Proceedings of the 2006 Winter Simulation Conference
L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, eds.

PACKET-LEVEL INTEGRATION OF FLUID TCP MODELS IN REAL-TIME NETWORK SIMULATION

Jason Liu

Department of Mathematical and Computer Sciences
Colorado School of Mines
Golden, CO 80401, U.S.A.

ABSTRACT

We present a hybrid network traffic model that combines time-stepping fluid model with discrete-event packet-oriented simulation. We propose an integration scheme allowing packet flows to interact with fluid flows within each network queue. Different from previous schemes that require physical division of the virtual network between the fluid model and packet-oriented simulation, our hybrid model allows full integration of the two paradigms making it possible to dynamically change the composition of traffic flows to allow the simulation to keep up with real time. Experiments show that our model provides a good prediction of the network behavior. More important, as we increase the proportion of packet flows, the simulation is capable of capturing more detail of the network traffic behavior at the expense of more computing time. Hence the tradeoff.

1 INTRODUCTION

Simulation has been used extensively as an effective tool for studying, evaluating, and prototyping large-scale computer networks despite its well-documented difficulties in rendering realistic global network behaviors (Floyd and Paxson 2001). The networking research community has recently seen a healthy growth in the capability of simulating and modeling large-scale networks (e.g., Fujimoto et al. 2003, Nicol et al. 2003). This capability enables real-time simulation. Real-time simulation allows the simulation system to interact with the physical world in real time (Nicol, Liljenstam, and Liu 2005). In such systems, one must allow the network traffic from real applications to go through the simulator and interact with any arbitrary virtual network entities (such as hosts, routers, and links). Since real applications operate in real time, a real-time network simulation must meet real-time requirements. In essence, the performance of a large-scale network simulation must be able to keep up with the wall-clock time.

The difficulty in real-time network simulation is to ensure processing real-time events before their deadlines. Failure to process an event at the designated real-time deadline indicates a timing fault. An elevated occurrence of timing faults can cause a simulator to become "sluggish"—making it less responsive to real-time interactions. To speed up simulation, on the one hand, we can leverage parallel and distributed discrete-event simulation to harness the computing resources of parallel computers to physically increase the event-processing power (Fujimoto 2000). On the other hand, we resort to multi-resolution modeling techniques using models with high levels of abstraction to induce less computational demand. In this paper, we proposes a solution for integrating a fluid traffic model with the discrete-event model in our packet-oriented network simulator.

Fluid network simulation capitalizes on the notion that one can model network traffic and network transactions as fluid flows rather than individual packets. In a discrete-event fluid model (e.g., Nicol, Goldsby, and Johnson 1999), events are used to indicate changes in the flow rate as they propagate through the virtual network. One advantage of this approach is that there exists a natural way to integrate events of the fluid model with those of the packet-driven simulation and therefore create a hybrid simulation (see, for example, Kiddle et al. 2003 and Riley et al. 2002). Conceptually, this gives rise to an important distinction between packet-oriented foreground traffic and fluid-based background traffic. The foreground traffic is the traffic stream we are interested in and needs to be modeled in high fidelity. In contrast, the background traffic that represents the bulk of the network traffic is of secondary interest and does not require significant accuracy. It is nonetheless important as the background traffic interferes with the foreground traffic as both compete for network resources (Melamed, Pan, and Wardi 2004). It has been reported that, using proper smoothing techniques to avoid the dramatic increase of simulation events (so-called "ripple effect"), the event-oriented fluid models can achieve a desirable modeling accuracy and a speedup of approximately

1-4244-0501-7/06/$20.00 ©2006 IEEE

an order of magnitude over pure-packet simulations (Nicol and Yan 2004).

Alternative to discrete-event fluid simulation are analytical models. Misra, Gong, and Towsley (2000) proposed a fluid-based model that uses a set of ordinary differential equations (ODE) to describe the behavior of persistent TCP flows. These differential equations can be solved numerically with great computational efficiency. Liu et al. (2004) later improved this model by incorporating idiosyncratic network topology information and formulated a set of delayed differential equations solvable using the time-stepped Runge-Kutta algorithm. Experimental results show that this model is accurate compared to the packet-level simulation and its run-time performance scales well to a large number of TCP flows.

One potential problem of the analytical-based fluid model is the difficulty of integrating with the discrete-event packet-oriented simulation. The fluid model and the packet model are based on two distinct simulation paradigms. The fluid model describes the network and the traffic flows within as a set of differential equations with state variables inherently continuous in time. In contrast, packet flows are described using discrete events. Our interest here is to integrate fluid-based traffic flows with discrete-event packet flows and place them under the same time management scheme.

One effort with a goal similar to ours was reported by Gu, Liu, and Towsley (2004). In their approach, the simulation maintains two separate networks: a fluid network representing the core of the global network whose state evolution is dictated by the solution of differential equations, and a packet network where network transactions are modeled as discrete events. The interaction happens at the boundary. Packets entering the fluid network are smoothed and transformed into a fluid flow (as a piece-wise constant function) and then compete with the pure fluid flows for network resources according to the set of differential equations. The packet delay and drop probability are computed cumulatively within the fluid network. Packets departing from the fluid network are scheduled according to the delay and drop probability. Another similar approach was proposed by Zhou et al. (2004). In order to achieve a better response time for the ODE solver (implemented in MATLAB) when processing packet events "traversing through" the fluid network, two instances of the same fluid model are included in their system with simulation clocks of the two models interleaved with each other. This effectively doubles the speed of execution of the ODE solver.

Both approaches physically separate fluid and packet traffic representations into two networks. This (geographical) division of the network must be done statically before simulation and must not be changed during simulation execution. Furthermore, in this framework, the fluid model is only used to model the end-to-end behavior of packet flows (such as delay and drop probability). The detailed state of the fluid network (such as the dynamics of the queue size of a router inside the fluid network) cannot be accessed directly to interact with the packet flows. After all, packet flows are treated only as cross-traffic in the fluid network. This scheme places a unnecessary restriction on where packets can be sent in the virtual network. In a true real-time network simulation scenario, one should be allowed to establish real network connections and therefore conduct real packet flows to any part of the virtual network.

We propose a different integration scheme. In our approach, the interaction of fluid and packet flows happens at the network interface within a router at any part of the virtual network. The pure fluid flows arriving at each network queue are augmented with packet flows arriving at the queue. The queuing dynamics are still governed by the set of differential equations only modified to include both packet flows and fluid flows. These differential equations are still solved using the fixed time-stepped Runge-Kutta method. Packets entering a network queue are simulated as discrete events; they should observe the delay and drop probability according to the state of the queue calculated by the Runge-Kutta method. The benefit of our approach can be summarized as follows:

1. The network is no longer divided between the fluid-based model and the packet-oriented model. The interaction between fluid flows and packet flows is resolved dynamically. This allows emulated network traffic (e.g., ICMP packets generated by a real host) to reach any part of the virtual network and interact seamlessly with the fluid flows.
2. The interaction between the fluid flows and packet flows happens at a much finer granularity. In the previous scheme, packet flows are "fluidized" as they enter the fluid network. In our approach, the packet flows remain in the packet format and only at times when interacting with fluid flows in a network queue are they "smoothed" into flow rates. As a consequence, the packet model can better maintain the desired packet-level precision (e.g., with respect to delay variations).
3. Our approach allows the modeler to dynamically make changes to the composition of network traffic between fluid and packet representations. This enables us to make the tradeoff between the accuracy of the network model and the real-time performance.

We prototyped our scheme in our PRIME SSFNet simulator developed at Colorado School of Mines specifically designed for real-time network simulation. Many network models in the simulator are inherited from the RINSE simulator from UIUC (Liljenstam et al. 2005). We evaluate

the accuracy and run-time performance of the hybrid model through preliminary simulation experiments. We find that our model provides accurate prediction of network behavior. More important, as we increase the proportion of packet flows in our hybrid model, the simulation is capable of capturing more detailed traffic behavior at the expense of more computing time. Hence the tradeoff.

The remainder of this paper is organized as follows. Section 2 provides the background of the fluid model. We discuss our extension to the fluid model to integrate with packet flows in Section 3. We conducted two sets of experiments to assess the accuracy and performance of our approach. The results are shown in Section 4. The conclusion and future work are discussed in Section 5.

2 THE FLUID MODEL

Our starting point is the fluid model developed by Gu, Liu, and Towsley (2004), which uses a set of ordinary differential equations to describe the mean traffic behavior of long-term TCP sessions in a network of Active Queue Management (AQM) routers. Here we briefly describe the model we adopted, in most part following the notations used in the original paper. In the next section, we extend this model to include packet flows.

In this model the network traffic is a collection of fluid classes. Each fluid class represents n_i homogeneous flows, having the same characteristics and following the same path from the same source to the same destination. The set of differential equations that control the system can be divided into three categories: those controlling the size of the congestion window of each fluid class, the state of each network queue, and the propagation of delay and loss probability inside the network.

For each TCP flow in fluid class i, the window size at time t, denoted by $W_i(t)$, can be described as

$$\frac{dW_i(t)}{dt} = \frac{1}{R_i(t)} - \frac{W_i(t)}{2} \cdot \lambda_i(t) \tag{1}$$

where $R_i(t)$ is the round-trip delay and $\lambda_i(t)$ is the packet loss rate at time t. This equation models TCP congestion window's additive-increase-multiplicative-decrease behavior in the congestion avoidance stage. Not included in this equation is the boundary condition that the size of the congestion window must stay between 0 and a prespecified maximum window size.

For each network link l, we denote C_l to be the bandwidth of the link and a_l to be the propagation delay. The size of the network queue l can be described using the following equation:

$$\frac{dq_l(t)}{dt} = \Lambda_l(t)(1 - p_l(t)) - C_l \tag{2}$$

where $\Lambda_l(t)$ is the aggregate arrival rate of all fluid classes traversing link l and $p_l(t)$ is the packet drop probability. $\Lambda_l(t) = \sum_{i \in N_l} A_i^l(t)$, where $A_i^l(t)$ is the instantaneous arrival rate of fluid class i, and N_l is the set of fluid classes traversing link l. Again, not included in this equation is the boundary condition that the queue length should be in the range between 0 and the maximum queue size Q_l.

For network queues with Random Early Detection (RED) queuing discipline, the packet dropping probability is based on the average queue size $x_l(t)$, which can be calculated from the instantaneous queue size $q_l(t)$ using the following differential equation:

$$\frac{dx_l(t)}{dt} = \frac{ln(1-\alpha)}{\delta} x_l(t) - \frac{ln(1-\alpha)}{\delta} q_l(t) \tag{3}$$

where α is the weight used in the RED Exponential Weighted Moving Average (EWMA) computation and δ is the step size.

The packet dropping probability $p_l(t)$ can then be computed from the average queue size $x_l(t)$. For example, we use the following piece-wise linear function for all our experiments described later in this paper:

$$p(x) = \begin{cases} 0 & 0 \le x < q^{min} \\ \frac{x - q^{min}}{q^{max} - q^{min}} p^{max} & q^{min} \le x < q^{max} \\ \frac{x - q^{max}}{q^{max}}(1 - p^{max}) + p^{max} & q^{max} \le x \le 2q^{max} \\ 1 & \text{otherwise} \end{cases} \tag{4}$$

where q^{min}, q^{max} and p^{max} are configurable parameters of the RED queue.

The arrival rate of fluid class i at the first queue s_i can be easily calculated as

$$A_i^{s_i}(t) = \frac{n_i W_i(t)}{R_i(t)}. \tag{5}$$

For subsequent hops of fluid class i, the arrival rate at the next queue $g_i(l)$ is simply the departure rate of the upstream queue after a time lag equal to the link's propagation delay a_l:

$$A_i^{g_i(l)}(t + a_l) = D_i^l(t). \tag{6}$$

The departure rate of queue l is associated with the arrival rate of queue l after a proper queuing delay. Let $t_f = t + q_l(t)/C_l$. If the aggregate arrival rate is less than the service capacity, the departure rate equals the arrival rate. Otherwise, it is proportional to the arrival rate as the service capacity is shared among the competing flows:

$$D_i^l(t_f) = \begin{cases} A_i^l(t)(1 - p_l(t)) & \text{if } \Lambda_l(t)(1 - p_l(t)) \le C_l \\ \frac{A_i^l(t)}{\Lambda_l(t)} C_l & \text{otherwise.} \end{cases} \tag{7}$$

By now, the only variables left are the round-trip time R_i and the packet drop rate λ_i for each fluid class i. They can be computed cumulatively along the path of each fluid class. Let $d_l^i(t)$ be the cumulative delay experienced by fluid class i at queue l and $b_i(l)$ be the predecessor (upstream) queue of queue l on the path of fluid class i.

$$d_l^i(t_f) = \begin{cases} \frac{q_l(t)}{C_l} & \text{if } l = s_i \\ d_{b_i(l)}^i(t - a_{b_i(l)}) + a_{b_i(l)} + \frac{q_l(t)}{C_l} & \text{otherwise.} \end{cases} \tag{8}$$

Similarly, let $r_l^i(t)$ be the cumulative packet drop rate experienced by fluid class i at queue l.

$$r_l^i(t_f) = \begin{cases} A_i^l(t)p_l(t) & \text{if } l = s_i \\ r_{b_i(l)}^i(t - a_{b_i(l)}) + A_i^l(t)p_l(t) & \text{otherwise.} \end{cases} \tag{9}$$

Assuming that ack loss and queuing delays are negligible on the reverse path and therefore considering only the traffic on the forwarding path, we can calculate the round-trip time and the loss rate:

$$R_i(t) = d_{f_i}^i(t - \pi_i) + \pi_i \tag{10}$$

$$\lambda_i(t) = \frac{r_{f_i}^i(t - \pi_i)}{n_i} \tag{11}$$

where π_i is the (one-way) path latency of fluid class i and f_i is the last queue traversed by fluid class i on the forwarding path.

Although this fluid model considers only TCP flows through RED queues, it can easily be extended to model a broader class of flows by changing the differential equation governing the sending window size (i.e., Equation (1)). Simple examples include fluid traffic for constant-rate UDP flows and on-off traffic generators using Pareto-Poisson Burst Processes (Zukerman, Neame, and Addie 2003). Note that this scheme actually provides a natural solution to the problem posed by Nicol and Yan (2005). In their approach, the (background) network traffic are induced by injecting aggregate flows as piece-wise constant rates. The authors formulate the problem as a set of nonlinear equations and they propose a solution using periodic fix-point computation. The problem can be solved here straightforwardly without imposing any change to the fluid model. Also, although the fluid model is designed for RED queues, we note that the model is more generic in that, if one sets $p_l(t) = (\Lambda_l(t) - C_l)/\Lambda_l(t)$ if the queue is full and $\Lambda_l(t) > C_l$, and zero otherwise, the RED queue is degenerated to a drop-tail queue.

In the next section, we extend the fluid model described in this section by adding packet flows.

3 PACKET-LEVEL INTEGRATION

In the simulation, packet flows traversing the network compete for network resources with the fluid traffic. Each packet arriving at a network queue is represented as a simulation event. The decision whether the packet should be admitted into the queue is based upon the state of the queue at the time of its arrival—specifically, the instantaneous queue size and the packet drop probability. The state of the queues is governed by the set of differential equations presented in the previous section. The fixed time-stepped Runge-Kutta method is used to calculate the time evolution of the state of the queues and the size of the congestion windows of all fluid classes. Suppose that δ is the time step size of the Runge-Kutta method. At each time step k, we compute the size of the congestion window $W_i(k\delta)$, the instantaneous queue length $q_l(k\delta)$, the average queue length $x_l(k\delta)$, and the packet dropping probability $p_l(k\delta)$. Also, for each fluid class i traversing network queue l, we update the fluid arrival rate A_i^l, the departure rate D_i^l, as well as the cumulative delay d_l^i and cumulative packet drop rate r_l^i.

To account for the packet flows arriving at the network queue, we keep track of the number of packets arriving at the queue since the last Runge-Kutta time step. Let $N_P^l(t)$ be the total number of packets arrived at queue l since beginning. The packet arrival rate is computed as:

$$A_P^l(k\delta) = \frac{N_P^l(k\delta) - N_P^l((k-1)\delta)}{\delta}. \tag{12}$$

The packet arrival is treated the same as fluid flows in terms of competing for queuing space. We modify Equation (2) to account for the packet arrival:

$$\frac{dq_l(t)}{dt} = \xi_l(t)(1 - p_l(t)) - C_l \tag{13}$$

where $\xi_l(t) = \Lambda_l(t) + A_P^l(t)$ is the aggregate arrival rate of both fluid and packet flows.

We also need to adjust the departure rate (Equation (7)) when the total arrival rate is more than the service capacity; the departure rate should be proportional to the arrival rate as the service capacity is shared among competing flows—whether they are fluid or packet.

$$D_i^l(t_f) = \begin{cases} A_i^l(t)(1 - p_l(t)) & \text{if } \xi_l(t)(1 - p_l(t)) \le C_l \\ \frac{A_i^l(t)}{\xi_l(t)} C_l & \text{otherwise.} \end{cases} \tag{14}$$

Between Runge-Kutta steps, the simulator is still event-driven. That is, each packet arriving at a network queue is handled as a simulation event. The following shows the algorithm we use to process packet arrival events during the k^{th} Runge-Kutta interval:

1. We create three shadow variables to keep track of the changes made to the instantaneous queue length, the average queue length, and the packet drop probability, and initialize them using values from the previous Runge-Kutta step: $\tilde{q}_l := q_l(k\delta), \tilde{x}_l := x_l(k\delta)$, and $\tilde{p}_l := p_l(k\delta)$.
2. Suppose $E_1, E_2, \cdots, E_m$ are the events representing packets arrived at queue l . Let $t_1, t_2, \cdots, t_m$ be the timestamps of these events and $t_0 = k\delta \le t_1 \le t_2 \le \cdots \le t_m \le (k+1)\delta$. Define the time between consecutive events as $\Delta t_i = t_i - t_{i-1}$, for $i = 1, 2, \cdots, m$. For each packet arrival event E_i:
   (a) We first update the instantaneous queue length at the time of the arrival: $\tilde{q}_l := \min\{Q_l, \max\{0, \tilde{q}_l + \Delta t_i(\Lambda_l(k\delta)(1 - \tilde{p}_l) - C_l)\}\}$ It is important that we include only the fluid arrival rate since changes to the queue length due to packet arrivals are handled individually at each event processing step.
   (b) If the queue is full or the packet is selected to be dropped according to the packet drop probability $\tilde{p}_l$, we discard this packet and continue to process the next packet arrival event.
   (c) Otherwise, we add the size of the packet to the instantaneous queue length $\tilde{q}_l$, and update the average queue length $\tilde{x}_l$ and the packet drop probability $\tilde{p}_l$ according to the RED policy.
   (d) We finish the processing of this packet arrival event by scheduling a packet departure event after a queuing delay equal to $\tilde{q}_l / C_l$.

Here, the use of shadow variables is important because the change otherwise made to the original variables controlled by the fluid model will be difficult to reconcile at the Runge-Kutta step. The value of these variables reflects the state of the queue at the beginning of the time step and is updated accordingly upon each packet arrival. On the other hand, the effect of packet flows on fluid flows can also be captured since we include the packet arrival rates into the equations used by the fluid model (Equations (13) and (14)).

4 EXPERIMENTS

To demonstrate the correctness of our hybrid model, we implemented the packet-level integration scheme in SSFNet. We start with a test network first used by Gu, Liu, and Towsley (2004). The topology of the network is shown in Figure 1. The network consists of 12 nodes and 11 links. The delay and bandwidth of all links are set to be 10 ms and 100 Mbps, respectively. There are a total of 22 network interfaces and, correspondingly, we have 22 RED queues. We set the maximum queue size to be 5 MB. The RED queue parameters are configured with q^{min} = 100 KB, q^{max} = 4.5 MB, and p^{max} = 0.1. The weight used in the EWMA computation is 0.0004. For packet-oriented simulation, we use TCP Reno with 128 as the maximum window size. We fixed the step size of the fluid model to be 1 ms.

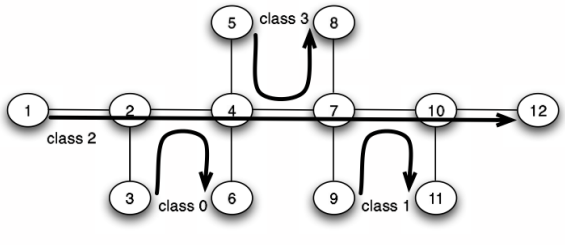

Figure 1: A Test Network Topology with Four Classes of Flows

There are four classes of flows on this test network. Class 0 and class 1 each consists of 10 TCP flows, while class 2 consists of 20 TCP flows. All flows in these classes start from time 0 and are persistent throughout the simulation. Class 3 consists of 40 TCP flows, which start at time 30 and only last for 30 seconds. As we will see, the competition between class 2 and class 3 traffic creates a congestion at the link connecting node 4 and 7 between 30 and 60 seconds.

To establish the baseline, we first compare the results from pure-fluid and pure-packet simulations. Figure 2 shows the lengths of the network queue (both instantaneous and average queue lengths) in node 4 connecting to node 7. Figure 3 shows the end-to-end delays experienced by TCP flows in both class 0 and class 2. In essence, the fluid model does capture the expected network behavior. The result from the packet simulation exhibits much larger variations, whereas the fluid model can only describe the average behavior. This is expected since information regarding the transient behavior is lost in the fluid model in favor of computational efficiency. We observe that the overall queuing level projected by the fluid model is slightly higher than that from the packet simulation. This indicates that further tuning of the fluid model may be necessary, a strategy that was suggested by Liu et al. (2004).

Next, we test the correctness of our hybrid model. We choose to use the fluid model to describe the TCP flows within classes 0, 1, and 3. And we model the TCP flows within class 2 as a combination of fluid and packet flows. In this experiment we purposefully varied the proportion of packet flows in class 2 from 10% to 50% and finally to 100%. That is, out of 20 TCP flows in class 2, we choose to model 2, 10, and 20 flows as packet flows. The results are shown in Figures 4, 5, and 6. The introduction of packet flows increases the variation in both queue length and end-to-end delay measurement. With more packet flows,

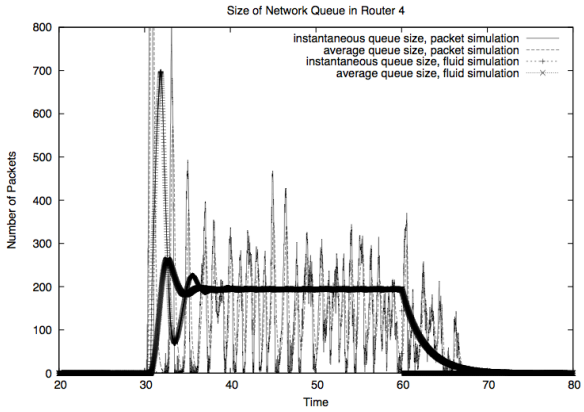

Figure 2: Compare Queue Length Between Pure-Fluid and Pure-Packet Simulations

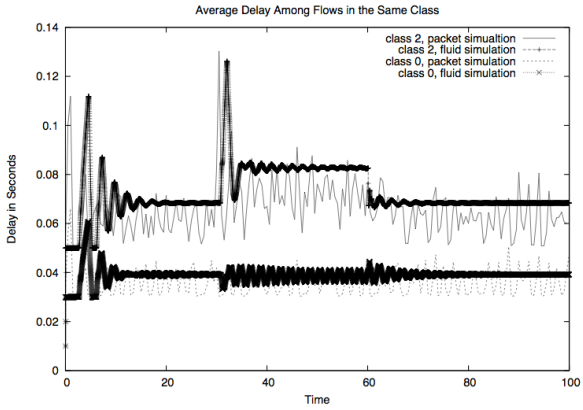

Figure 3: Compare Path Delay Between Pure-Fluid and Pure-Packet Simulations

the simulation does a better job at capturing more detailed traffic behavior and therefore the variation grows larger.

The next experiment shows the computational efficiency of the hybrid model. We use the dumbbell topology (shown in Figure 7), which is commonly used to analyze the TCP congestion control algorithms. There are N server nodes (acting as traffic sources) aligned at the left of the network and N corresponding client nodes (acting as traffic sinks) aligned to the right. During simulation, each pair of client-server nodes simultaneously engage in M TCP sessions. Each client/server node has a dedicated connection of bandwidth $(10 \cdot M)$ Mbps to a router and has a maximum queue size of M MB. The link between the two routers has a bandwidth of $(10 \cdot M \cdot N)$ Mbps. Each router has a maximum queue size of $(M \cdot N)$ MB. The RED queue

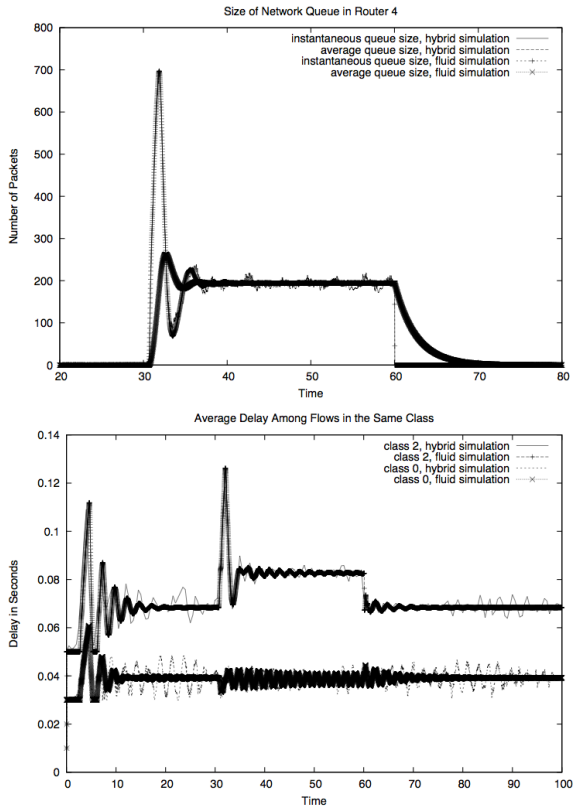

Figure 4: Hybrid Simulation with 10% Packet Flows in Class 2

parameters are set so that q^{min} is 1% of the maximum queue size and q^{max} is half of the maximum queue size.

We ran the simulation for 100 simulated seconds on an AMD Athlon64 machine with a 2.2 GHz CPU and 2 GB of memory. The SSFNet simulator was compiled using the GNU C/C++ compiler set at optimization level 3. Figure 8 shows the results of an experiment, in which we set the total client-server connections N to be 5, allowed each pair of client and server nodes to carry $M = \{5, 10, 20, 40\}$ simultaneous TCP flows, and varied the proportion of packet flows. Here we report the measured execution time of each simulation run. The pure-fluid model takes virtually no time to finish the simulation, while the cost of the packet model is roughly proportional to the number of packet TCP flows in the system. We believe the slight increase in runtime as one increases the proportion of packet flows is primarily attributed to the larger context switch overhead from process-oriented implementation of the foreground packet flows. The fluid model (using our ODE solver at each simulated router) can run several orders of magnitude faster than the packet-oriented simulation. Comparing to the cost of simulating packet TCP flows, increasing the number of fluid flows in the system introduces negligible

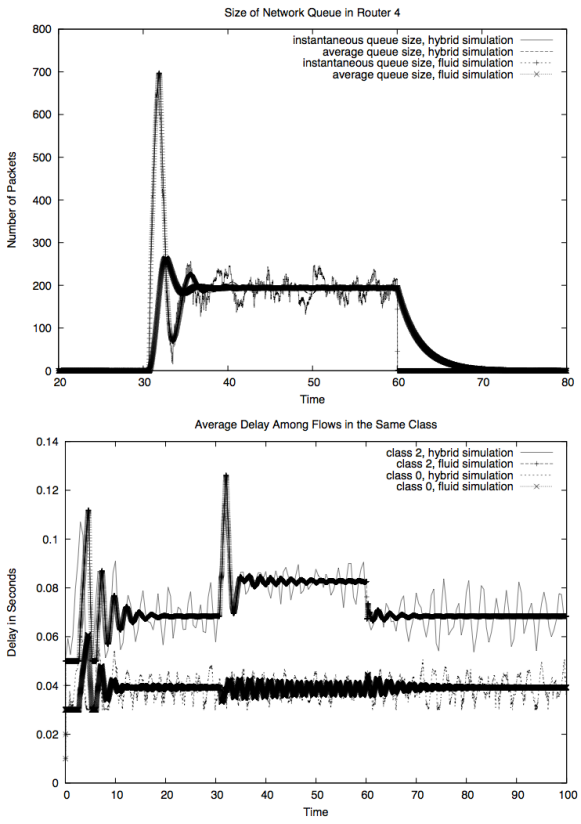

Figure 5: Hybrid Simulation with 50% Packet Flows in Class 2

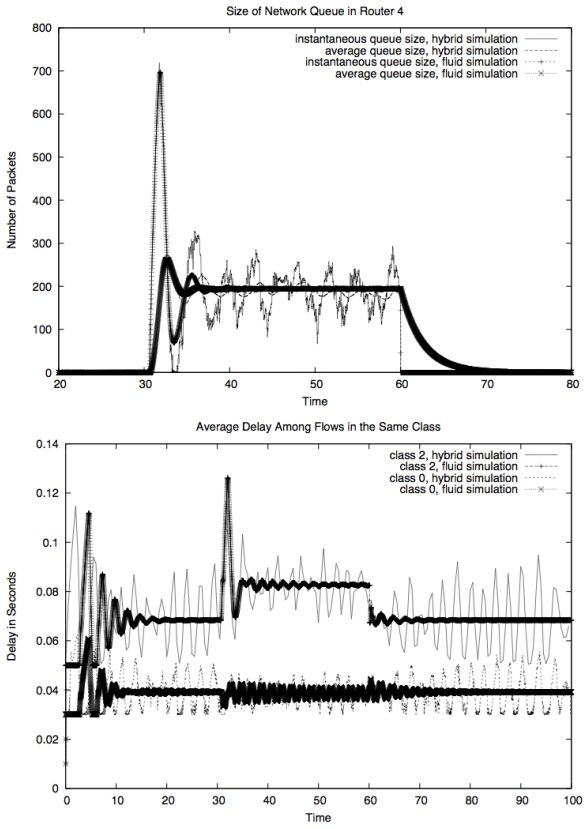

Figure 6: Hybrid Simulation with 100% Packet Flows in Class 2

overhead to the overall performance. In summary, this example highlights the need of reducing the overall number of packet flows and replacing them with fluid flows in order to keep up with the wall-clock time.

5 CONCLUSIONS

In this paper we propose a hybrid simulation model that merges continuous and discrete-event simulation paradigms by allowing interaction of network packet flows and fluid flows at the packet level within each simulation router. This scheme enables the tradeoff between packet-oriented simulation, which is accurate but comes with a high computational cost, and the fluid-based model, which describes only the average behavior of the traffic dynamics and yet is highly scalable.

Our immediate future work includes further testing in realistic large-scale network scenarios. We want to further investigate the effect on the loss of accuracy introduced by fluid flows of various proportions. Another important direction is to parallelize the fluid model to fully integrate the model with the discrete-event network simulator that can run both on parallel and distributed platforms. We are currently investigating approaches to dynamically change the traffic composition in order to meet the real-time requirement.

ACKNOWLEDGMENTS

We would like to thank Yong Liu for generously sharing the fluid model code. Also, we thank Yu Gu for sharing the details of the experiments presented in their Infocom 2004 paper.

This research is supported in part by the National Science Foundation grant CNS-0546712.

REFERENCES

Floyd, S., and V. Paxson. 2001, August. Difficulties in simulating the Internet. *IEEE/ACM Transactions on Networking* 9 (4): 392–403.

Fujimoto, R., K. Perumalla, A. Park, H. Wu, M. Ammar, and G. Riley. 2003, October. Large-scale network simulation – How big? How fast? In *Proceedings of the IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer Telecommunication Systems (MASCOTS)*, 116–125.

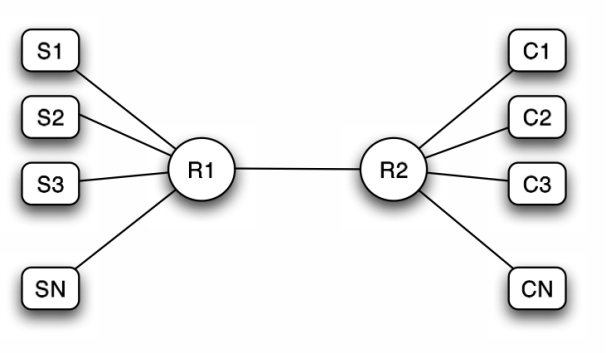

Figure 7: A Dumbbell Network Topology

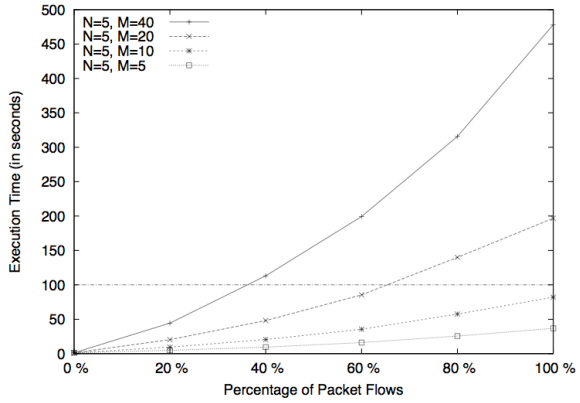

Figure 8: Execution Time for Various Traffic Compositions

Fujimoto, R. M. 2000. *Parallel and distributed simulation systems*. John Wiley & Sons.

Gu, Y., Y. Liu, and D. Towsley. 2004, March. On integrating fluid models with packet simulation. In *Proceedings of IEEE INFOCOM 2004*.

Kiddle, C., R. Simmonds, C. Williamson, and B. Unger. 2003, June. Hybrid packet/fluid flow network simulation. In *Proceedings of the 17th Workshop on Parallel and Distributed Simulation (PADS'03)*, 143–152.

Liljenstam, M., J. Liu, D. M. Nicol, Y. Yuan, G. Yan, and C. Grier. 2005, June. RINSE: the real-time interactive network simulation environment for network security exercises. In *Proceedings of the 19th Workshop on Parallel and Distributed Simulation (PADS'05)*, 119–128.

Liu, Y., F. L. Presti, V. Misra, D. F. Towsley, and Y. Gu. 2004, July. Scalable fluid models and simulations for large-scale ip networks. *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 14 (3): 305–324.

Melamed, B., S. Pan, and Y. Wardi. 2004, July. HNS: a streamlined hybrid network simulator. *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 14 (3): 251–277.

Misra, V., W.-B. Gong, and D. Towsley. 2000, August. Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED. In *Proceedings of the 2000 ACM SIGCOMM Conference*, 151–160.

Nicol, D. M., M. Goldsby, and M. Johnson. 1999, October. Fluid-based simulation of communication networks using SSF. In *Proceedings of the 1999 European Simulation Symposium*.

Nicol, D. M., M. Liljenstam, and J. Liu. 2005, December. Advanced concepts in large-scale network simulation. In *Proceedings of the 2005 Winter Simulation Conference (WSC'05)*.

Nicol, D. M., J. Liu, M. Liljenstam, and G. Yan. 2003, December. Simulation of large-scale networks using SSF. In *Proceedings of the 2003 Winter Simulation Conference (WSC'03)*.

Nicol, D. M., and G. Yan. 2004, July. Discrete event fluid modeling of background TCP traffic. *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 14 (3): 211–250.

Nicol, D. M., and G. Yan. 2005, June. Simulation of network traffic at coarse time-scales. In *Proceedings of the 19th Workshop on Parallel and Distributed Simulation (PADS'05)*, 141–150.

Riley, G. F., T. M. Jaafar, and R. Fujimoto. 2002, October. Integrated fluid and packet network simulations. In *Proceedings of the IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer Telecommunication Systems (MASCOTS)*, 511–518.

Zhou, J., Z. Ji, M. Takai, and R. Bagrodia. 2004, April. MAYA: integrating hybrid network modeling to the physical world. *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 14 (2): 149–169.

Zukerman, M., T. D. Neame, and R. G. Addie. 2003, April. Internet traffic modeling and future technology implications. In *Proceedings of IEEE INFOCOM 2003*.

AUTHOR BIOGRAPHY

JASON LIU is an Assistant Professor of Computer Science at the Colorado School of Mines. His research focuses on parallel discrete-event simulation, performance modeling and simulation of computer systems and communication networks. He received a B.A. in Computer Science from Beijing Polytechnic University in China in 1993, an M.S. in Computer Science from College of William and Mary in 2000, and a Ph.D. in Computer Science from Dartmouth College in 2003. He currently serves as a WSC proceedings editor. His e-mail address is `<xliu@mines.edu>`.