On Integrating Fluid Models with Packet Simulation

Yu Gu
Computer Science Department
University of Massachusetts
Amherst, MA 01003
Email: yugu@cs.umass.edu

Yong Liu
Computer Science Department
University of Massachusetts
Amherst, MA 01003
Email: yongliu@cs.umass.edu

Don Towsley
Computer Science Department
University of Massachusetts
Amherst, MA 01003
Email: towsley@cs.umass.edu

***Abstract*— Fluid models have been shown to be efficient and accurate in modelling large IP networks. However, unlike packet models, it is difficult to extract packet-level information from them. In this paper, we present a hybrid simulation method that maintains the performance advantage of fluid models while providing detailed packet level information for selected packet traffic flows. We propose two models to account for the interaction between background TCP traffic in a fluid network and foreground packet traffic of interest. The first assumes that the packet traffic poses a negligible load on the fluid network whereas the second accounts for the added load by transforming the packet traffic into fluid flows and solving the resulting enhanced fluid model. The first of these yields an efficient one pass solution algorithm whereas the second requires an additional pass to account for the packet traffic load. We establish the correctness of both approaches and present their implementation within ns-2. Comparisons between the hybrid models and a classical packet simulation show the two pass approach to be quite accurate and computationally efficient.**

I. Introduction

Networks, and the Internet in particular, have seen an exponential growth over the past several years. This growth is likely to continue for the foreseeable future, and understanding the behavior of such systems is critically important. A number of discrete event-driven simulators [1], [2], [3], [4] have been developed for this purpose. These simulators provide accurate information for every simulated packet. They provide an important tool for designing new protocols, improving existing protocols, and verifying new observations. However the simulation capabilities of these simulators have fallen far behind the scale of the Internet today and this gap is growing as the size and speed of the network is growing. As an alternative, fluid models have been proposed in recent works [5], [6], [7] to analyze the performance of networks. The fluid models can predict the behavior of large networks both accurately and efficiently. In [6], networks consisting of hundreds of routers and thousands of high bandwidth links supporting millions of flows can be simulated in minutes on a desktop PC, a feat that is unachievable by current discrete event-driven simulators. However the fluid model provides no detailed information regarding individual packets, and its application is hindered by this limitation.

A simulation method that combines the best features of the above fluid and packet-level approaches is desirable for several reasons. First, by efficiently simulating large networks and providing detailed information for selected individual traffic flows, it is possible to study the performance of communication protocols deployed at end hosts across a wide area high speed network. Research on existing and future communication protocols for the Internet can potentially benefit from such a simulation method. Second, it would permit the performance study of small edge networks, such as wireless networks that exchange traffic with wide area networks. A small edge wireless network would best be simulated as a packet-level network whereas the WAN would be more efficiently modelled as a fluid network. Finally, introducing non-TCP traffic into the fluid model provides an alternative method to study wide area traffic, which may lead to better models of real Internet traffic. Such models could be useful in creating more efficient simulation methods and studying a large class of network problems.

In this paper, using the topology-aware fluid model presented in [6], we develop a hybrid simulation approach that consists of a set of packet flows and fluid flows. We present two models of the interaction between the packet flows and the fluid flows traversing a network. Algorithms are presented for deriving the behavior of the packets, e.g., end-to-end latency and loss. The simplest of these models ignores the effect that the packets have on the fluid network and simply uses the solutions to the fluid network model to determine the outcome of each packet. The second approach accounts for the load introduced by the packet flows by transforming them into fluid flows that feed the fluid network. The solution of this augmented network is then used to determine the outcomes of the packets. These approaches require one and two passes, respectively and will henceforth be referred to as the one pass and two pass interaction models. Figure 1 shows an example of such a hybrid simulation.

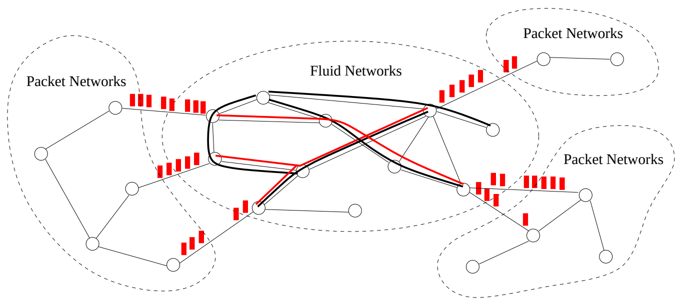


Fig. 1. An Example Hybrid Simulation

For both approaches, we establish the correctness of the resulting simulators and evaluate their accuracy and computational speedup through simulation. We have implemented both approaches in several popular simulators, including ns-2. We find that the one pass algorithm is fast and accurate provided that the offered packet load is very small compared to the load offered by fluid flows. Its accuracy degrades as the offered load due to the packet flows increases. On the other hand, the two-pass algorithm is very accurate, independent of the load introduced by the packet flows while incurring a slight slowdown. In the rest of the paper, the word *simulator* refers to the discrete event-driven simulator used in the hybrid simulation unless further clarified.

Several works are particularly related to our study. In [8], a hybrid model is established for fast network simulation. The idea is to model network traffic as fluid flows and generate discrete events for packet losses within the network. [9] outlines the design of a fluid-oriented hybrid simulator admitting both packet flows and fluid flows. The hybrid simulator is implemented using the discrete event simulation framework. Recently, a dynamic simulation backplane was developed for creating distributed, component-based simulations of communication networks by interconnecting models of sub-networks drawn from different network simulation packages [10]. Based on this backplane, a multi-paradigm simulation framework, MAYA, is proposed to integrate three disparate modelling paradigms: discrete event models, analytical models and physical network interfaces. In particular, the fluid model proposed in [5] is integrated with the packet simulator Qualnet [11], [3].

The rest of the paper is organized as follows. In Section II, we review the fluid model of [6], which forms the basis for our work. In Section III, two traffic interaction models are proposed describing how the foreground packet flows interact with the background fluid flows. In Section IV we analyze the synchronization between the fluid model solver and the simulator in the context of a hybrid simulation. We describe our implementation in ns-2 in Section V and present the experimental results in Section VI. Section VII concludes the paper and points out future research directions.

II. The Fluid Model

In [5], a dynamic fluid flow model is established for a network of routers serving a population of persistent TCP flows working in congestion avoidance stage. Based on this model, a refined fluid model is proposed in [6] to accurately model traffic propagation inside networks. In the following, we briefly describe the fluid flow model. Details can be found in [5], [6].

A network is modelled as a directed graph $G = (V, E)$ where V is a set of routers and E is a set of links. Each link $l \in E$ has capacity of c_l and propagation delay of r_l. In addition, associated with each link is an AQM policy, characterized by a probabilistic discarding/marking function $p_l(t)$, $t \geq 0$, which may depend on link state such as queue length. The queue length of l is $q_l(t)$. The network G serves a population of N classes of TCP flows. TCP flows within the same class have the same characteristics, follow the same route and experience the same round trip propagation delays. We denote by n_i the number of flows in class i. For a flow in class i, let $W_i(t)$ be its window size, $E_i = (l_{i,1}, l_{i,2}, \ldots, l_{i,h_i})$ be the sequence of h_i links on its path, and $a_i^l(t)$ and $d_i^l(t)$ be its arrival rate and departure rate on link $l \in E_i$ respectively. The system evolution is governed by a set of differential equations:

- *TCP Window Dynamics.*

$$\frac{dW_i(t)}{dt} = \frac{\mathbf{1}(W_i(t) < M_i)}{R_i(t)} - \frac{W_i(t)}{2}\lambda_i(t) \qquad (1)$$

  where M_i is the maximal TCP window size, $R_i(t)$ and $\lambda_i(t)$ denote the round trip time and the loss indication rate at time t. $\mathbf{1}(\mathcal{P})$ is an indicator function which assumes value 1 if the predicate $\mathcal{P}$ is true and 0 otherwise. The first term on the right hand side models TCP additive increase: TCP increases its congestion window by one every round trip time if there is no packet loss. The second term models TCP multiplicative decrease: TCP reduces its window by half upon a packet loss.

- *Traffic Propagation.* Traffic generated by TCP class i propagates inside the network according to the following set of expressions of $\{a_i^l(t), d_i^l(t)\}$.
  Arrival Rate

$$a_i^l(t) = \begin{cases} W_i(t)/R_i(t), & l = l_{i,1} \\ \\ d_i^{l_{i,j-1}}(t - r_{l_{i,j-1}}), & l = l_{i,j}, \\ & 2 \leq j \leq h_i \end{cases} \qquad (2)$$

  TCP class i's arrival rate at the first queue on its path is just its sending rate. At any other queue, its arrival rate is its departure rate from the upstream queue after a delay equal to the link propagation delay.
  Departure Rate

$$d_i^l(t) = \begin{cases} a_i^l(t), & q_l(t) = 0 \\ \\ \frac{a_i^l(t-w)}{\sum_{j \in N_l} a_j^l(t-w)} c_l, & q_l(t) > 0 \end{cases} \qquad (3)$$

  where w is the queueing delay experienced by the traffic departing from l at time t. When the queue length is zero, the departure rate at time t equals the arrival rate. When the queue is not empty, the service capacity is shared among the competing flows in proportion to their arrival rates.

- *Queue Evolution and AQM policy.*

$$\frac{dq_l(t)}{dt} = -\mathbf{1}(q_l(t) > 0)c_l + \Big(\sum_{i \in N_l} n_i a_i^l(t)\Big) \times (1 - p_l(t)), \qquad (4)$$

  where N_l is the set of TCP classes traversing link l. AQM schemes calculate a packet drop probability $p_l(t)$ based on queue length and traffic rate. Models for different AQM schemes, e.g. PI controller [12], RED [13], [6], have been established.

Key network performance metrics, e.g., packet drop probability, queueing delay and TCP throughput, are obtained by solving the model numerically.

III. Traffic Interaction Models

In this section, we address the problem of modelling interactions between the foreground packet traffic and the background TCP traffic. When the foreground packet traffic traverses the fluid network, the delay and drop probabilities of the packets are determined by the queue lengths and drop probabilities of the queues in the fluid network, which are computed by the fluid model solver. On the other hand, the queue lengths and drop probabilities of these queues are affected also by the traversing foreground packet traffic, which then affect the background TCP traffic accordingly.

A hybrid model is built based on the fluid model. In the hybrid model, the network G serves not only a population of N classes of TCP flows, but also a population of M packet flows. A packet flow consists of a sequence of packets that share the same path in the fluid network. Each packet flow i has a packet arrival process, $\gamma_i(t)$, which denotes the number of packets arriving at the fluid network by time t, $\{s_i\}$, which denotes the sequence of packet lengths associated with the flow, and a path in the fluid model defined as $E_i = (l_{i,1}, l_{i,2}, \ldots, l_{i,h_i})$[1].

We present two algorithms for dealing with the traffic interaction problem. The first assumes that packet flows impose a load on the fluid network that can be neglected. The second accounts for the load that the packets impose on the fluid network. The first will require a single pass through time in order to determine outcomes for each packet (loss/no loss, end-end delay) and will be referred to as the one-pass traffic interaction model; whereas the second requires an additional preliminary pass to solve the fluid network accounting for the load introduced by the packet flows and will be referred to as the two-pass traffic interaction model.

A. One-pass Traffic Interaction Model

Since the one-pass traffic interaction model assumes that the packet flows have negligible effect on the background TCP traffic, the queue lengths and drop probabilities seen by the packets are identical to those predicted by the fluid model. Thus, the delay and drop probability experienced by a packet traversing through a fluid network are estimated using the queue lengths and drop probabilities derived directly from the fluid model in [6]. The delay and drop probability for each packet are computed cumulatively along its path in the fluid network. The packet is scheduled to depart from the fluid network according to this delay and drop probability.

Consider an arbitrary packet D belonging to the i^{th} foreground flow of size $s(D)$. Let $t(l_{i,1})$ denote the time that D arrives at the fluid network and $t(l_{i,j})$ the time D arrives at link $l_{i,j}$ ($j = 2, \ldots, h_i$). Also assume that D is the k^{th} packet traversing $l_{i,j}$ and $e^{k-1}_{l_{i,j}}$ represents the time when the $(k-1)^{th}$ packet leaves link $l_{i,j}$, then D will arrive at link $l_{i,j+1}$ at $t(l_{i,j+1}) = e^k_{l_{i,j}} + r_{l_{i,j}}$, where $e^k_{l_{i,j}}$ is

$$\begin{aligned} e^k_{l_{i,j}} &= \max(t(l_{i,j}) + (q_{l_{i,j}}(t(l_{i,j})) + s(D))/c_{l_{i,j}}, \\ &\qquad e^{k-1}_{l_{i,j}} + s(D)/c_{l_{i,j}}) \\ &\qquad\qquad 1 \le i \le h \end{aligned} \tag{5}$$

The first term in the $\max$ function is the time at which D would leave $l_{i,j}$ in the absence of any other packets in the queue. However, D cannot begin transmission before the departure of the $(k-1)^{st}$ packet. This is accounted for by the second item of the $\max$ function, where $e^{k-1}_{l_{i,j}} + s(D)/c_{l_{i,j}}$ defines the earliest time D can leave $l_{i,j}$. With this restriction, the time order of two consecutively arrived packets at any link is maintained, and therefore, no out-of-order packets will occur in the fluid network in the context of a hybrid simulation. D departs the fluid network at time $e^k_{l_{i,h_i}} + r_{l_{i,h_i}}$, which we will denote as $t(D)$.

D will also experience a drop probability at each queue it passes. Assume that drop events at different queues are independent, the overall drop probability $p(D)$ is

$$p(D) = 1 - \prod_{j=1}^{h_i}(1 - p_{l_{i,j}}(t(l_{i,j}))), \tag{6}$$

Therefore, with probability $p(D)$ packet D will be dropped or marked by the fluid network and with probability $1-p(D)$, D is scheduled by the fluid network to depart the fluid network at time $t(D)$.

The one-pass model is based on the assumption that the packet flows have negligible effect on the fluid network. This should be reasonable provided that the rate of the traversing foreground packet traffic is small enough. However, it is often the case that the foreground flows have considerable throughput such that the accuracy of the simulation results given by the one-pass model can be impaired. One extreme example is a hybrid simulation in which there is one background TCP class traversing two identical queues in the fluid network and there are foreground packets traversing the second queue. If the throughput of the foreground packet traffic is high enough, the second queue becomes the bottleneck. However, in the one-pass model, the second queue is always empty regardless of the rate of the traversing foreground packet traffic. We will study this further in Section VI.

B. Two-pass Traffic Interaction Model

The two-pass traffic interaction model extends the one-pass model to account for the interaction between the packet flows and the background TCP traffic in the fluid network.

In the two-pass model, the behavior of the fluid network is determined during a first pass and the effect on the packets is determined once they traverse the network during the second pass.

In the first pass of the two-pass model, the packet flows is transformed into foreground fluid flows that can be incorporated into the solution of the fluid model. This transformation

[1]The path in the fluid network traversed by packets can be determined by any routing algorithm. In this paper, we assume that this path is known when the packet arrives the fluid network.

and the resulting solution of the model constitutes the first pass. In the second pass, the delay and the drop probabilities of the traversing packets are estimated using the queue states obtained during the first pass.

1) The First Pass: As previously defined, there are M packet flows traversing the fluid network in the hybrid model and $\gamma_i(t)$ denotes the number of packets in the i^{th} foreground flow that arrive at the fluid network by time t. Time is divided into constant length *smoothing intervals* of length δ_s The packet flow during each interval is modelled as a constant rate fluid with an arrival rate $a_i(t)$ given by

$$\begin{aligned} a_i(t) &= \frac{\gamma_i((k+1)\delta_s) - \gamma_i(k\delta_s)}{\delta_s} \\ &\quad \forall t, k, k\delta_s \le t < (k+1)\delta_s, k = 0, 1, \ldots \quad (7) \end{aligned}$$

Here $a_i(t)$ is treated as the sending rate of the i^{th} foreground flow. Note that the smaller the smoothing interval, the more accurately the packet flow rate reflects the actual packet arrival rate.

Assume that link l in the fluid network is traversed by $N_T(l)$ classes of TCP flows and $N_P(l)$ foreground flows. For the i^{th} class of TCP flow that traverses l, we denote its arrival rate and departure rate at l by $a^l_{T,i}(t)$ and $d^l_{T,i}(t)$. And for the i^{th} foreground flow, we denote its arrival rate and departure rate at l by $a^l_{P,i}(t)$ and $d^l_{P,i}(t)$. $a^l_{\cdot,i}(t)$ and $d^l_{\cdot,i}(t)$ are used where it makes no difference whether the fluid flow is a class of TCP flows or a foreground flow.

For each queue in the fluid network, its service capacity is now shared not only by the TCP flows that traverse the queue, but also by the foreground flows that traverse it. When the queue at l is not empty, its service capacity c_l is shared among the $N_T(l)$ classes of TCP flows and the $N_P(l)$ foreground flows in proportion to their arrival rates. Thus, the expressions relating the departure and arrival process of the i^{th} flow at link l (2) and (3) are now modified to include both the TCP flows and the packet flows

$$a^l_{\cdot,i}(t) = \begin{cases} a_i(t), & l = l_{i,1} \\ d^{l_{i,j-1}}_{\cdot,i}(t - r_{l_{i,j-1}}), & l = l_{i,j}, \\ & 2 \le j \le h_i \end{cases} \quad (8)$$

$$d^l_{\cdot,i}(t) = \begin{cases} a^l_{\cdot,i}(t), & q(t) = 0 \\ \frac{n_i a^l_{\cdot,i}(t-w)}{A_T + A_P} c_q, & q(t) > 0 \end{cases} \quad (9)$$

in which

$$A_T = \sum_{j \in N_T(l)} n_j a^q_{T,j}(t-w)$$

is the total arrival rate of the TCP classes and

$$A_P = \sum_{j \in N_P(l)} a^l_{P,j}(t-w)$$

is the total arrival rate of the foreground flows. Again, w is the queueing delay (or waiting time) experienced by the traffic departing from q at time t and n_i is the number of flows in the i^{th} TCP class, for foreground flow, n_i is 1. Also recall that if the flow is a TCP flow, $a_i(t)$ is the sending rate of the TCP class computed by (1).

The throughput of the foreground flows passing through link l should also be taken into account when computing the queue lengths. At l, the arrival rate of the packet flows is $\sum_{i \in N_P(l)} a^l_{P,i}(t)$. This factor is added to each of the differential equations that models the queue length variation, and equation (4) becomes

$$\begin{aligned} \frac{dq_l(t)}{dt} &= -\mathbf{1}(q_l(t) > 0)c_q + \Big(\sum_{i \in N_T(l)} (n_i a^l_{T,i}(t)) \\ &\quad + \sum_{i \in N_P(l)} a^l_{P,i}(t) \Big) (1 - p_l(t)) \quad (10) \end{aligned}$$

These equations are closely coupled when the fluid model is solved. Equation (7) determines the sending rate of foreground flows, (8) and (9) determine the arrival and departure rates among competing TCP classes and foreground flows at each queue, and (10) describes the queue lengths with regard to both the TCP classes and the foreground flows. Together with (1), the equation that defines the behavior of TCP class, these equations describe the behavior of the fluid network and capture the impact of the traversing foreground packet traffic.

2) The Second Pass: The second pass in the two-pass model is identical to the single pass in the one-pass traffic interaction model. Delays and drop probabilities are computed according to (5) and (6), and packets are scheduled to arrive at the next nodes on their path in the packet network accordingly. The only difference is that the network states used in these equations are now results obtained by the first pass and reflect the effect of the traversing packets' throughput.

IV. Synchronization in Hybrid Simulation

While most existing network simulators, [1], [2], [4] etc., are discrete event-driven simulators, the fluid model is a set of (coupled) ordinary differential equations of TCP network dynamics as functions of time [5], [6]. The states of the TCP network at any time can be directly obtained, in theory, by solving the set of differential equations. In practice, in order to save computational resources, the fluid model is solved incrementally as a function of time and all the network states prior to the current time in the fluid model can be obtained. For example, in [6], a fixed step-size Rungge-Kutta algorithm is implemented to solve the fluid model; thus the fluid model is solved using a time-stepped network simulator. Thus, the simulator generating packets arriving to the fluid network and the fluid model solver are likely to have separate time management systems and, if integrated in the context of a hybrid simulation, they will have to be synchronized. For example, in a two-pass hybrid simulation, the fluid model takes packet traffic from the packet simulator. It is essential to have the packet traffic rate available in time to advance the fluid model. On the other hand, the packet simulator needs those packets traversing fluid network to be delivered by the fluid model in time.

In the following, we study the synchronization problem between the fluid model solver and the packet simulator. We discuss issues on how to correctly advance the packet simulator and the fluid model solver in both the one-pass model hybrid simulation and the two-pass model hybrid simulation. We identify the necessary conditions for the hybrid simulation to be carried out correctly and present our synchronization solution. We assume that all of the packets are generated and consumed by a single packet simulator; however, the results also apply to cases where there are multiple packet simulators.

A. Definition

We first introduce some notation:

Definition 1 (T_f): the simulation clock of the fluid model solver, meaning the states of the fluid network, e.g. queueing delay, packet loss, flow rate, etc., have been obtained for all $t \le T_f$.

Definition 2 (T_s): the simulation clock of the simulator, meaning all the packet events before T_s have been processed by the simulator.

Definition 3 (τ): the minimal propagation delay of links that are the last hops on the packet's paths inside the fluid network.

B. Simulation Clock Constraints

The purpose of synchronization between the fluid model and packet simulator is to avoid out of order events. More specifically, we want to ensure that the fluid model never deliver a packet to the packet simulator with a timestamp $t_0 < T_s$. At the same time, the packet simulator must never inject a packet into the fluid network with timestamp $t_0 < T_f$. This places constraints on the rates that $\{T_s, T_f\}$ can advance. In this section, we study the constraints on $\{T_s, T_f\}$ due to the interaction between the fluid model and the packet simulator. The following Lemma describes how the fluid model affects the advancement of time in the packet simulator.

Lemma 1: Given T_f, it is safe to advance T_s up to $T_f + \tau$.

Proof: We prove this by contradiction. Assume that there is a packet D belonging to the i-th flow. Assume that it is the k-th packet traversing link h_i. Now, suppose that packet D departs the fluid network at time $e^k_{l_{h_i}} + r_{l_{h_i}} \le T_f + \tau$ and $t_{l_{h_i}} > T_f$. If this were the case, then it would not be safe to advance T_s up to $T_f + \tau$. However, $e^k_{l_{h_i}} + r_{l_{h_i}} \le T_f + \tau$ implies that $e^k_{l_{h_i}} \le T_f + \tau - r_{l_{h_i}} \le T_f$. We know that $t_{l_{h_i}} \le e^k_{l_{h_i}}$. Therefore, $t_{l_{h_i}} \le T_f$ resulting in a contradiction. ■

Now we turn to the constraint placed on T_f by the packet simulator. In the one-pass model, since we ignore the impact of packet traffic on the fluid network, the advance of T_f is independent of the packet simulator. In the two-pass model, the fluid model needs the foreground traffic rate vector $\{a_i(t)\}$, which is calculated as in (7), to advance its simulation clock. Due to the discrete nature of the smoothing process, T_f will be advanced in unit of the smoothing interval δ_s. The following theorem states the necessary condition to advance the two-pass hybrid simulation correctly.

Theorem 1: The two-pass hybrid simulation can be advanced correctly only if $\delta_s \le \tau$

Proof: From Lemma 1, we have

$$T_s \le T_f + \tau. \tag{11}$$

At the same time, in order to advance T_f to $T_f + \delta_s$, all the packets which are supposed to arrive at the fluid network by time $T_f + \delta_s$ are required to be processed by the packet simulator. This translates into

$$T_f + \delta_s \le T_s \tag{12}$$

From (11) and (12), we must have $\delta_s \le \tau$. ■

C. Synchronization Approach

In this section, we present our approach for synchronizing the fluid model solver and the simulator. Our approach aims at satisfying the synchronization requirements for the two-pass model, which is more strict than those for the one-pass model. So when the one-pass model is used in our implementation, its synchronization requirements are naturally satisfied.

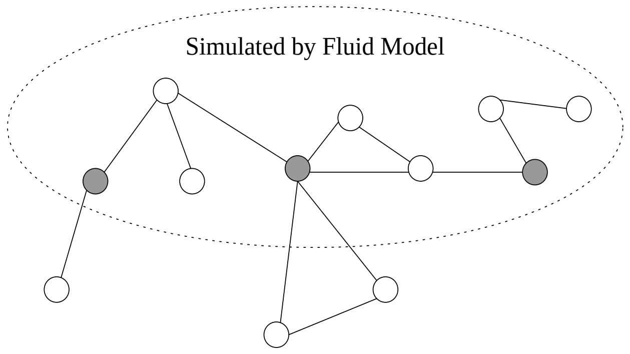
In our implementation, the fluid model solver is synchronized to the simulator every smoothing interval δ_s which is chosen to be smaller than τ. At each time when $T_s = k\delta_s$, $k = 1, 2, \ldots$, a synchronization event is scheduled to happen. During this event, the fluid model solver evolves from $T_f = (k-1)\delta_s$ to $T_f = T_s = k\delta_s$. Due to Lemma 1, fluid model schedules out all packets that are supposed to arrive at packet network before $T_f + \tau$. Since $\delta_s \le \tau$, we can advance T_s to the next synchronization point $(k+1)\delta_s$.

V. IMPLEMENTATION

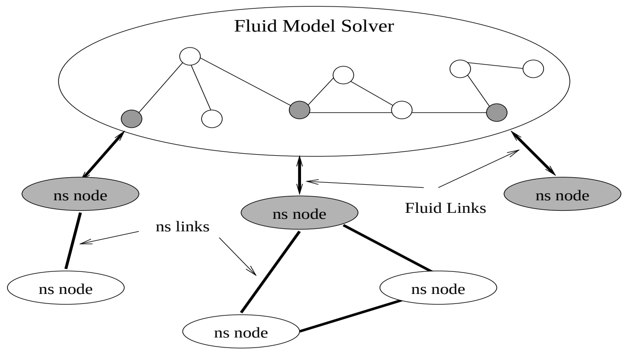
We have implemented the two traffic interaction models in ns-2[1], pdns[4] and the Backplane[10] by integrating the fixed step-size fluid model solver developed in [6]. In this section, we describe our implementation in the ns-2 network simulator as an example of how the hybrid simulation approach might be implemented.

The design objective of our implementation is to achieve maximum flexibility when dividing the network under simulation into fluid networks and packet networks and deploying foreground packet traffic in the network.

Figure 2(a) shows a network topology under hybrid simulation. Circled by the dotted oval is the fluid network simulated by the fluid model solver. The grey nodes in the fluid networks are points where packets enter or leave the fluid network. These nodes are duplicated in the form of ns node objects and serve as *access points* to the fluid networks. Figure 2(b) is the structure of the corresponding ns-2 network topology which consists of a fluid model solver and six ns node objects. The access points are connected to the fluid model solver via specially designed 'fluid links'. These fluid links are virtual links that have infinite service capacity and no propagation



(a) A network under hybrid simulation



(b) Structure of the hybrid simulation configuration

Fig. 2. Fluid Model in a Hybrid Simulation in ns-2

delay. Their only function is passing packets to the fluid model solver and vice versa. With this design, the network can be divided arbitrarily into fluid networks and packet networks and the foreground packet traffic can reach anywhere in the network.

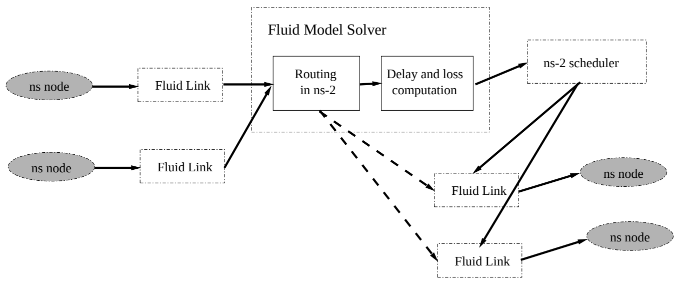


Fig. 3. Connecting the Fluid Model and NS Node Objects

In our current implementation of the hybrid simulation, we assume that the paths between two access points in the fluid network are known and statically pre-configured. When a packet is routed to pass across the fluid network, it is first sent to one of the access points. The access point knows from its routing table that it should send the packet to the 'fluid link', which directly passes the packet to the fluid model solver and identifies the ingress point. The fluid model solver is an extension of an ns node object. After receiving a packet from the fluid link, the routing mechanism embedded in the ns node object is able to tell the next packet level node that the packet is going to arrive, which is also the egress point of the packet from the fluid network. Thus the two access points (the ingress point and the egress point) of the packet is known and so is the path of the packet in the fluid network. When the delay and the drop probability of the packet is computed, the packet is scheduled to arrive at the fluid link that connects to the egress point ns node on the path of the packet. This procedure is shown in Figure 3.

VI. Experimental Results

We have carried out extensive experiments to test the accuracy and performance of the hybrid simulation approach. The accuracy and efficiency are evaluated by comparing the simulation results given by the hybrid simulation with those obtained in a packet level simulation. Five sets of experiments are presented to answer the following questions:

- How should the smoothing interval length be set?
- How does the one-pass model perform in terms of its accuracy?
- How does the background TCP traffic in the fluid network interact with traversing unresponsive packet flows? These flows might be used to model video or audio flows running on top of UDP.
- How do the TCP flows simulated by the fluid model compete with TCP flows simulated by a packet source when they share the same bottleneck queue? The fluid model has been shown to be accurate in describing the behavior of TCP networks, so the question is to judge its accuracy when the flows interact with real packet TCP flows.
- Does the two-pass model hybrid simulation capture the interactions between multiple traversing foreground flows?
- Do hybrid simulations scale?

In Section IV, we've proved an upperbound of the smoothing interval for the two-pass model. In the simulation, the smaller the smoothing interval, the more accurate the packet arrival rate is transformed into sending rate of the foreground flows in the fluid networks, which gives more accurate simulation results. On the other hand, smaller smoothing interval may cause more frequent interactions between the fluid model solver and the simulator. We test this potential cost that small interval may bring to the experiment. The result is that decreasing the smoothing interval doesn't bring any obvious increase in the simulation cost. This can be explained by the fact that, in our implementation, the extra work brought by deceasing the smoothing interval is an proportional increasing of synchronizing events and measuring the packet arrival rate.

The first part grows in reverse proportion to the smoothing interval, but its cost is negligible compared to the computation cost of the fluid model solver and packet simulation. The second part is a cheap division operation. In our case, the smoothing interval is set to be the same length as the fluid model solver step size since a smoothing interval smaller than the model solver step size is meaningless.

The first set of experiments will explore the accuracy of the one-pass model as we increase the number of packet flows sharing a bottleneck link with background TCP fluid flows.

The remaining experiments are all performed using the two-pass model. The second experiment observes a class of background TCP flows in a single bottleneck topology interacting with a UDP source whose sending rate changes with time. The accuracy is tested by comparing the delay and drops of the UDP traffic and the TCP network behavior given by the hybrid simulation and the packet simulation. Then, we present a set of simulation results showing the interaction between the background TCP traffic and different number of traversing foreground packet TCP flows. These results show that when sharing a bottleneck queue, the TCP flows modelled by the fluid model have the same competing capability for bandwidth as those simulated by the packet TCP sources, which further proves the correctness of both the hybrid simulation and the original fluid model. After that, simulation results of a hybrid simulation which consists of multiple foreground flows are given. In the last experiment, we show the capability of the hybrid simulation by simulating a network with more than 3,000 nodes and thousands of TCP classes consisting up to 12,378,340 TCP flows.

All the experiments are performed multiple times and yield similar results. In all the experiments, we use TCP Newreno and RED with ECN marking as the AQM policy. The step size of the fluid model solver is fixed at $1ms$ except for the last experiment where the step size is set to $5ms$. More results are available to interested readers.

A. Accuracy of One-pass Model

The setting of the experiment is shown in Figure 4. The six queues on the top are simulated in the fluid model and the other four queues on the bottom are simulated in packet form. $B1$ and $B2$ are access points between the fluid network and the packet network. The queue between $B1$ and $B2$ has a capacity of $100Mbps$ and a propagation delay of $10ms$ and is the bottleneck. Other queues have a capacity of $200Mbps$ and a $10ms$ propagation delay. Class 0 and Class 1 are classes of TCP flows from node $S1$ to $D1$ and from node $S2$ to $D2$ respectively. Class 0 is the background TCP traffic simulated in the fluid model and Class 1 is the foreground packet traffic.

We perform the experiment 5 times. Each time there are 40 TCP flows going through the bottleneck queue and the 40 flows are divided between Class 0 and Class 1. In the k^{th} time experiment, Class 0 contains $40 - k$ TCP flows and Class 1 contains k TCP flows. The TCP flows start at time 0 and ends at time $100s$. We measure the throughput of the flows from

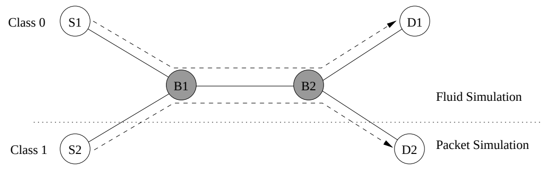


Fig. 4. Network with a Single Bottleneck

k	1	2	3	4	5
Hybrid (Class 1 Throughput)	18309	36646	57331	75736	104654
Packet (Class 1 Throughput)	18203	35752	54307	71512	92441
Error	0.58%	2.50%	5.57%	5.90%	13.21%

TABLE I
ACCURACY OF ONE-PASS MODEL

$30s$ to $90s$. For comparison, we perform experiments with the same scenario in a packet level simulation.

Table I compares the throughput of the packet TCP flow(s) (Class 1) in a hybrid simulation with that given by the packet simulation. In all cases, the throughput given in the hybrid simulation is larger than that in the packet simulation. However, we can see that when the packet traffic is small, the one-pass model can be pretty accurate and in these cases, generally the error is under 10%. As the fraction of the packet traffic increases, there is an increasing trend in error.

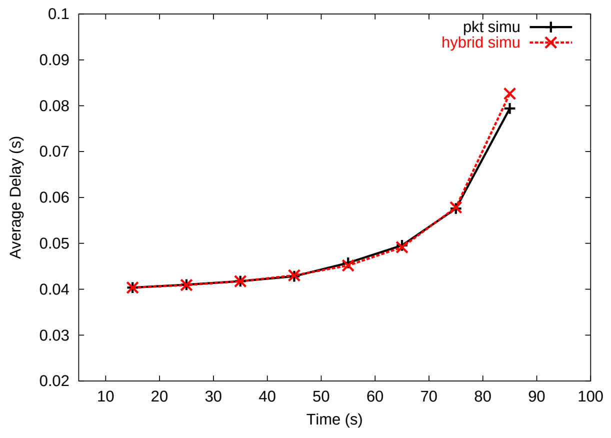
We will come back to this same scenario in the coming experiments and show how the two-pass model performs.

B. Interaction with UDP Traffic

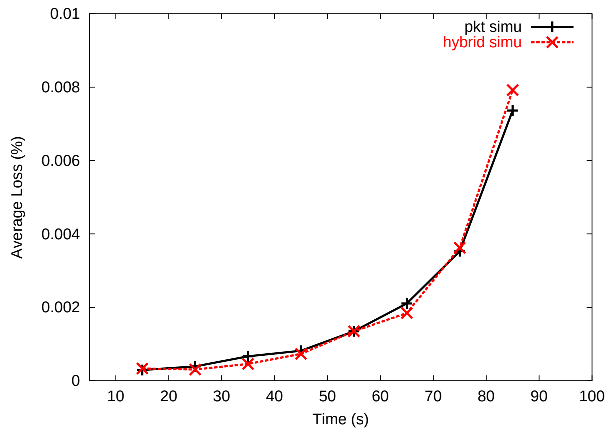
This experiment is to test the accuracy of the hybrid simulation when simulating the interaction between the background TCP traffic and a UDP traffic whose sending rate changes with time.

The same network setting in Figure 4 is used. This time, Class 0 is a class of 10 TCP flows from node $S1$ to $D1$ and Class 1 is UDP traffic from node $S2$ to $D2$. A $100s$ simulation is performed. The rate of the UDP traffic is $10Mbps$ at time 0 and increases to $20Mbps$ at $10s$, $30Mbps$ at $20s$, $\ldots$, until $90Mbps$ at $80s$. The packet size is set to 1000 bytes.

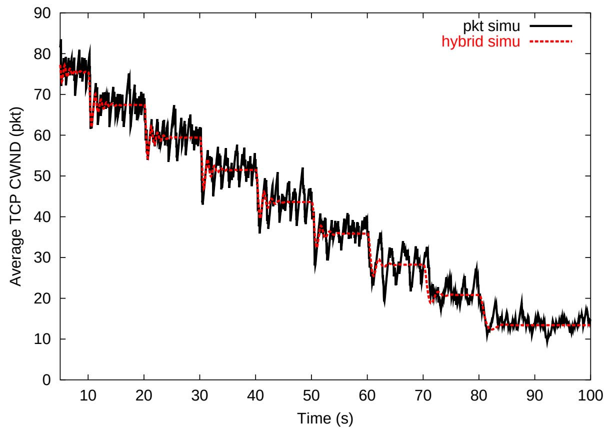
Figure 5(a) and 5(b) compare the average delay and drop probability experienced by the UDP traffic in different simulation time slots given by the hybrid simulation and the packet simulation. In both cases, they show a good match. This implies that the hybrid simulation is able to predict the behavior of a UDP foreground flow when it traverses the background TCP traffic. Figure 5(c) and 5(d) show the behavior of the TCP window size for Class 0 and the queue behavior of the bottleneck link. In the hybrid simulation, these results are given by the fluid model, which predicts the average behavior. We can also see a close match in these results. This accuracy of the fluid model predicting the average behavior is also reflected by the correct estimation of the average delay and drop probability that the UDP traffic experience. It shows



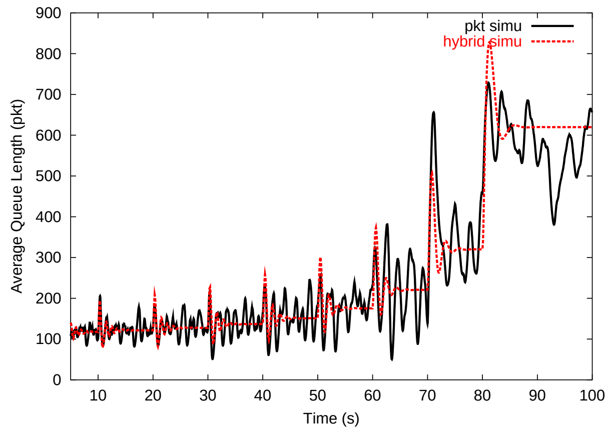
(a) UDP Average Delay



(b) UDP Average Loss



(c) TCP Window Size



(d) Bottleneck Queue

Fig. 5. Hybrid Simulation between the Fluid Model and UDP Traffic

that while the hybrid simulation can provide accurate packet level information for foreground packet flows, it can also estimate the average behavior for background TCP traffic and queues in the fluid network.

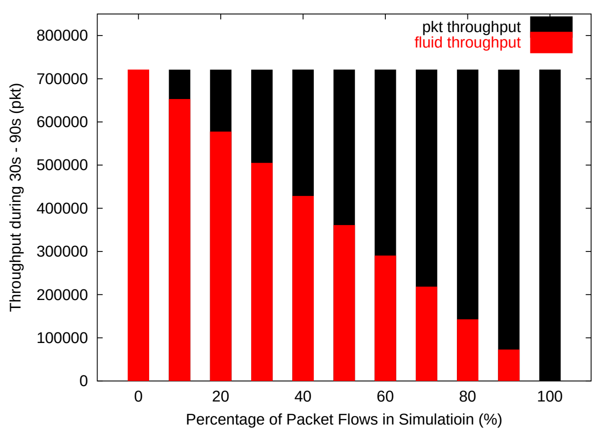
C. Interaction with TCP Traffic

We come back to the first experiment here. Now, we illustrate the accuracy of the two-pass model with the same setting as in the first experiment except that the packet traffic in this experiment takes more fraction of the total TCP flows.

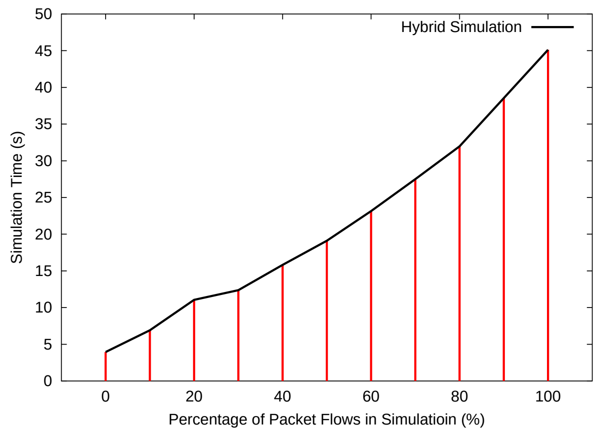
The experiment is carried out 10 times. Each time there are 40 TCP flows going through the bottleneck queue and the 40 flows are divided between Class 0 and Class 1. In the k^{th} time experiment, Class 0 contains $4*(11-k)$ TCP flows and Class 1 contains $4*(k-1)$ TCP flows. So the fraction of the packet traffic increases proportionally to k and the experiment is a fluid model simulation at the first time.

We record the throughput of the bottleneck from $30s$ to $90s$ in simulation and compare the computation time of each simulation. These experiment results are shown in Figure 6(a), 6(b) and Table II. In Table II, the first row shows the throughput of Class 0, which is the TCP class simulated by the fluid model. The second row shows the throughput of Class 1, which are TCP flows simulated by the packet simulator. The third row is the total throughput of the two and the forth row is the percentage of the packet throughput in the total throughput. As a reference, the simulation results given by a packet simulation are shown at place where 100% of the flows are packet flows. Figure 6(c) 6(d) and 6(e) show the average TCP window sizes of class 0 and class 1 and the bottleneck queue behavior in both hybrid simulation and peer packet simulation when both Class 0 and Class 1 contain 50% traffic (20 TCP flows) and evenly share the bottleneck service capacity. In Figure 6(c) and 6(e), the comparison is between solutions of the fluid model used in the hybrid simulation and those given by a packet simulation.

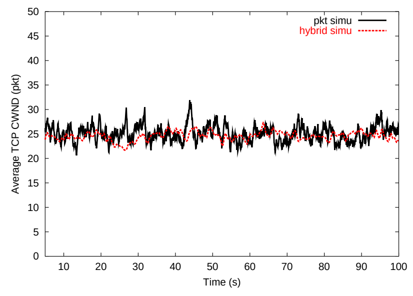
From these results, we see that the percentage of the packet throughput increases in proportion to its percentage of the total number of TCP flows, which strongly supports that



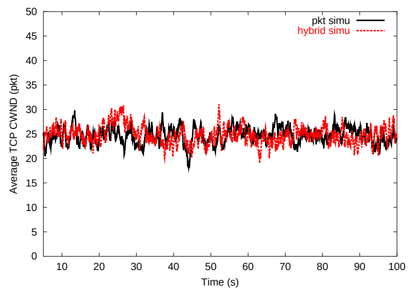
(a) Bottleneck Throughput Between 30s-90s



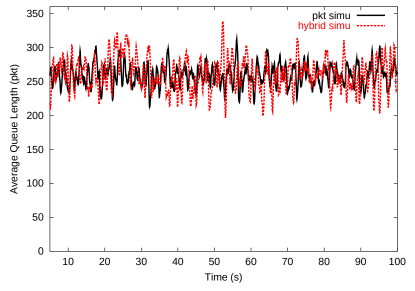
(b) Efficiency of the Hybrid Simulation



(c) TCP Window Size of Class 0



(d) TCP Window Size of Class 1



(e) Bottleneck Queue

Fig. 6. Interaction Between Fluid TCP Flows and Packet TCP Flows

k	1	2	3	4	5	6	7	8	9	10	pkt
Fluid	721153	652906	577602	505252	428611	361022	290473	218446	142799	72899	0
Packet	0	68211	143526	215763	292512	360166	430703	502700	578332	648228	721168
Total	721153	721127	721128	721015	721173	721188	721176	721146	721131	721127	721168
Pkt/Tot(%)	0	9	20	30	41	50	60	70	80	90	100

TABLE II
INTERACTION BETWEEN FLUID TCP FLOWS AND PACKET TCP FLOWS

the TCP flows simulated by the fluid model have the same behavior as those simulated by the packet source when sharing a bottleneck queue in a hybrid simulation. This further proves the correctness of the original fluid model and the hybrid simulation's ability to simulate the interaction between TCP flows in both fluid form and packet form. Also from this experiment, we can see the performance advantage brought by the fluid model from Figure 6(b). Compared with a packet simulation, the speedup of this hybrid simulation depends on the amount of the packet traffic in the hybrid simulation. If C_{pkt} denotes the packet simulation cost, C_{hf} denotes the fluid model solver computation cost in a hybrid simulation and C_{hp} denotes the simulation cost for the packet traffic in the hybrid simulation, there is

$$Speedup = \frac{C_{pkt}}{C_{hf} + C_{hp}}. \tag{13}$$

And as the simulation cost of the packet traffic in a hybrid simulation with small network topology is almost the same as that in the packet simulation, the fraction of the packet traffic decides the upper bound of the speedup. For example, if 10% of the traffic is packet traffic, the speedup can goes up to at most 10. In this experiment, we see that when 10% of the traffic is packet traffic, the hybrid simulation reaches a speedup of 6.53 compared to the packet simulation.

D. Multiple Foreground Flows

In this experiment, we show that the two-pass model hybrid simulation captures the additional interaction between multiple traversing foreground flows.

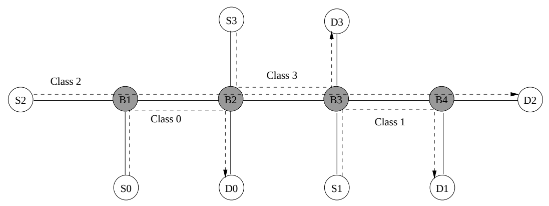


Fig. 7. Network with Three Bottlenecks

The network under simulation is shown in Figure 7. Node S2, D2, S3 and D3 are four packet nodes connected to the fluid network via different access points. Class 0 and Class 1 are both classes of 10 TCP flows simulated in the fluid model. Class 2 is a class of 20 TCP flows. Class 3 is a class of 40 TCP flows. Class 0, 1 and 2 start at time 0. Class 3 are only active from $30s$ to $60s$.

Figure 8 shows the matching simulation results between the hybrid simulation and the packet simulation. At $30s$, the average TCP window size of Class 2 decreases because of the new traffic brought by Class 3, which is another foreground flow, and the queue behavior at each bottleneck also changes accordingly. At $60s$, the network behavior changes back because Class 3 stops sending traffic.

E. Experience with Large IP Networks

In this section, we show the ability of the hybrid simulation to simulate a large IP network. We use the Inet Topology Generator from University of Michigan [14] and generate a network of 3500 nodes and 11334 links, each with a capacity of $2.5Gbps^2$. Then we randomly create 5000 classes of TCP flows which is a total of $12,378,340$ TCP flows in the network. Then we randomly pick two nodes as the source and the destination of a UDP traffic. We set the UDP traffic as a CBR at $50kbps$ with a packet size of 50 bytes. A $100s$ simulation is performed. Our experiment is carried on a Dell Precision Workstation 530, which is configured with two Pentium IV processors(2.2GHz) and 2GB memory. Since our program is not parallelized, only one processor is utilized. We have taken several trials. On average, the simulation takes about 30 minutes and $659M$ memory. Figure 9 shows the time sequence of the UDP packets between time $74s$ and $75s$ in one of the experiments. This UDP traffic has an average delay of $101.66ms$ and an average drop probability of 15.70%.

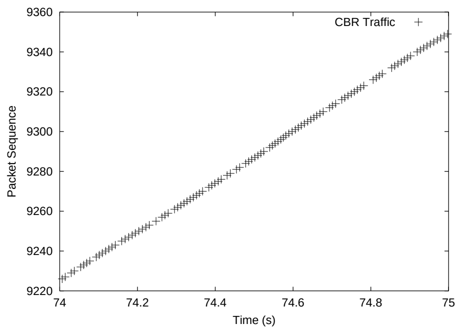


Fig. 9. Time Sequence of the UDP Traffic Between $74s$ and $75s$

VII. Conclusion & Future Work

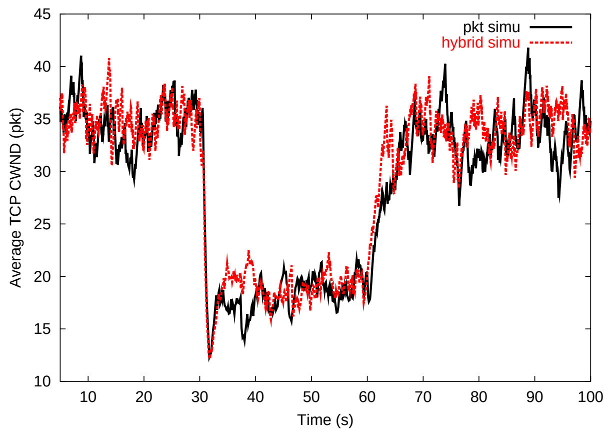
In this paper, we develop a simulation method that takes the advantage of the highly efficient fluid model and at the same time provides detailed information at packet level for selected individual traffic flows. We achieve this by an effort to simulate the network using fluid model solvers and discrete event-driven simulators, the hybrid simulation. Two models are proposed to describe the interactions between the background TCP traffic and the foreground packet traffic. Synchronization between the fluid model solver and the simulator are analyzed. Simulation results show that our method maintains the performance advantage of the fluid model and generates accurate simulation results comparable to those given by the discrete event-driven simulator. Our work can be reached and downloaded at `http://www-net.cs.umass.edu/fluid/ffm.html`.

As a future work, we will implement our hybrid simulation approach in more simulators and at the same time, incorporate new features of the fluid model as well as other network traffic models. Also, our current implementation in ns-2 could be further improved, such as supporting more functions and increasing its performance. Another exciting future work direction is to further boost the speed of the hybrid simulation such that the hybrid simulation can generate real time traffic information. This information can be used in network emulators to provide more controlled and more realistic delay and loss reference. Our current approach can handle real time traffic to some extent. Further performance boost can be reached by parallelization, for which the time-stepped nature of the fluid model solver is perfectly suitable.

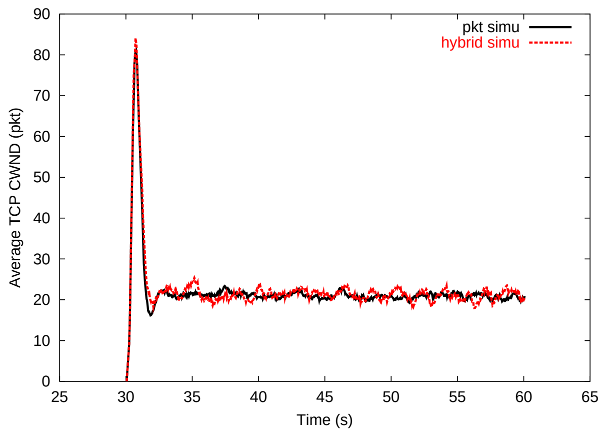
[2] We made some modifications so that all the links have a propagation delay larger than $5ms$, which is the smoothing interval used in the two-pass model hybrid simulation.

References

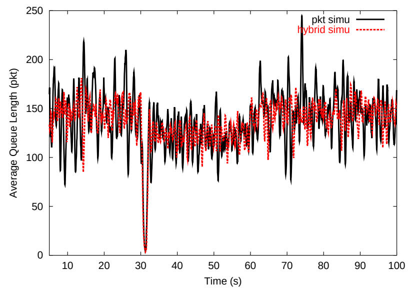
[1] "The Network Simulator - ns-2," http://www.isi.edu/nsnam/ns/.
[2] "GloMoSim - Global Mobile Information System Simulation Library," http://pcl.cs.ucla.edu/projects/glomosim/.
[3] "QualNet," http://www.scalable-networks.com.
[4] "Parallel and Distributed NS," http://www.cc.gatech.edu/computing/compass/pdns/.
[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 *Proceedings of ACM/SIGCOMM*, 2000.
[6] Y. Liu, F. L. Presti, V. Misra, D. Towsley, and Y. Gu, "Fluid models and solutions for large-scale ip networks," in *Proceedings of ACM/SIGMETRICS '2003*, June 2003.
[7] D. M. Nicol and G. Yan, "Discrete event fluid modeling of background tcp traffic," Department of Computer Science, Dartmouth College, Tech. Rep., To appear at ACM Transactions on Modeling and Computer Simulation 2004.
[8] S. Bohacek, J. P. Hespanha, J. Lee, and K. Obraczka, "A hybrid systems modeling framework for fast and accurate simulation of data communication networks," in *Proceedings of ACM/SIGMETRICS '2003*, 2003.



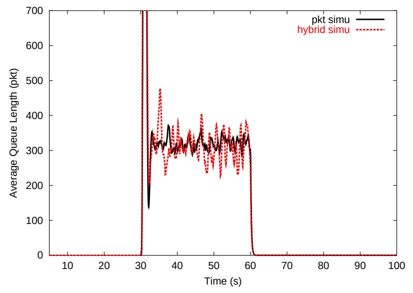
(a) TCP Window Size of Class 2



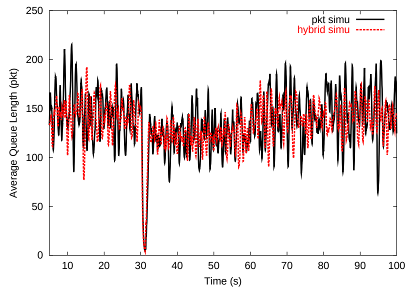
(b) TCP Window Size of Class 3



(c) Queue of Bottleneck 1



(d) Queue of Bottleneck 2



(e) Queue of Bottleneck 3

Fig. 8. Hybrid Simulation with Multiple Foreground Flows

[9] B. Melamed, S. Pan, and Y. Wardi, “Hybrid discrete-continuous fluid-flow simulation,” in *Proceedings of ITCOM 2001, Scalability and Traffic Control in IP Networks*, August 2001.

[10] G. Riley, R. Fujimoto, M. Ammar, K. Permula, and D. Xu, “Distributed network simulations using the dynamic simulation backplane,” in *Proceedings of International Conference of Distributed Computing Systems*, 2001.

[11] T. Yung, J. Martin, M. Takai, and R. Bagrodia, “Integration of fluidbased analytical model with packet-level simulation for analysis of computer networks,” 2001. [Online]. Available: citeseer.nj.nec.com/yung01integration.html

[12] C. V. Hollot, V. Misra, D. Towsley, and W.-B. Gong, “On Designing Improved Controllers for AQM Routers Supporting TCP Flows,” in *Proceedings of IEEE/INFOCOM*, April 2001.

[13] S. Floyd and V. Jacobson, “Random early detection gateways for congestion avoidance,” *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397–413, 1993. [Online]. Available: citeseer.nj.nec.com/floyd93random.html

[14] J. Winick and S. Jamin, “Inet-3.0: Internet topology generator,” 2002. [Online]. Available: citeseer.nj.nec.com/526211.html