Balancing Accuracy and Efficiency in TCP Flow Simulation

Joel Sommers
Department of Computer Science
Colgate University
Hamilton, NY, USA
jsommers@colgate.edu

YeonJu Mok
Department of Computer Science
Colgate University
Hamilton, NY, USA
ymok@colgate.edu

***Abstract*—Efficient and accurate network simulation techniques are critical for evaluating protocols and systems *at scale*. For example, the magnitude of modern data center deployments requires the use of fast simulation techniques in order to evaluate newly proposed algorithms and system architectures. Unfortunately, efficient simulation can come at the cost of accuracy and realism. In this work-in-progress paper, we examine specific limitations of existing closed-form models of TCP throughput, which are commonly used to simulate aggregate performance of TCP flows. We evaluate two models in particular, comparing predictions of the models with actual performance measured using the Mininet platform. We find that the open-loop nature of these models and sensitivity to different parameters contributes to significant inaccuracies, which may in turn lead to incorrect conclusions when using such models as the basis for simulation. We describe our ongoing work on a new method for scalable TCP flow simulation that is based on ideas from XCP. Our proposed technique is highly efficient, incorporates network feedback in a closed-loop manner, and in our initial experiments shows appreciable improvement in accuracy over prior models.**

I. Introduction

Network simulation platforms are critical elements in the effective evaluation of new protocols, network architectures, and applications. Simulation is especially important for protocols and applications designed for deployment in very large scale environments. In such settings, it is important for the simulation system to be highly *efficient* in order to be able to effectively evaluate a range of scenarios in a timely manner. Moreover, it is important for the simulation system to be *accurate*, in order for the results gleaned from simulation to be meaningfully predictive of expected performance in a live deployment.

In simulators designed to evaluate large-scale scenarios such as in data center networking studies, closed-form or fluid-flow models of TCP throughput and delay are commonly used to imitate the behavior of TCP flows, *e.g.*, see [1], [2]. Closed-form models vary in complexity from the simple and widely-used model developed by Mathis *et al.* [3] (abbreviated as MSMO97), which only considers the congestion avoidance phase of TCP, to more complex models that consider various phases of TCP, *e.g.*, the model by Cardwell *et al.* [4] (abbreviated as CSA00). Other simulation systems use fluid-flow models (*e.g.* [5]), which simulate flows in an aggregated, abstract manner. The key advantage of these simulation methods is their efficiency, which is achieved mainly because detailed packet-level behaviors are ignored. Indeed, several recent studies in data center networking have used fluid-flow models, simply because no other method is as scalable. On the other hand, packet-level simulation systems like ns-3 [6] offer more fine-grained simulation, but are typically too slow for large-scale simulations because of the detail at which the simulation takes place.

In this work-in-progress paper, we examine the tradeoff between speed and accuracy in simulation of TCP flows. Specifically, we examine limitations of two closed-form models of TCP throughput behavior in the context of the *fs* flow-level simulator [7]. We compare predictions of the models with actual performance measured using the Mininet platform [8]. We find that three key factors lead to significant inaccuracies in the results generated from simulations that rely on these models, which could lead to incorrect conclusions. Specifically, we find that (1) the open-loop nature of these models is a significant factor in simulation scenarios in which there is network congestion, (2) similar to findings of other researchers [9], the models are sensitive to parameter settings, which can lead to errors, and (3) the models assume a much lower frequency of packet loss recovery through retransmission timeouts (as opposed to through "fast recovery"), which leads to errors in estimation of flow durations.

We describe our ongoing work on a new method for efficient and scalable discrete-event flow-level simulation that accommodates closed-loop feedback from the network, and as a consequence offers significant accuracy improvements in congested network scenarios. Our approach is based on the ideas of the eXplicit Control Protocol (XCP) [10], in which routers provide explicit feedback to sending nodes in order to modulate sending rates and efficiently share network bandwidth. Our technique achieves a balance between efficiency and accuracy by modeling the *effects* of a TCP flow through modeling its AIMD nature in a closed-loop manner rather than the detailed *operation* of TCP, as is done with packet-level simulation.

A. Background and Related Work

The design of network simulation platforms involves tradeoffs between detailedness, efficiency, scalability, and faithfulness to the way that real systems operate. On one end of the spectrum, packet-level simulators such as ns-3 [6] make explicit decisions to simulate detailed packet-level network behaviors at the cost of being able to scale to very large networks. Although these systems can be made to execute simulation scenarios reasonably quickly (*e.g.*, through parallelization), they have inherent scalability and efficiency problems because of the low level of abstraction (the packet) at which they operate.

In contrast, *flow-level simulators* attempt to model higher-level abstractions—network flows between two endpoints. Since they are not geared to simulate packet-level interactions, they cannot accurately replicate some network behaviors. They can, however, scale much better than packet-based simulators and still accurately reflect aggregate network behavior. One approach in flow-level simulation has been based on fluid-flow modeling [5]. This approach has been used extensively in large-scale network studies such as data center experiments, see *e.g.*, [1], [2]. Although fluid-flow simulations can be made to operate in a closed-loop manner, Liu *et al.* observed that in congested network scenarios the computational overheads of fluid-flow simulation can be *greater* than packet-level simulation [11]. Thus, they are typically used in an open-loop manner to realize their potential performance benefits.

Another flow-level simulation approach is based on using analytic models of TCP performance[3], [4]. This approach also leads to highly scalable simulation, but the results derived from these platforms are dependent on the accuracy of the underlying TCP models used. Moreover, flows simulated using both (typical) fluid-flow and closed-form models are *non-reactive* to network conditions, unlike real TCP implementations. To compensate for this issue, the authors in [12] use the MSMO97 model in a piece-wise manner in order to use network feedback. In [9], the authors examine the accuracy of Mathis-like TCP models, comparing live traces with what the model would predict. They find significant inaccuracies, specifically with respect to sensitivity to loss and delay.

II. Limits of Closed-form TCP Models

In this section, we describe experiments we carried out to better understand the limits and sensitivities of closed-form TCP throughput models.

A. Experiments

Our approach was to set up identical environments in the *fs* simulator, which has capabilities to use either the MSMO97 model or the CSA00 model, and in the Mininet emulator [8]. In each environment, we created a simple "dumbbell"-like topology, with a shared bottleneck link connecting traffic sources and traffic sinks. We used the Harpoon traffic generator [13] in Mininet to generate "real" traffic and the Harpoon-like traffic generation capabilities in *fs* to create similar conditions in the simulator. We used identical configurations in each environment to create an Internet-like mixture of flow sizes drawn from a heavy-tailed (Pareto) distribution. All experiments lasted for 900 (wall-clock, in the case of Mininet, or simulated, in the case of *fs*) seconds.

We used a set of configurations to emulate a range of round-trip times and loss conditions on the bottleneck link. We used three one-way propagation delay settings of 25, 50, and 100 milliseconds, resulting in round-trip times of approximately 50, 100, and 200 milliseconds. We used three different "imposed" loss rates on the bottleneck link (*i.e.,* the bottleneck queue was configured to randomly drop some proportion of packets) of 0.5%, 1% and 5%. In each setting, we used two different TCP receive-window values (16KB and 1MB) to emulate both a mix of congestion-window and receive-window-limited flows. In Mininet, we ran experiments using two different settings of the TCP congestion control algorithm: NewReno and CUBIC. In total, our experiment setup resulted in 36 separate scenarios in the Mininet environment and 27 separate scenarios in the *fs* environment.

In each setting, we collected measurements for each TCP flow, including its start and end times, as well as counters of bytes, packets, and active flows per second. *fs* can be easily configured to collected these measurements; in Mininet, we used CAIDA's CoralReef toolset to collect the same data.

B. Results

We now discuss results from our experiments, focusing on comparisons of *flow completion time* in comparable Mininet and *fs* experiments. We also examined scaling behavior and other metrics, but due to space constraints we do not discuss results from those analyses in this paper.

At a high level, our results show that there are significant differences between flow completion times measured in Mininet as compared with *fs*, which, again, relies on TCP throughput models to drive its flow-level simulation. First, similar to observations made by the authors in [9], we find that results derived from the TCP models are more sensitive to changes in round-trip time (delay) than loss, at least for modest and, arguably, realistic levels of packet loss.

Second, we find that flows in Mininet exhibit much higher variability in completion time. In many ways, this result should not be surprising: since closed-form TCP models are deterministic and, by their very nature, cannot incorporate network feedback, they should exhibit no variability whatsoever. On the other hand, our results suggest that while the simplicity and efficiency of these models makes them appealing for use in simulation, results derived from their use may be limited in scope. To illustrate this particular result, we show in Figure 1 scatterplots of flow size versus flow completion time for *fs* and Mininet in three separate scenarios (notice that the axes in the plots are log-log scale). The top plot shows results using a 100 millisecond round-trip time with 0.5% loss rate on the bottleneck link, and with *fs* using the MSMO97 model. The middle plot shows results using the CSA00 model with a 100 millisecond RTT, a loss rate of 1% on the bottleneck link, and a TCP receive window of 16 KB. The bottom plot shows results with the CSA00 model with a 50 millisecond RTT and a loss rate of 0.5% on the bottleneck link, and a receive window of 1 MB. Note that the MSMO97 model does not consider the TCP receive window; in all three experiments, the receive window configured for TCP in Mininet is 1MB.

From the plots in Figure 1, we observe that with the MSMO97 model (top plot), durations of short flows are significantly *underestimated* and that the flow durations are, overall, a poor match for measurements collected using Mininet. Recall that the MSMO97 model only considers the congestion-avoidance phase of TCP and that short flows using real TCP implementations spend most or all of their lifetime in the slow-start phase. For the plots using the CSA00 model (middle and bottom plots), we observe that while the estimation is improved, especially for short flows, the model generally *overestimates* flow durations. Since the plot is drawn on log-log scale, the inaccuracies are significant. In results with longer RTT and higher loss, we observe that the CSA00 model can also *underestimate* flow completion times in a significant way. Our hypothesis for why this underestimation occurs is that in higher loss and delay situations the model underestimates the frequency of loss recovery through retransmission timeout (RTO). Measurements we have collected provide some support for this hypothesis, and in our ongoing work we are continuing to examine this issue. We also note that work by Flach *et al.* to examine traces collected in the live Internet shows that loss recovery through RTO occurs in the Internet much more frequently than commonly assumed [14]. While differences in a model of TCP behavior from an actual implementation (*e.g.*, acking behavior, etc.) may explain *some* of the differences we observed between Mininet and *fs*, we argue that since models cannot accommodate network feedback (*e.g.*, packet loss and queuing delay) "tweaking" the model by including additional parameters is not likely to lead to better accuracy. We further argue that an approach such as that by [12] to incorporate network feedback into an existing TCP throughput model is not likely to significantly improve accuracy because of weaknesses inherent in existing analytic models.

III. Proposed Flow Generation Approach

Results from the previous section motivate a new approach for flow-level simulation of TCP. The goals of our ongoing work are to develop a technique that (1) is closed loop,

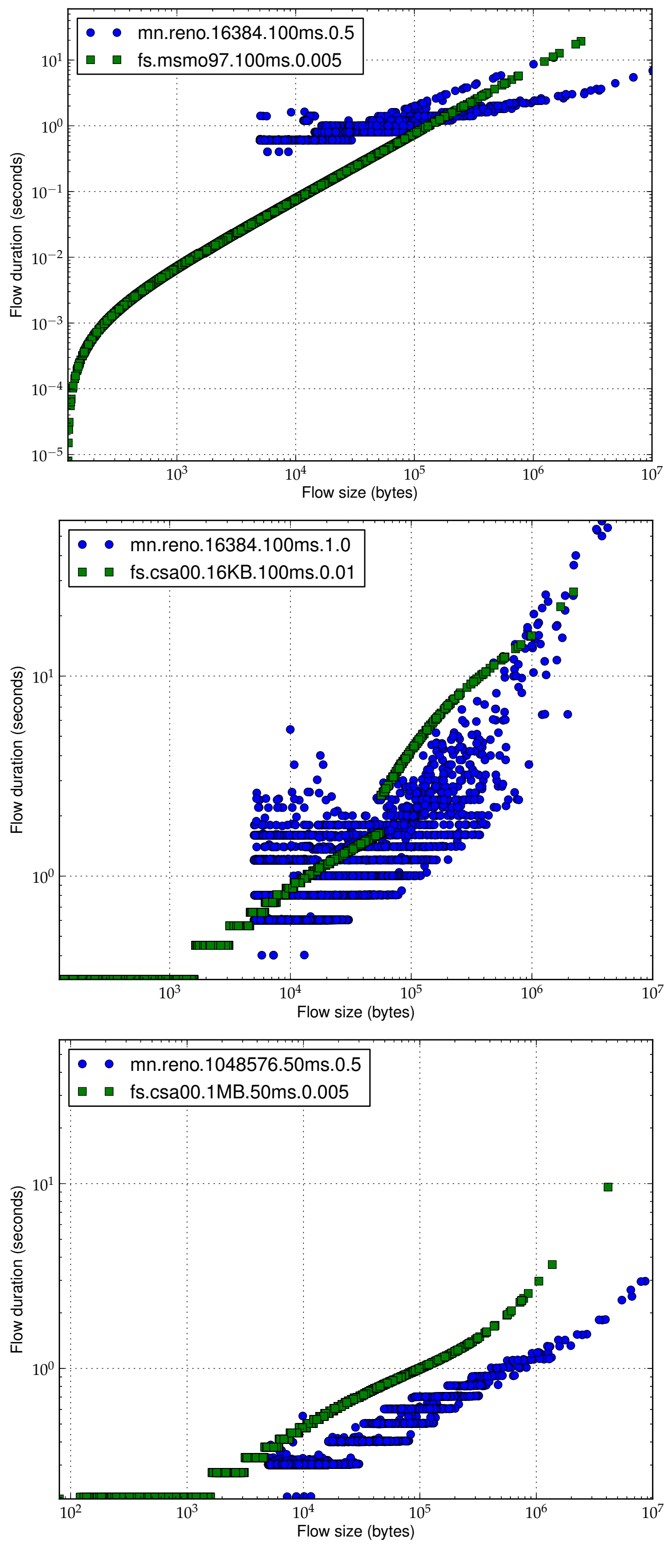

Figure 1. Scatterplots (log-log scale) of flow completion time versus flow size for *fs* using the MSMO97 model (top) and *fs* using the CSA00 model (middle and bottom) in three separate network configurations.

thus permitting accurate simulation of congested network scenarios, (2) enables use of arbitrary flow sizes, allowing empirically measured flow sizes from live environments to be used, and (3) is highly efficient and thus able to scale to large scenarios. We argue that no existing approach satisfies these requirements: packet-level simulators cannot scale to the degree necessary, fluid-flow simulation cannot easily accommodate arbitrary flow sizes, closed-form TCP models are not accurate enough, and neither fluid-flow (in its most efficient form) nor closed-form models are closed-loop.

Our proposed approach is based on ideas used in the eXplicit Control Protocol (XCP) [10]. Specifically, routers and switching nodes in the network explicitly convey information back to senders about how to modulate their emitted traffic stream so as to efficiently and fairly share end-to-end bandwidth. The main motivation for implementing an XCP-like design is that centralizing the congestion control logic and using explicit feedback at routers offers efficiency gains over implementing TCP-like logic at individual flow endpoints since XCP maintains no per-flow state. Also, the XCP algorithm separately considers efficiency and fairness, thus enabling a simulation environment to implement bandwidth sharing that mimics the way TCP works (*i.e.*, to implement TCP's bias against long RTT flows). The sender algorithm is simple, and lends itself to scalable flow-based simulation.

We have implemented an initial version of our XCP-like algorithm in the *fs* simulator. Our preliminary results show that the technique leads to significantly improved accuracy in congested scenarios over using closed-form TCP models. Furthermore, while there is some performance degradation with our new approach, our initial evaluations show that the difference in number of simulation events—which directly relates to simulation performance—is less than 10%. The original version of *fs* used a particular technique for handling congestion and queuing at routers (in some ways, similar to [12]) which caused an increase in simulation events. Thus our new technique appears, so far, to yield far better accuracy at little performance cost. Our current efforts are focused on refining our implementation, and evaluating it over a broad set of network scenarios.

Acknowledgments

Thanks to Martin Liu (Colgate '16) and Jack Sneeringer (Colgate '16) for their comments and assistance on this work. Thanks also to the anonymous reviewers for their helpful comments.

This material is based upon work supported by the National Science Foundation under grant CNS-1054985. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.

References

[1] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: Dynamic flow scheduling for data center networks," in *Proceedings of USENIX NSDI '10*, 2010.

[2] B. Stephens, A. Cox, W. Felter, C. Dixon, and J. Carter, "PAST: scalable Ethernet for data centers," in *Proceedings of ACM CoNeXT '12*, 2012.

[3] M. Mathis, J. Semke, J. Mahdavi, and T. Ott, "The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm," *ACM SIGCOMM Computer Communication Review*, vol. 27, no. 3, p. 82, 1997.

[4] N. Cardwell, S. Savage, and T. Anderson, "Modeling TCP Latency," in *Proceedings of IEEE INFOCOM '00*, Tel Aviv, Israel, March 2000.

[5] V. Misra, W.-B. Gong, and D. Towsley, "Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED," in *ACM SIGCOMM Computer Communication Review*, vol. 30, no. 4, 2000.

[6] "The ns-3 network simulator," http://www.nsnam.org, 2014.

[7] J. Sommers, R. Bowden, B. Eriksson, P. Barford, M. Roughan, and N. Duffield, "Efficient network-wide flow record generation," in *Proceedings of INFOCOM '11*, April 2011.

[8] N. Handigol, B. Heller, V. Jeyakumar, B. Lantz, and N. McKeown, "Reproducible network experiments using container-based emulation," in *Proceedings of ACM CoNeXT '12*, 2012.

[9] R. G. Clegg, J. T. Araújo, R. Landa, E. Mykoniati, D. Griffin, and M. Rio, "On the relationship between fundamental measurements in TCP flows," in *IEEE ICC*, June 2013.

[10] D. Katabi, M. Handley, and C. Rohrs, "Congestion control for high bandwidth-delay product networks," in *ACM SIGCOMM Computer Communication Review*, vol. 32, no. 4, 2002.

[11] B. Liu, D. R. Figueiredo, Y. Guo, J. Kurose, and D. Towsley, "A study of networks simulation efficiency: Fluid simulation vs. packet-level simulation," in *Proceedings of IEEE INFOCOM '01*, 2001.

[12] T. Li, N. Van Vorst, and J. Liu, "A rate-based tcp traffic model to accelerate network simulation," *Simulation*, vol. 89, no. 4, 2013.

[13] J. Sommers and P. Barford, "Self-Configuring Network Traffic Generation," in *Proceedings of ACM Internet Measurement Conference*, October 2004.

[14] T. Flach, N. Dukkipati, A. Terzis, B. Raghavan, N. Cardwell, Y. Cheng, A. Jain, S. Hao, E. Katz-Bassett, and R. Govindan, "Reducing web latency: the virtue of gentle aggression," in *Proceedings of the ACM SIGCOMM '13*, 2013.