably more meaningful than degree-based generators [Barabási and Albert 1999]. Each of the four sets comprises either small (50 nodes) or large (200 nodes) and either mostly homogeneous (B_l follows a uniform distribution in $[100, 128]$ MB/s) or heterogeneous (B_l follows a uniform distribution in $[10, 128]$ MB/s) topologies. Network link latencies are computed by BRITE based on the Euclidean distance and are in the interval $(0, 5]$ms. Each platform has twice as many edges than nodes. 100 flows are generated between random pairs of end-points, using shortest path routing. 10 different sets of such random flows are simulated for each topology configuration. For the experiment described hereafter, all flows transfer 100MB and the maximum TCP window size is set to 64 KB. GTNetS is configured to simulate TCP Vegas with Droptail as in [Low 2003], hence leading to a system theoretically governed by Eq. (4). Overall, there are 160 experimental scenarios[1].

In the simulation, all flows start at exactly the same time. The simulation ends as soon as one flow completes, and the amount of data transferred by each flow is determined. The rationale for stopping the simulation after the first completion is to avoid computing data transfer rates over a time period in which a variable number of flows are active. Based on the amounts of data transferred, the bandwidth allocated to each

[1]All scenarios, simulation traces and analysis scripts are available at http://simgrid.gforge.inria.fr/network_validation/.

flow is then computed. This enables us to focus on instantaneous bandwidth sharing errors rather than on their accumulation and possible canceling. Such experiments are performed with the Max-min flow-level model in [Velho and Legrand 2009] as well as with the GTNetS packet-level simulator. The *mean error* and the *max error* of the flow-level model for one experiment is computed as:

$$\text{MeanLogErr} = \frac{1}{|\mathcal{F}|} \sum_{f \in \mathcal{F}} \left| \log\left(\varrho_f^{flow}\right) - \log\left(\varrho_f^{packet}\right) \right| \text{ , and}$$

$$\text{MaxLogErr} = \max_{f \in \mathcal{F}} \left| \log\left(\varrho_f^{flow}\right) - \log\left(\varrho_f^{packet}\right) \right| \ .$$

We adopt the above logarithmic error measure because it is completely symmetrical. Consider a standard relative error measure defined as $(\varrho_f^{flow} - \varrho_f^{packet})/\varrho_f^{packet}$. If $\varrho_f^{flow} = 2\varrho_f^{packet}$ its value is 1 (i.e., 100%) whereas if $\varrho_f^{flow} = \varrho_f^{packet}/2$ its value is -1/2 (i.e.,-50%). Instead, the logarithmic error leads to symmetrical values of $\log 2$ and $-\log 2$. Therefore, additive aggregation operators like the maximum, the mean or the variance can be applied to this metric more sensibly. The drawback of this metric is that to interpret the error as a percentage one needs to revert from the log-space. For instance, a logarithmic error of 0.1 corresponds to a relative error of $\exp(0.1) - 1 = 10.5\%$ (close to 10%) but a logarithmic error of 1.0 corresponds to a relative error of $\exp(1) - 1 = 170\%$ (far from 100%).

Figure 2 depicts obtained results for the mean error (left graph) and the max error (right graph). The vertical axis shows the error, and the horizontal axis corresponds to the different experimental scenarios, sorted by increasing error. The main observation is that while some scenarios have low mean and max error, some lead to egregious errors. In particular, the max error reaches a value close to 3. Consequently, over all experiments, there is at least one flow whose bandwidth is a factor $e^3 \approx 20$ away from the bandwidth obtained using packet-level simulation. The negative conclusion is that simulation results obtained with flow-level simulations are "mostly reasonable," which casts doubts about how such simulations could be used meaningfully.

In what follows we investigate the source of the large observed error. A cursory exploration of the results does not reveal any pattern and over- or under-estimation errors seem random. Instead of seeking a pattern in the existing results, we pick a scenario in which at least one of the flows has large error, and iteratively remove flows that do not lead to significant error. We obtain a simplified experimental scenario (only two flows and a few links) that still leads to large errors. Finding that the source of the error remains mysterious even in such a simple case, we attempt to round up platform parameters (latency and bandwidth) to integral values to allow a by-hand analytical evaluation of the flow-level bandwidth sharing model. Surprisingly, the results from packet-level simulation after this rounding off change dramatically. In fact, in our experiments it can easily be shown that packet-level simulation is extremely sensitive to platform parameters. For instance, a slightly modified latency can lead to a radically different bandwidth sharing among two competing flows, a phenomenon which cannot be captured by a (continuous) flow-level model. This phenomenon, which is well-known in the network community, is called *phase effect* [Floyd and Jacobson 1992] and is due to the default Droptail router queue policy algorithm. An *artifact* of deterministic packet-level simulation is that it makes phase effects significantly more likely. Fortunately, phase effects are unlikely in actual wide-area networks because of frequent small burst flows that randomly break the Droptail queue pattern. That is why most recent routers now include implementations of RED or WRED policies that do not behave deterministically (phase effects were one of the motivations for devel-

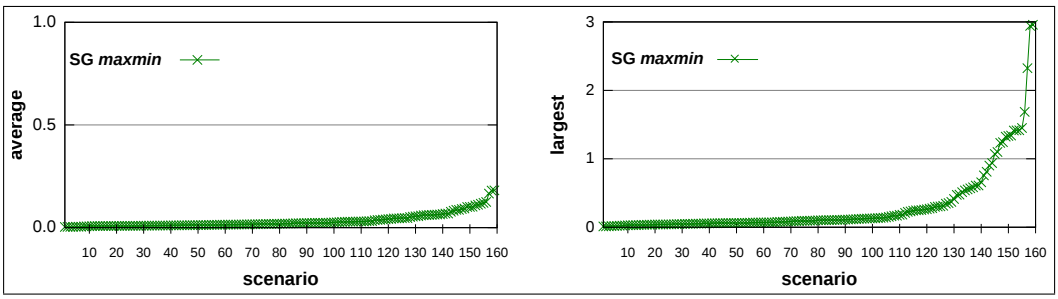


Fig. 2. Mean logarithmic error (left) and max logarithmic error (right) for all experimental scenarios for the flow-level model in [Velho and Legrand 2009].

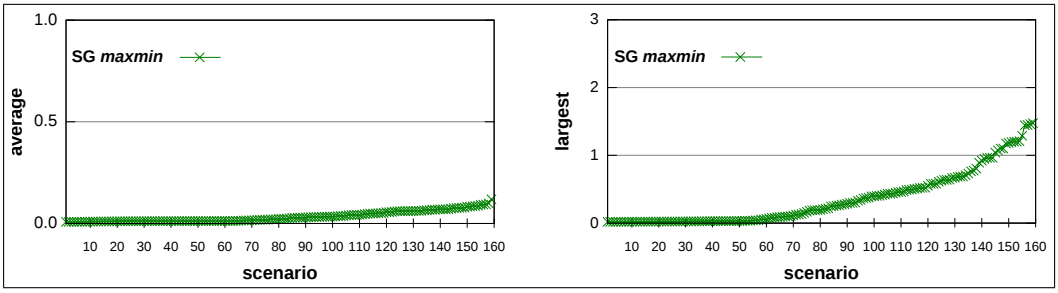


Fig. 3. Mean logarithmic error (left) and max logarithmic error (right) for all experimental scenarios for the flow-level model in [Velho and Legrand 2009], after removing *phase effects* by configuring GTNetS with the RED queue policy.

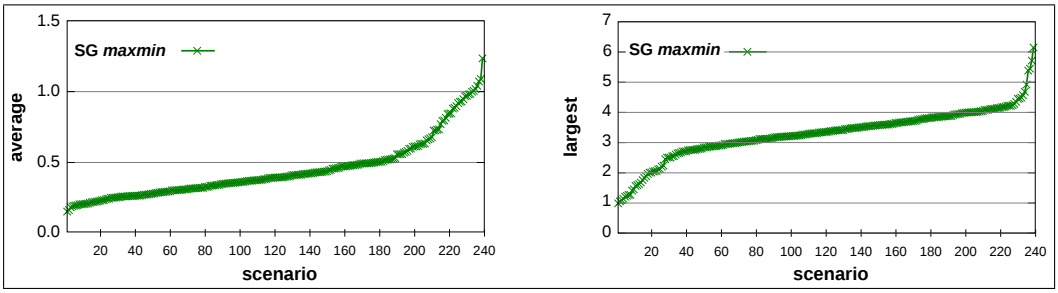


Fig. 4. Mean logarithmic error (left) and max logarithmic error (right) for all experimental scenarios in the critical workload described in Section 3.2 (reduced bandwidth values, hierarchical topologies).

oping RED-like policies). While "rediscovering" phase effect is not a new contribution of this work, it is important to point out that we were able to do so thanks our critical method approach to model invalidation.

Note that we do not claim the superiority of queue policy against another one but we simply notice that phase effects significantly hinder the evaluation of scenarios using Droptail with packet-level simulation and make it inadequate for comparison with fluid models. We can now revisit the results in Figure 2 after configuring GTNetS so that the RED policy[2] is used instead of Droptail. Since the work of [Low 2003] provides fluid approximations for Vegas/Droptail and Reno/RED, we addition-

[2]The RED parameters used were the default from GTNetS, which are the most commonly found in the literature [Floyd and Jacobson 1993; Pentikousis 2001; de Cnodder et al. 2000]: the weight of the queues was set to 0.002, max (resp. min) threshold was set to MiB (resp. max/3), maximum size queue was set to

ally reconfigured GTNetS so that it uses Reno, hence leading to a system theoretically governed by Eq. (3) as demonstrated in [Low 2003]. The flow-level model needs to be re-instantiated as well, meaning that the α, β, and γ parameters described in Section 2.5 must be re-estimated based on new packet-level simulation results obtained for elemental experimental settings. The resulting values are: $\alpha = 13.01$, $\beta = 0.97$, and γ = 20537.14. Newly obtained results for the 160 experimental scenarios are shown in Figure 3. The mean error decreases marginally. More important, the max error is greatly improved from close to 3 (a factor $e^3 \approx 20$) to a value below 1.6 (a factor lower than $e^{1.6} \approx 5$). We now know that the very large errors that remained unexplained in [Velho and Legrand 2009] were caused by phase effects. While a factor 5 is better than a factor 20, there is still much room for improvement. Furthermore, comparing the graph on the right-hand side of Figure 2 to the graph on the right-hand side of Figure 3, we see that there are no longer clear "outliers" in term of maximum error. Although outliers correspond to poor results, in our invalidation approach outliers are helpful as they make it easier to identify flaws in the model. Without outliers, identifying the sources of error is thus potentially more difficult. The critical method drives the search for more critical scenarios, i.e., scenarios for which it is easier to devise crucial experiments.

3.2. Picking a critical workload

Following the critical method, one should evaluate a model on cases in which it performs poorly. While some of the experimental scenarios used in the previous section lead to substantial errors, many lead to low mean error, and to a lesser extent to low max error. Consequently, many of these scenarios fall into the "accumulating cases in which the model is effective" category instead of being "crucial experiments." Inspecting the scenarios in detail, we notice that most flows are throughput-limited by their RTTs. Such flows are good cases for a flow-level model because the constraints in Eq. (6) essentially bypass the core of the bandwidth sharing formulation. This observation is confirmed in Figure 5. The top part of the figure plots the max error over all experiments, sorted by increasing error. The bottom part, which uses the same experiment order, plots the percentage of flows that are RTT-bound. We see a clear inverse correlation between the max error and the fraction of flows that are RTT-bound.

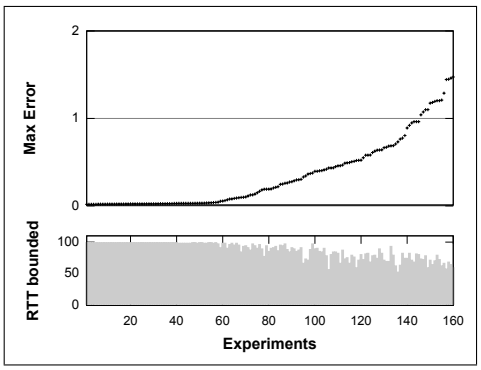


Fig. 5. Max error and the percentage of RTT-bound flows for all experiments, sorted along the horizontal axis by increasing max error.

We address this shortcoming of our dataset in two ways. First, we regenerate our dataset so that bandwidths are sampled uniformly in the interval $[10, 12.8]$MB/s and latencies are sampled uniformly in the interval $[1, 30]$ms. While such low bandwidth values are likely uncommon and impractical in today's (wired) networks, they should increase the number of flows that are not RTT-bound. Second, we use an additional alternative topology model. The Waxman model used in our original dataset is likely to create several possible paths between two sets of end-points (Figure 1(a) shows an example of this type of topology). The number of flows contending for bandwidth on the same link may be low, meaning that most flows could be RTT-bound. Consequently,

400Mib, marking probability was set to 0.02, and average packet size was set to the same as TCP segment size (i.e., 1000B).

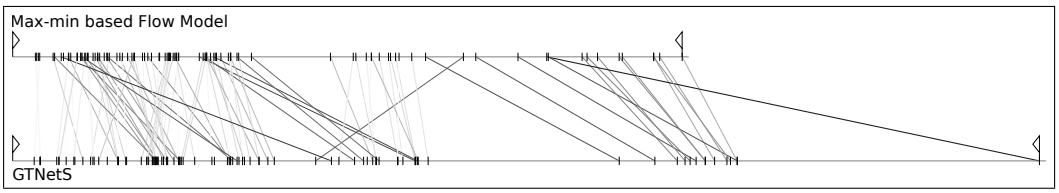


Fig. 6. Comparison of flow completion times with flow-level simulation (top timeline) and packet-level simulation (bottom timeline); lines connecting timeline markers correspond to the same flow in both simulations and are darker for larger completion time mismatches.

the Waxman model may not be ideal for generating crucial experiments when evaluating bandwidth sharing models. By contrast, Figure 1(b) shows an example topology created with the Tiers algorithm [Doar 1996], which uses a three-step space-based hierarchical approach. The resulting topologies are hierarchical with low bisection bandwidth, and have thus both global (in the core of the network) and local (on the edges of the network) bottleneck links. We use Tiers to generate an additional set of 80 experimental scenarios for further invalidating the flow-level model. Figure 4 shows results for all $160 + 80 = 240$ topologies in our new combined dataset. We see that the mean error is now larger than 1.25 (compared to below 0.5 for only the Waxman topologies), and the max error is now above 6 (compared to below 2 for only the Waxman topologies).

Note that in Figure 4, results have been re-sorted. Hence, the first 160 values of Figure 3 do not correspond to the first 160 values of Figure 4. It turns out that the 80 Tiers scenarios are randomly interspersed with the 160 Waxman scenarios. Although hierarchical topologies tend to exhibit higher error, some non hierarchical topologies also lead to high maximum and average error. Still, it can be seen that the 160 scenarios from Figure 3 have a much higher error in Figure 4, which is caused by bandwidth reduction. It is with this new set of experimental scenarios that we continue the (in)validation study of our flow-level model in the next section.

3.3. Reverse traffic effects

Our new experimental scenarios have larger errors, making "bad cases" easier to identify. To understand the root causes of the errors we use a multi-scale, interactive and exploratory visualization tool called Triva [Schnorr et al. 2011; Schnorr et al. 2010; Triva 2011]. We select some of the scenarios that have large maximum error and run them until completion of all flows so that we can compare flow completion times and not only instantaneous bandwidth shares. Figure 6, which is created using Triva, shows two timelines, the top one corresponding to our flow-level model and the bottom one corresponding to packet-level simulation. Each timeline marker represents the completion time of a flow. Triva connects two markers on the two timelines by a line if they correspond to the completion of the same flow. In a no-error situation, all these lines are vertical. For easier visualization, Triva uses gray shades for these lines: the darker the line the larger the error (i.e., the less vertical the line). It is thus straightforward to identify which flows have large errors without being "distracted" by low-error flows.

Three observations can be made from Figure 6. First, the flow-level timeline is shorter, indicating that the simulated time for the entire workload is shorter with flow-level simulation than with packet-level simulation (about 35% in this example). Second, some flows do have low errors but many of them have large errors. Third, except for one flow that completes significantly earlier in packet-level simulation than in the flow-level simulation, the flow-level simulation almost always underestimates flow

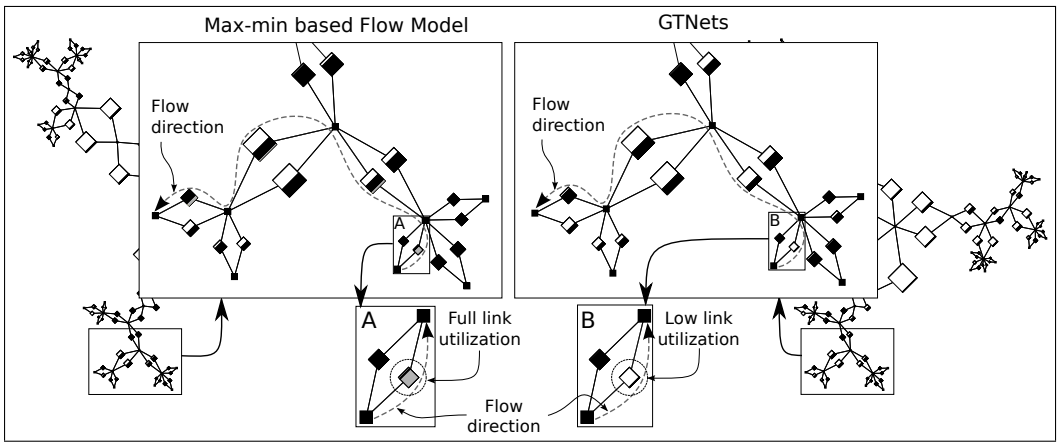


Fig. 7. Visualization of network link utilization when all communication flows are active in the topology with a highlighted large-error flow.

completion time. In spite of our best efforts we were not able to explain the behavior seen for this particular flow.

We now use a graph-based visualization, another Triva feature, which shows the bandwidth utilization for all networks links in the topology, paying close attention to those large-error flows identified in Figure 6. This visualization is shown in Figure 7. Routers and hosts of the platform are represented by black squares, while network links are represented by diamonds. The size of a diamond is proportional to the link's bandwidth capacity. A dark color fill indicates link utilization (the more filled the diamond the more utilized the link). Utilization is computed as an average over a given time frame.

Figure 7 shows two visualizations, one for flow-level simulation (left) and one for packet-level simulation (right). Routing is identical in both cases. Utilization values are averaged over the time from the onset of the simulation until the completion time of the first flow, thus ensuring that all flows are present in our visualization. We select a flow with large error, and show part of its network path in a zoomed region of the full topology. In the case of flow-level simulation, the network link that limits this flow, shown in the region labeled A, is, as expected, fully saturated. A gray fill in the figure shows the capacity allocated to our target flow. Surprisingly, in packet-level simulation a much lower bandwidth share is allocated to this flow and this network link, now in the region labeled B, is not saturated at all. In fact, our target flow receives an insignificant fraction of the bandwidth along the whole path whereas none of the links on the path are saturated. We conclude that TCP under-utilizes this network path and that a bandwidth sharing model based on a global optimization under the constraints in Eq. (2) has no chance of ever capturing this behavior. The same observation can be made for other flows in this experimental scenario, over other time frames, and in other experimental scenarios. Under-utilization of network links by TCP is thus a key reason that contributes to the large errors seen in our experimental scenarios: our flow-level model gives "too much" bandwidth to flows.

To better understand the cause of the error we remove all the flows that do not use the link that showed widely different utilization in flow-level and packet-level simulations. The simplified experimental scenario still suffers from the same discrepancy between the two simulations. Our expectation is that on a single full-duplex link the bandwidth allocated to flows going in one direction is independent from that allocated to flows going in the reverse direction. However, the above simulation results seem to

contradict this notion. To invalidate it we conduct "crucial experiments" using packet-level simulation as follows. Let us denote by $\{B, (n_1, n_2)\}$ a scenario with a single full-duplex bottleneck link of capacity B, n flows going through the link in one direction, and p flows going through the link in the reverse direction. We denote the outcome of the scenario by (B_1, B_2), where B_1 is the bandwidth allocated to the n flows and B_2 is the bandwidth allocated to the p reverse flows (in all cases all flows in the same direction are allocated the same bandwidth).

Figure 8 depicts five example scenarios, showing approximate results achieved in simulation. The bandwidth allocation for $\{B, (n, 0)\}$ (resp. $\{B, (0, p)\}$) is always approximately $(B/n, 0)$ (resp. $(0, B/p)$). Since the network link is full-duplex, expectedly simulations also show that $\{B, (1, 1)\}$ leads to the allocation (B, B). Likewise, $\{B, (2, 2)\}$ leads to $(B/2, B/2)$. Surprisingly though, $\{B, (2, 1)\}$ does not lead to $(B/2, B)$ as one would expect but instead to $(B/2, B/2)$. More generally, our experiments show that $\{B, (n_1, n_2)\}$ leads to allocation $(B/\max(n_1, n_2), B/\max(n_1, n_2))$. Given how surprising this result seems, one may wonder whether it is not a simulation artifact. We conducted dozens of experiments on real-world networks, with host pairs connected via a direct full-duplex link and host pairs connected via a sequence of full-duplex links. The phenomenon above was confirmed in every tested configuration. Using network monitoring tools, namely `tcpdump` and `wireshark`, we were able to understand that link under utilization is due to the *interference* between acknowledgments for the traffic in one direction and the traffic in the other direction, which we term reverse traffic. Acknowledgment packets are queued with data packets from the traffic, thus slowing down the reverse traffic. It turns out that this phenomenon is known in the network community as *ACK compression* [Zhang et al. 1991], and is very common for ADSL connections. In perfectly symmetrical cases, although *ACK compression* may still occur, delayed ACKs should make it disappear. As recently noted in [Heusse et al. 2011], the poor link utilization we observe is more likely to be explained by a *data pendulum* effect (also known as *ACK-clocking*) where data and ACK segments alternatively fill only one of the link buffers. At any rate, such a phenomenon, which is not modeled by any of the previously mentioned flow-level models, results in seemingly over-utilized bottleneck links in flow-level simulations. Once again, through our use of the critical method, we have "rediscovered" a low-level network phenomenon.

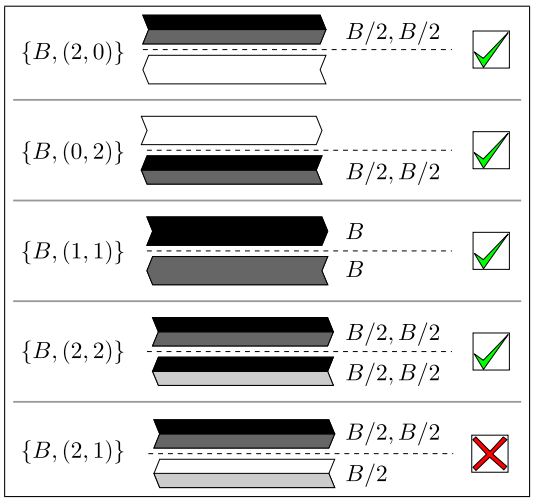
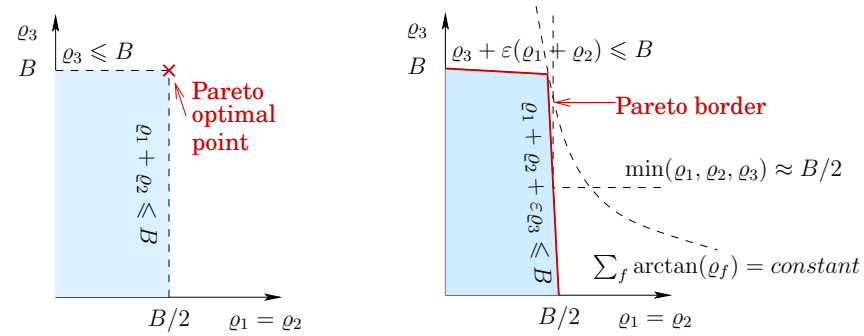


Fig. 8. Five reverse traffic interference scenarios: $\{B, (n_1, n_2)\}$ denotes a scenario with one full-duplex link of capacity B, with n flows in one direction (shown above the dashed lines) and p flows in the reverse direction (shown below the dashed lines). The bandwidths allocated to the flows are shown on the right-hand side of each scenario. The last scenario shows an asymmetric situation.

3.4. Accounting for reverse traffic

It turns out that, under the constraints in Eq. (2), our flow-level model is inherently incapable of capturing the reverse-traffic effect identified in the previous section. To illustrate this, consider a scenario with two hosts M_1 and M_2 connected via one link



(a) Utility set defined by constraints in Eq. (8)

(b) Utility set defined by constraints in Eq. (9)

Fig. 9. Consequence of utility set deformation

of capacity B. If we have two flows from M_1 to M_2 and one flow from M_2 to M_1, the constraints in Eq. (2) are rewritten as:

$$\begin{cases} \varrho_1 + \varrho_2 \leqslant B \\ \varrho_3 \leqslant B \end{cases} \quad \text{and (by symmetry) } \varrho_1 = \varrho_2 \ . \tag{8}$$

Figure 9(a) depicts the corresponding utility set, in which there is a single Pareto optimal point $(\varrho_1 = \varrho_2, \varrho_3) = (B/2, B)$. This point is far from the actual bandwidth share achieved by TCP due to the reverse-traffic effect. And yet, formulating bandwidth sharing as an optimization problem always produces a Pareto-optimal solution. Consequently, such flow-level models cannot account for reverse-traffic effects with the constraints in Eq. (2). As a solution, we propose to change these constraints as follows:

$$\forall l \in \mathcal{L}, \quad \sum_{f \text{ going through } l} \varrho_f + \varepsilon \cdot \left(\sum_{f\text{'s ack through } l} \varrho_f \right) \leqslant B_l \ . \tag{9}$$

Figure 9(b) depicts the utility set defined by these new constraints. The utility set is deformed so that the unique Pareto-optimal point has become a whole Pareto border. With such constraints, the solution of Max-min optimization is much closer to the outcome of packet-level simulation. We obtain a new Max-min model that we can compare to packet-level simulation and to our original model.

Figure 10 is similar to Figure 6, but also displays the timeline for the improved model at the bottom. It can plainly be seen that there are fewer dark lines to the time-

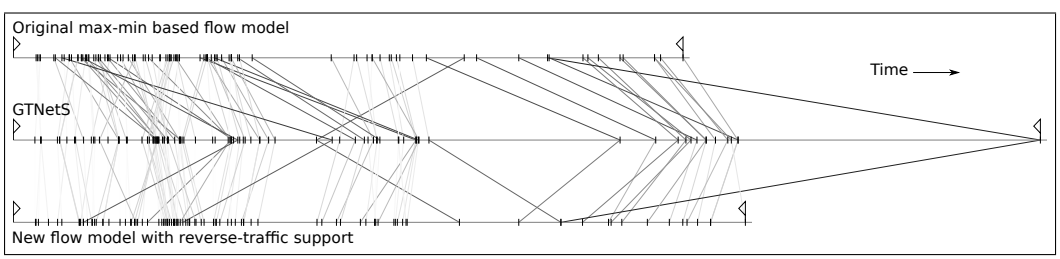


Fig. 10. Comparison of flow completion times with the original flow-level simulation (top timeline); packet-level simulation (middle timeline); and the improved flow-level simulation with modified constraints to account for reverse-traffic (bottom timeline).

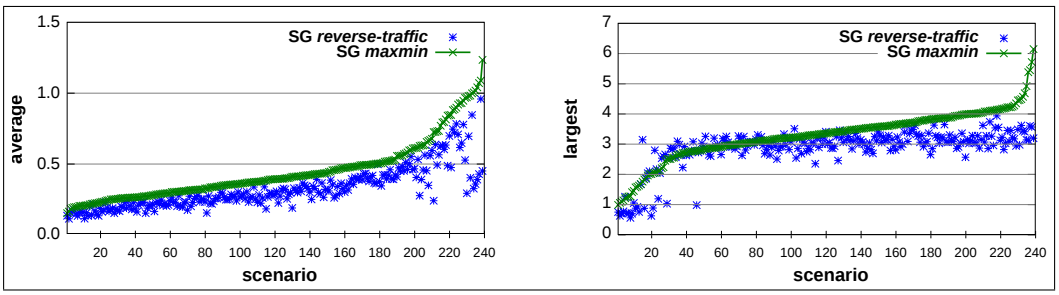


Fig. 11. Mean logarithmic error (left) and max logarithmic error (right) for all experimental scenarios for the flow-level model SG_{maxmin} in [Velho and Legrand 2009] and the improved reverse-traffic model $SG_{reverse\text{-}traffic}$ proposed in Section 3.3, evaluated with topologies generated by Tiers.

line at the bottom than to the timeline at the top. The modified constraints improve the quality of our flow-level model.

Figure 11 shows the benefit of the modified model over the original model for the entire set of critical experimental scenarios, with scenarios sorted by increasing error of the original model. We see that the mean error is always improved by the modified model, while most max errors are also improved. Importantly, the largest max error with the modified model is below 4, while it was higher than 6 with the original model.

3.5. Comparison with bottom-up modeling

Figure 12 shows results for our improved Max-min model and the model from [Low 2003], which is attractive because it is built bottom-up from first principles of the TCP algorithms. Experimental scenarios are sorted by increasing error of the Max-min model. We see that the mean error is almost always slightly larger for the model in [Low 2003] than that obtained with our improved top-down Max-min model. More importantly perhaps, the Max-min model leads to significantly lower max error, i.e., better worst-case behavior. Note that reverse-traffic issue is not encountered and thus not identified in [Low et al. 2002; Low 2003; Low and Srikant 2004] because all flows are along the same direction in their experiments.

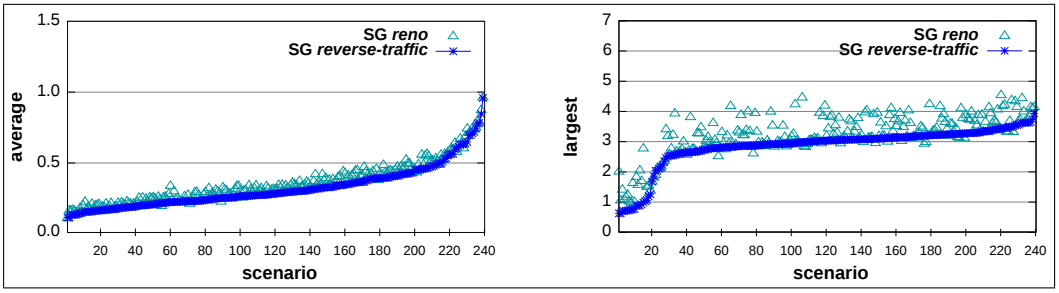


Fig. 12. Mean logarithmic error (left) and max logarithmic error (right) for all experimental scenarios for the reverse-traffic aware model based on Max-min, $SG_{reverse\text{-}traffic}$, and the arctan-based model in [Low 2003], SG_{reno}.

It is not surprising that the model in [Low 2003] leads to high error with reverse-traffic since the absence of interference between acknowledgments and data traffic is one of the main assumptions used to build bottom-up flow-level models (i.e., making the analogy between the equations governing expected window size and a distributed gradient algorithm that optimizes a sum-based function). [Low 2003] defines "d_f, *the*

equilibrium round trip time (propagation plus equilibrium queuing delay) of flow f, which is assumed to be constant." However, d_f is only a constant for a given traffic, meaning that it actually depends on the throughputs of the other flows, which influence the queuing delay. Consequently, it is difficult to use the models in [Low 2003] as a basis for simulation since there is a circular dependency between the d_f parameter, which is needed to instantiate the model, and the throughput of the simulated flows. For the results in Figure 12 we set d_f to the end-to-end latency of each flow f, which is a constant. While it may be difficult to make good choices of d_f values when instantiating the model for the purpose of simulation, one may wonder about the impact of these choices. It turns out that the shape of the isolines of sum-based objective functions is such that, unless the utility set is heavily distorted, the Pareto-optimal solution is always the rightmost and uppermost vertex (see Figure 9), which we have seen is far from the bandwidth share achieved by TCP. This is thus also the case with the model in [Low 2003] since its objective function is a sum of $\arctan$ or $\log$ functions. Consequently, regardless of the chosen d_f values, there is little hope that the model could lead to good results in the presence of reverse-traffic. We conclude that although bottom-up modeling is attractive, a bottom-up model that does not account for reverse-traffic effects is inferior to a top-down model that does, such as our own modified Max-min model.

The reverse-traffic limitations of the model in [Low 2003] have actually been identified and acknowledged by the authors in subsequent work. Reverse flows cause small scale burstiness in sub-RTT timescales that impact flow level stability and equilibria [Tang et al. 2008; Tang et al. 2010]. Tang *et al.* propose new *Differential Algebraic Equation* (DAE) models that link window size, instantaneous throughput and congestion measures more finely than [Low et al. 2002; Low 2003; Low and Srikant 2004]. These improvement make it possible to capture burstiness and hence macroscopic phenomena. In particular, Jacobsson *et al.* model and study ACK-Clocking dynamics in [Jacobsson et al. 2008]. More specifically, they study fine interactions between different versions of TCP, deriving stability conditions of specific versions of TCP, and the interaction of paced and unpaced flows. In simple situations there is an almost perfect match between NS2 simulations and the DAE model, even in not-yet stabilized phases.

While these improved models are promising, it is unclear whether they can be used effectively in the context of simulation (this is admittedly not their intended use, since the aforementioned works focus on understanding TCP behavior rather than producing instantiable simulation models). First, all these models require solving complex differential equations, which would likely prove unscalable if used to drive simulations with hundreds of flows. Second, although such work is motivated by burstiness caused by reverse-traffic, the reverse-traffic considered in the validation experiments of [Tang et al. 2008; Jacobsson et al. 2008; Tang et al. 2010] is always UDP-based and does not saturate links in the opposite directions, hence leaving room for acknowledgment packets. This situation does not correspond to all relevant simulation scenarios. Third, the experimental scenarios used to evaluate the models involve at most three nodes and three flows, and one may wonder how the models would behave if used for simulating hundreds of flows. Consequently, although in the future effective bottom-up simulation models could arise, for the time being our top-down model above appears to be the best available flow-level network simulation model to date, at least in the context of large-scale grid/cloud computing simulations.

4. LIMITS OF FLOW-LEVEL MODELING

Accounting for reverse-traffic effects has improved our flow-level model substantially in terms of mean error over a range of crucial experimental scenarios. However, when

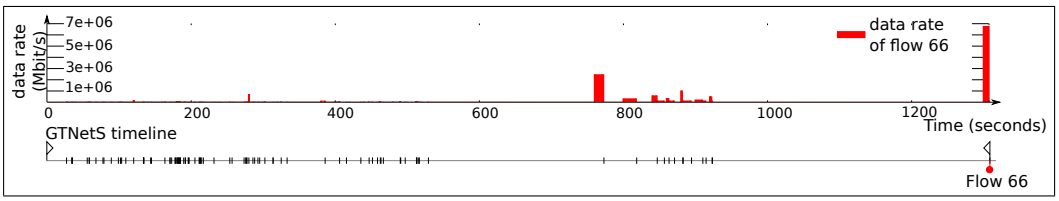


Fig. 13. Evolution of the throughput of the last finishing flow, which has large error. Because of high contention, this flow stalls 65% of the time. When there is no other remaining connection in the network, it does not transmit a single byte for 380 seconds, and finally completes the transfer in a few seconds.

considering flow completion times rather than bandwidth shares when all flows are active, many worst-case scenarios show no improvement at all. These flows complete generally much later in packet-level simulation than in flow-level simulation, and the cause of these errors is not the reverse-traffic effect. Using our visualization tool we attempted to find a topological pattern common to all these worst-case scenarios, but were not successful. We also considered particular flows such as the ones that surprisingly complete earlier in GTNetS than within the flow model but again, we failed to identify any particular topological pattern that would explain these differences. We also tried the traditional approach of simplifying scenarios by iteratively removing flows to isolate those with the worst errors. These simplifications were also unsuccessful as errors always disappeared early in the iterative process.

Eventually, we isolated "unexpected" behavior in the packet-level simulations, e.g., for the last flow to complete in the packet-level simulation timeline in Figure 10. If this flow were the only flow, packet level simulation shows that it would complete in less than 4 seconds. And yet, in the full simulation, the next-to-last flow finishes 390 seconds before the completion of this last flow! Analyzing the communication trace from the simulation shows a long period of inactivity for this flow. Figure 13 shows the data transfer rate of the last flow vs. time. We see that this flow receives some bandwidth in the beginning of the simulation, then stalls for 380 seconds, and then completes rapidly at the end of the simulation, receiving the full bandwidth of all the network links on its path. These results are with the GTNetS packet-level simulator, but we have observed the same behavior with NS3, thus indicating that this phenomenon is likely not a simulation artifact. In fact, the long inactivity period is due to TCP timeouts that occur for highly contended flows. We surmise that such effects may be impossible to model using a (continuous) flow-level model. A simulation of the state of the TCP stack would certainly be required, thus blurring the line between flow-level and packet-level simulation.

We conclude that the quality of our improved Max-min model is high for large messages in almost all the scenarios we have explored, the exceptions being extremely contended ones (i.e., with extremely small latencies and low bandwidth capacities). In these scenarios, the high error is due to the discrete nature of the TCP protocol, which, by design, is not captured by simple flow-level models. While as part of our critical method approach we have purposely designed these extreme scenarios to invalidate our approach, it is comforting that such scenarios are bound to be unlikely, or so low-performance as to be irrelevant, in practical (simulation) settings.

5. CONCLUSION

When simulating large-scale grid/cloud computing systems and their applications, the use of packet-level simulation for network communications, albeit widely accepted to be accurate, typically proves unscalable. An alternative is to use simulation that relies on flow-level models of network communications. Several flow-level models of

TCP have been proposed and used in the literature but, perhaps surprisingly, few works have thoroughly investigated their validity. One such work is our previous study in [Velho and Legrand 2009]. Although the core of a flow-level model is the bandwidth sharing model, that study showed that bandwidth sharing was not the primary source of model errors: errors were due primarily to poor instantiating of model parameters. The final evaluation in [Velho and Legrand 2009] showed that a model based on Max-min fairness leads to good results on average even in complex scenarios. However, some flows suffered from very large errors, which remained unexplained. Generally speaking, it is never possible to establish the validity of an empirical model entirely. Instead, one must seek to invalidate the model as a way to ascertain the amount of confidence that can be placed in it, and, perhaps, in the process propose improvements. In this article, we have followed such an invalidation path.

Our first source or error comes from the reference packet-level simulator, which was configured to use the Droptail router queue policy. This choice was motivated by the aim of ultimately comparing with the fluid model of Vegas/Droptail proposed in [Low 2003]. However the use of Droptail creates simulation artifacts known as phase effects, which compromise the results in [Velho and Legrand 2009]. When removing these artifacts by using the RED queue policy, errors are decreased overall. It is interesting to note that we rediscovered this issue due to our implicit trust in our flow-level model, but this trust was not fully justified since large errors remained. While looking for a second source of error, we determined that our benchmark workload, although seemingly standard, led to low-contention experimental scenarios. These scenarios are easy cases for flow-level models since with low contention the core of the bandwidth sharing algorithm is bypassed. We thus generated a new set of experimental scenarios and, expectedly, modeling errors were vastly increased. Actively seeking scenarios in which the error of the model proposed by the authors is as large as possible is perhaps atypical in the literature. Nevertheless, it is the course of action dictated by the critical method.

Turning our attention to the sources of these enlarged errors, we eventually discovered that many were due to reverse-traffic interference. The reason is that the data transfer rate of a TCP flow is strongly dependent upon the rate at which acknowledgment packets arrive, which is easily demonstrated to be influenced by flows going in the reverse direction both in packet-level simulation and in real-world experiments. Fortunately, it is straightforward to modify our Max-min-based flow-level model to account for this reverse-traffic effect. By contrast, such modifications seem out of reach for the flow-level models proposed by others in the literature. In fact, these models were built in a bottom-up fashion, completely ignoring reverse-traffic effects. Furthermore, their "validation" had been done by illustrating their accuracy on a few simple cases. It is likely that another model (like our Max-min-based model) would have produced the same results in these simple settings, showing the inherent problem with a methodology that only explores "good cases."

To the best of our knowledge, our Max-min based model augmented with reverse-traffic support is the best flow-level model of TCP available to date to drive simulations in the context of grid/cloud computing. Our results show that this model leads to high-quality results in a wide range of settings although it is far from being perfect. We have found situations in which the bandwidth allotted to a flow according to the model is very different from the bandwidth allotted to it according to packet-level simulation. A detailed study of these results shows that flow-level models have little hope to model such situations correctly. Yet, these situations are extreme (high contention on links with very low capacity) and thus unlikely in most practical settings.

Our investigation of flow-level models seems to have reached its end because the remaining erroneous behaviors can be imputed to violations of the steady-state as-

sumption, which is the central tenet of flow-level modeling. A future direction would consist of providing sufficient conditions for the steady-state assumption to be invalid. This would allow us to better identify the validity domain of flow-level models and allow further investigation of invalidation scenarios. Ultimately, building on such results, a simulator relying on flow-level models should warn its users when it wanders outside the validity domain of its simulation models.

Acknowledgments

We warmly thank the editor in chief, the associate editor and all the reviewers for their thorough work and for all their comments that helped us improving our article.

References

ALEXANDROV, A., IONESCU, M., SCHAUSER, K., AND SCHEIMAN, C. 1995. LogGP: Incorporating Long Messages into the LogP Model – One Step Closer Towards a Realistic Model for Parallel Computation. In *ACM Symposium on Parallel Algorithms and Architectures (SPAA)*.

BARABÁSI, A. AND ALBERT, R. 1999. Emergence of scaling in random networks. *Science 59*, 509–512.

BAUMGART, I., HEEP, B., AND KRAUSE, S. 2009. OverSim: A Scalable and Flexible Overlay Framework for Simulation and Real Network Applications. In *Proc. of the 9th Int. Conference on Peer-to-Peer Computing*.

BELL, W. H., CAMERON, D. G., MILLAR, A. P., CAPOZZA, L., STOCKINGER, K., AND ZINI, F. 2003. OptorSim: A grid simulator for studying dynamic data replication strategies. *International Journal of High Performance Computing and Applications 17,* 4.

BERTSEKAS, D. P. AND GALLAGER., R. 1992. *Data Networks*. Prentice-Hall.

BLYTHE, J., JAIN, S., DEELMAN, E., GIL, Y., VAHI, K., AND ET AL. 2005. Task scheduling strategies for workflow-based applications in grids. In *IN CCGRID*. IEEE Press, 759–767.

BRAUN, T. D., SIEGEL, H. J., BECK, N., BÖLÖNI, L. L., MAHESWARAN, M., REUTHER, A. I., ROBERTSON, J. P., THEYS, M. D., YAO, B., HENSGEN, D., AND FREUND, R. F. 2001. 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,* 6, 810–837.

BUYYA, R. AND MURSHED, M. 2002. GridSim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. *The Journal of Concurrency and Computation: Practice and Experience (CCPE) 14(13)*.

CALHEIROS, R. N., RANJAN, R., BELOGLAZOV, A., DE ROSE, C. A. F., AND BUYYA, R. 2011. CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms. *Software: Practice and Experience 41,* 1, 23–50.

CASANOVA, H. 2001. SimGrid: A toolkit for the simulation of application scheduling. In *First IEEE International Symposium on Cluster Computing and the Grid (CCGrid'01)*. Brisbane, Australia.

CASANOVA, H., LEGRAND, A., AND MARCHAL, L. 2003. Scheduling distributed applications: the SimGrid simulation framework. In *Proceedings of the Third IEEE International Symposium on Cluster Computing and the Grid (CCGrid'03)*. IEEE Computer Society Press.

CASANOVA, H., LEGRAND, A., AND QUINSON, M. 2008. SimGrid: a generic framework for large-scale distributed experiments. In *Proceedings of the 10th Conference on Computer Modeling and Simulation (EuroSim'08)*.

CASANOVA, H. AND MARCHAL, L. 2002. A network model for simulation of grid application. Tech. Rep. 2002-40, LIP. Oct.

CHEN, Q., CHANG, H., GOVINDAN, R., AND JAMIN, S. 2002. The origin of power laws in Internet topologies revisited. In *INFOCOM*. 608–617.

CHEN, W. AND DEELMAN, E. 2012. Workflowsim: A toolkit for simulating scientific workflows in distributed environments. In *8th IEEE International Conference on eScience*. IEEE.

CHIU, D. N. 1999. Some observations on fairness of bandwidth sharing. Tech. rep., Sun Microsystems.

CLAUSS, P.-N., STILLWELL, M., GENAUD, S., SUTER, F., CASANOVA, H., AND QUINSON, M. 2011. Single node on-line simulation of MPI applications with smpi. In *25th IEEE International Parallel and Distributed Processing Symposium (IPDPS'11)*. Anchorage (Alaska) USA.

CULLER, D., KARP, R., PATTERSON, D., SAHAY, A., SCHAUSER, K., SANTOS, E., SUBRAMONIAN, R., AND VON EICKEN, T. 1993. LogP: Towards a Realistic Model of Parallel Computation. In *ACM SIGPLAN Symp. on Principles and Practice of Parallel Programming*.

DABEK, F., COX, R., KAASHOEK, F., AND MORRIS, R. 2004. Vivaldi: A Decentralized Network Coordinate System. In *ACM SIGCOMM*.

DE CNODDER, S., ELLOUMI, O., AND PAUWELS, K. 2000. Red behavior with different packet sizes. In *Proceedings of the Fifth IEEE Symposium on Computers and Communications (ISCC 2000)*. ISCC '00. IEEE Computer Society, Washington, DC, USA.

DOAR, M. B. 1996. A better model for generating test networks. In *IEEE GLOBECOM*. 86–93.

FALOUTSOS, M., FALOUTSOS, P., AND FALOUTSOS, C. 1999. On power-law relationships of the internet topology. In *SIGCOMM*. 251–262.

FLOYD, S. AND FALL, K. 1999. Promoting the use of end-to-end congestion control in the Internet. *IEEE/ACM Transactions on Networking 7,* 4, 458–472.

FLOYD, S. AND JACOBSON, V. 1992. On traffic phase effects in packet-switched gateways. *Internetworking: Research and Experience 3*, 115–156.

FLOYD, S. AND JACOBSON, V. 1993. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking 1,* 4.

FUJIWARA, K. AND CASANOVA, H. 2007. Speed and accuracy of network simulation in the simgrid framework. In *Proceedings of the 2nd international conference on performance evaluation methodologies and tools*. 1–10.

GIL, T. M., KAASHOEK, F., LI, J., MORRIS, R., AND STRIBLING, J. 2005. P2PSim: a simulator for peer-to-peer protocols. http://pdos.csail.mit.edu/p2psim/.

GIULI, T. AND BAKER, M. 2002. Narses: A Scalable Flow-Based Network Simulator. Tech. Rep. cs.PF/0211024, Stanford University. Available at http://arxiv.org/abs/cs.PF/0211024.

HEUSSE, M., MERRITT, S. A., BROWN, T. X., AND DUDA, A. 2011. Two-way TCP Connections: Old Problem, New Insight. *ACM Computer Communication Review*.

HOEFLER, T., SCHNEIDER, T., AND LUMSDAINE, A. 2010. LogGOPSim - Simulating Large-Scale Applications in the LogGOPS Model. In *Proc. of the 2nd Workshop on Large-Scale System and Application Performance*.

INO, F., FUJIMOTO, N., AND HAGIHARA, K. 2001. LogGPS: a parallel computational model for synchronization analysis. In *Proc. of the 8th ACM SIGPLAN symposium on Principles and practices of parallel programming*.

ISSARIYAKUL, T. AND HOSSAIN, E. 2008. *Introduction to Network Simulator NS2*. Springer Link.

JACOBSSON, K., ANDREW, L., TANG, A., JOHANSSON, K., HJALMARSSON, H., AND LOW, S. 2008. Ack-clocking dynamics: Modelling the interaction between windows and the network. In *INFOCOM*.

JAIN, M., PRASAD, R. S., AND DOVROLIS, C. 2003. The TCP bandwidth-delay product revisited: network buffering, cross traffic, and socket buffer auto-sizing. Tech. Rep. GIT-CERCS-03-02, Georgia Institute of Technology.

JANSEN, S. AND MCGREGOR, A. 2007. Validation of simulated real world TCP stacks. In *Winter Simulation Conference*.

JUNG, J. AND KIM, H. 2012. Mr-cloudsim: Designing and implementing mapreduce computing model on cloudsim. In *International Conference on ICT Convergence (ICTC)*. 504 –509.

KELLY, F., MAULLOO, A., AND TAN, D. 1998. Rate control for communication networks: Shadow prices, proportional fairness and stability. *Journal of the Operational Research Society 49,* 3.

KIELMANN, T., BAL, H., AND VERSTOEP, K. 2000. Fast Measurement of LogP Parameters for Message Passing Platforms. In *Proc. of the 4th Work. on Run-Time Systems for Parallel Programming*.

LAKHINA, A., BYERS, J., CROVELLA, M., AND XIE, P. 2003. Sampling biases in ip topology measurements. In *INFOCOM*.

LEDLIE, J., GARDNER, P., AND SELTZER, M. 2007. Network Coordinates in the Wild. In *Proc. of the 4th Symposium on Networked Systems Design and Implementation (NSDI)*.

LOW, S. H. 2003. A duality model of TCP and queue management algorithms. *IEEE/ACM Transactions on Networking 11,* 4.

LOW, S. H., PETERSON, L. L., AND WANG, L. 2002. Understanding vegas: a duality model. *Journal of the ACM 49,* 2.

LOW, S. H. AND SRIKANT, R. 2004. A mathematical framework for designing a low-loss, low-delay internet. *Network and Spatial Economics 4*, 75–102.

MARFIA, G., PALAZZI, C., PAU, G., GERLA, M., SANADIDI, M., AND ROCCETTI, M. 2007. Tcp libra: Exploring rtt-fairness for tcp. In *NETWORKING 2007. Ad Hoc and Sensor Networks, Wireless Networks, Next Generation Internet*, I. F. Akyildiz, R. Sivakumar, E. Ekici, J. C. d. Oliveira, and J. McNair, Eds. Lecture Notes in Computer Science Series, vol. 4479. Springer Berlin Heidelberg, 1005–1013.

MATHIS, M., SEMKE, J., AND MAHDAVI, J. 1997. The macroscopic behavior of the TCP congestion avoidance algorithm. *Computer Communications Review 27,* 3.

MEDINA, A., LAKHINA, A., MATTA, I., AND BYERS, J. 2001. BRITE: An approach to universal topology generation. In *International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems - MASCOTS'01*. Cincinnati, Ohio.

MO, J., LA, R., ANANTHARAM, V., AND WALRAND, J. 1999. Analysis and comparison of tcp reno and tcp vegas. In *INFOCOM*.

MO, J. AND WALRAND, J. 2000. Fair end-to-end window-based congestion control. *IEEE/ACM Transactions on Networking 8,* 5.

MONTRESOR, A. AND JELASITY, M. 2009. PeerSim: A scalable P2P simulator. In *Proc. of the 9th Int. Conference on Peer-to-Peer Computing*.

NS3. 2011. The Network Simulator 3: http://www.nsnam.org/.

NÚÑEZ, A., VÁZQUEZ-POLETTI, J., CAMINERO, A., CARRETERO, J., AND LLORENTE, I. M. 2011. Design of a New Cloud Computing Simulation Platform. In *Proc of the 11th International Conference on Computational Science and its Applications*.

OSTERMANN, S., PRODAN, R., AND FAHRINGER, T. 2010. Dynamic Cloud Provisioning for Scientific Grid Workflows. In *Proc. of the 11th ACM/IEEE Intl. Conf. on Grid Computing (Grid)*.

PENTIKOUSIS, K. 2001. Connector: active queue management. *Crossroads 7,* 5.

POPPER, K. 1972. *Objective Knowledge: An Evolutionary Approach*. Oxford University Press.

RAMASWAMY, S. AND BANERJEE, P. 1993. Processor allocation and scheduling of macro dataflow graphs on distributed memory multicomputers by the paradigm compiler. In *In Proceedings of the 1993 International Conference on Parallel Processing, volume II-Software*. CRC Press, 134–138.

RILEY, G. F. 2003. The georgia tech network simulator. In *ACM SIGCOMM workshop on Models, Methods and Tools for Reproducible Network Research*. Karlsruhe, Germany, 5–12.

SCHNORR, L., LEGRAND, A., AND VINCENT, J.-M. 2011. Detection and analysis of resource usage anomalies in large distributed systems through multi-scale visualization. *Concurrency and Computation: Practice and Experience.*.

SCHNORR, L. M., HUARD, G., AND NAVAUX, P. O. A. 2010. Triva: Interactive 3D visualization for performance analysis of parallel applications. *Future Generation Computer Systems 26,* 3, 348–358.

SHI, Y., JIANG, X., AND YE, K. 2011. An energy-efficient scheme for cloud resource provisioning based on cloudsim. In *IEEE International Conference on Cluster Computing (CLUSTER)*. 595 –599.

TANG, A., ANDREW, L., JACOBSSON, K., JOHANSSON, K., HJALMARSSON, H., AND LOW, S. 2010. Queue Dynamics With Window Flow Control. *IEEE/ACM Transactions on Networking 18,* 5.

TANG, A., ANDREW, L., JACOBSSON, K., JOHANSSON, K., LOW, S., AND HJALMARSSON, H. 2008. Window flow control: Macroscopic properties from microscopic factors. In *INFOCOM*.

TANGMUNARUNKIT, H., GOVINDAN, R., JAMIN, S., SHENKER, S., AND WILLINGER, W. 2002. Network topology generators: Degree-based vs structural. In *SIGCOMM*.

TENG, F., YU, L., AND MAGOULÈS, F. 2011. Simmapreduce: A simulator for modeling mapreduce framework. In *5th FTRA International Conference on Multimedia and Ubiquitous Engineering (MUE)*. 277 –282.

TOPCUOGLU, H., HARIRI, S., AND WU, M.-Y. 1999. Task scheduling algorithms for heterogeneous processors. In *Eighth Heterogeneous Computing Workshop*. IEEE Computer Society Press.

TRIVA. 2011. http://triva.gforge.inria.fr.

VARGA, A. AND HORNIG, R. 2008. An overview of the omnet++ simulation environment. In *Simutools*.

VELHO, P. AND LEGRAND, A. 2009. Accuracy study and improvement of network simulation in the simgrid framework. In *Proceedings of the 2nd International Conference on Simulation Tools and Techniques*.

WAXMAN, B. M. 1988. Routing of multipoint connections. *IEEE Journal on Selected Areas in Communications 6,* 9, 1617–1622.

YAÏCHE, H., MAZUMDAR, R. R., AND ROSENBERG, C. 2010. A Game Theoretic Framework for Bandwidth Allocation and Pricing in Broadband Networks. *IEEE/ACM Transactions on Networking 8,* 5.

ZHANG, L., SHENKER, S., AND CLARK, D. D. 1991. Observations on the dynamics of a congestion control algorithm: The effects of two-way traffic. In *ACM Computer Communication Review*. 133–147.

ZHENG, G., KAKULAPATI, G., AND KALÉ, L. V. 2004a. BigSim: A parallel simulator for performance prediction of extremely large parallel machines. In *Proc. of IPDPS*.

ZHENG, G., WILMARTH, T., LAWLOR, O. S., KALÉ, L. V., ADVE, S., AND PADUA, D. 2004b. Performance modeling and programming environments for petaflops computers and the blue gene machine. In *Computer Science*. IEEE Press.