Network Simulation via Hybrid System Modeling: A Time-Stepped Approach

Amogh Kavimandan[†], Wonsuck Lee[‡], Marina Thottan[‡], Anirudha Gokhale[†], and Ramesh Viswanathan[‡]
[†]Department of EECS, Vanderbilt University, Nashville, TN 37235
[‡]Bell Laboratories, Lucent Technologies, Murray Hill, NJ 07920

***Abstract*— The ever increasing complexity of networks dramatically increases the challenges faced by service providers to analyze network behavior and (re)provision resources to support multiple complex distributed applications. Accurate and scalable simulation tools are pivotal to this cause. Currently, network modeling and simulation tools are based either on packet-based discrete event models or fluid models. Both are limited in that the performance of discrete event based simulators does not scale to large networks while fluid models cannot accurately capture the behavior of all phases of a network's dynamics. The recently proposed *hybrid systems model* for data communication networks shows promise in achieving performance characteristics comparable to fluid models while retaining the accuracy of discrete models. Using the hybrid systems paradigm, this paper provides contributions to the modeling of TCP behavior and the analysis/simulation of data communication networks based on these models. An important distinguishing feature of our simulation framework is a faithful accounting of *link propagation delays* which has been ignored in previous work for the sake of simplicity. Other salient aspects of our work include a new finite state machine model for a drop-tail queue, a new model for fast recovery/fast retransmit mode, a revised sending rate model, and an embedded time-out mode transition mechanism all of which employ a time-stepped solution method to solve the hybrid system network models. Our simulation results are consistent with well-known packet based simulators such as ns-2, thus demonstrating the accuracy of our hybrid model. Our future efforts will be directed towards studying and improving the computational performance of hybrid model based simulations.**

I. Introduction

Communication service providers use a variety of tools including visualization, analysis and simulations to track the performance of their networks and (re)provision existing and new applications. For example, the Network Simulator **ns-2** [2] is the most widely used packet-based simulation tool to understand network behavior. Simulators like **ns-2** are based on discrete event simulation of per-packet behavior in the network. Although these simulators provide accurate analysis of network behavior, they have a number of shortcomings when applied to the networks of today and the future, as outlined below:

- *Heterogeneous, multilayered networks* – where networks are made up of several layers starting with the physical layer on which are overlaid data link layers like ATM and MPLS followed by the Internet protocols like TCP and IP. Existing techniques to analyze network behavior are typically restricted to individual layers. For example, simulators like **ns-2** are particularly useful to analyze IP traffic. Thus, it is tedious and in many cases infeasible, with existing techniques, to determine the effect of disruptions at one layer across multiple layers of the network.
- *Networks of networks* – where the ever expanding size of the Internet has given rise to a large collection of interconnected but independently administered networks that include both public and private networks. Analyzing the behavior of these networks requires accurately modeling the structure, policies and behavior of the networks while also scaling to the large size of the networks. Current state of the art in network analysis do not provide these capabilities.
- *Computational complexity and scalability of discrete event simulators* – as the scale of the network increases it becomes computationally infeasible to use sequential packet-based discrete event simulators to analyze the networks. Moreover, the storage requirements for the trace data generated by these simulations become cumbersome to manage and mine.

Addressing the challenges outlined above requires a new approach to analyzing network behavior where the computational requirements must be kept manageable.

Hybrid system models combine the strengths of continuous and discrete models. A system is modeled as a discrete event system at a coarse granularity where a discrete state transition occurs when certain guard conditions are met. Within each state, however, a system is modeled to evolve according to some continuous dynamics.

II. Survey of Work on Network Modeling & Simulation

Network modeling and simulation paradigms can be categorized into three mainstream approaches: packet-level models, flow-level models, and hybrid models.

Packet-level simulation has been the most widely used simulation methodology by the network research community. Packet-level simulators are based on an event-logic driven simulation paradigm. For example, **ns-2** [2], **SSFNet** [3] and **JSim-INET** [4], [5] fall into this category. Packet-level simulators keep track of all the individual packets in the network. Therefore, they offer highly accurate and detailed results; however, the computational complexity increases rapidly as the size of the network becomes large.

During the past decade fluid-level simulations have been studied extensively and continue to be researched actively. In

the framework of fluid models, a chunk of packets is modeled as a single continuous artifact, namely, a fluid. The dynamics of the network is then represented by a system of ordinary differential equations, which describe fluid rate changes inside the network. Most of the research efforts in this area [9], [6], [7], [8] have focused on modeling TCP protocols [10]. In this framework, the arrival and the loss of packets are often modeled as stochastic processes [8], [6]. The traffic source and queue dynamics constitute a system of equations which governs the TCP controlled data transfer over an IP network.

Recently, hybridization of the packet and the flow modeling approaches have been studied. Guo *et. al.* proposed the Time-stepped Hybrid Simulation (TSHS) [11] method to deal with the scalability issue faced by traditional packet-level discrete-event simulation methods. TSHS considers a chunk, grouped from the packets that are in the same time-step, as a unit entity. The packets in a chunk are assumed to be evenly spaced within the time-step. In order to identify any event, every node in the network is checked at every time-step. TSHS is capable of offering flexible choice in simulation fidelity based on the simulation abstraction level. With a slightly different objective, Gu *et. al.* studied the integration of a fluid model with a packet-level simulation to maintain both the performance advantage as well as providing details for selected packet traffic [12]. They proposed models that can account for the interaction between flow-based background TCP traffic and foreground packet-based traffic flows. Another approach called Time-Driven Fluid Simulation (TDFS) [13] models traffic nodes in a network as fluid servers which process workload continuously. In TDFS models, traffic sources may employ either continuous or discrete event models. Using TDFS framework, Yan *et. al.* studied a single class of traffic. They also proved that the discretization error does not depend on the network size but is determined by the length of the discretization interval used. TDFS can also be easily parallelized to improve computational performance. All the above mentioned hybrid approaches show promising avenues for network simulation but have not yet been used as an alternative to **ns-2**.

A newly proposed hybrid system modeling framework [14], which is different from the aforementioned hybridization approaches, describes continuous dynamic behavior of a network within an organized finite state machine (FSM) formalism. Bohacek *et. al.* proposed such a framework for data communication networks and studied various communication protocols using hybrid systems [15]. Bohacek *et. al.*'s work forms the basis for what we report in this paper. We note that this paper shares some common ground with Bohacek *et. al.*'s work but differs since we model explicitly the propagation delay and provide a more mathematically accurate hybrid system model of the TCP protocol.

III. Hybrid Systems Modeling of TCP Networks

The hybrid system modeling approach for data communication network simulation is motivated by the drawbacks of the discrete-event driven packet level simulations and the flow-level simulations. Though the packet level simulators provide the most detailed network simulation, they often suffer from scalability issues. It is also difficult to analyze network performance related to combinations of network parameters. The flow-level simulation models, on the other hand, statistically average quantities for better computational performance and are capable of providing information on network parameters and their impact on network performance. However, accounting for network dynamics over the entire duration of the simulation experiment is difficult to model naturally in the fluid modeling framework because discrete phase changes may be caused instantaneously by external events, such as a packet drop.

The hybrid system modeling approach [14] proposed by Bohacek *et. al.* effectively reduces the complexity of network simulations by using flow-level models in each state of the FSM description. Logic based events trigger transitions between states to capture the discrete nature of queue dynamics.

In this section, we describe the details of a hybrid system TCP model that we have developed. We focus on a TCP New-Reno implementation for the source, a drop-tail queuing policy, and we model their interactions using FSM models.The aspects of our model that differs from Bohacek *et. al.*'s work will be highlighted. Our solution method differs from Bohacek *et. al.*'s in that we use the time-stepped solution method to solve the constructed hybrid system. This approach resembles the solution techniques used in TSHS [11].

A. Mathematical Model of a Network

Mathematically a network is a directed graph $(\mathcal{N}, \mathcal{L})$ where a set of vertices $\mathcal{N}$ denotes nodes and a set of edges $\mathcal{L}$ denotes a collection of links connecting two nodes. For instance, two nodes $u \in \mathcal{N}$ and $v \in \mathcal{N}$ are connected by a link $l = l_{u,v} \in \mathcal{L}$. Three real positive numbers are assigned to each link $l \in \mathcal{L}$, these are: bandwidth (link-rate), propagation delay, and maximum queue size. Let B^l, τ^l, and $q^l_{\max}$ denote the bandwidth, the propagation delay, and the maximum queue size of a link l, respectively. These three parameters characterize each link in a network. We remark that the *propagation delay* has not been included in the previous study [14], hence only simplified topological settings were studied.

Let $\mathcal{F}$ represent a set of *flows*. Each flow $f \in \mathcal{F}$ is associated with a source node $u_s \in \mathcal{N}$ and a destination node $u_d \in \mathcal{N}$ where f flows from u_s to u_d. We assume that flow f is generated and enters node u_s with an incoming rate a_f at a prescribed time $t = t^0_f$. Once the network is loaded with data flows, all or part of the links become path ways for data flows. A flow f on a link l can be quantified by a link-in-rate (a sending rate), s^l_f, and a link-out-rate (an arrival rate), a^l_f. Bohacek *et. al.* named s^l_f as *l-link/f-flow transmission rate* and a^l_f as *l-link/f-flow arrival rate* [14].

Note that we assume the source and the destination nodes are unchanged over time for a given flow f. However, intermediate nodes, that flow f passes through, may dynamically change in time but at a much slower scale compared to the data transmission rates. Hence network dynamics can be well defined on top of any flow path changes within network.

Bandwidth B^l imposes an upper bound on flow sending rate

on a link l.as follows:

$$\sum_{f\in\mathcal{F}} s_f^l \le B^l, \qquad \forall\, l \in \mathcal{L}\ . \tag{1}$$

Here $s_f^l = s_f^l(t)$ is a dynamically evolving quantity.

In our model, a queue is associated with a link $l \in \mathcal{L}$. q_f^l denotes the size of the flow f in the queue of a link l. Therefore, given the maximum capacity of the queue $q_{\max}^l$ of the link l, the following inequality should hold for all times.

$$\sum_{f\in\mathcal{F}} q_f^l \le q_{\max}^l, \qquad \forall\, l \in \mathcal{L}\ . \tag{2}$$

Here q_f^l is also a function of time, *i.e.*, $q_f^l = q_f^l(t)$.

In general the transmission delay is composed of link propagation delay, processing delay, and queuing delay. Associated with each link l, we only consider a fixed propagation delay τ^l and a queuing delay q^l/B^l where $q^l = \sum_{f\in\mathcal{F}} q_f^l$. For the sake of simplicity, we assume the processing delay is minimal compared to the other two terms.

Round-trip-time (RTT) is a measure of the current delay on a network or time elapsed to receive an acknowledgment packet (ACK) from the destination node. To compute RTT for a flow f, RTT_f, the queuing delay and the propagation delay of the original data packet plus the propagation delay of the ACK is considered. In other words, we ignore the queuing delay of the ACK from the destination node to the source node. If $\mathcal{L}_f$ is an ordered set of links that flow f takes on its way to the destination, RTT_f becomes

$$RTT_f = \sum_{f\in\mathcal{L}_f} (2\tau^l + \frac{q^l}{B^l})\ . \tag{3}$$

Here RTT_f and q^l are functions of the time variable t and the coefficient 2 in front of τ^l accounts for propagation delay of the original packet and the ACK. We note that $RTT_f(t)$ is the estimation of RTT at certain time t. In a real system a packet sent at time t may experience queuing delay different from $q^l(t)/B^l$ at the link l since the packet will exit link l at a certain time past t.

B. The TCP Protocol

In order to study transient behavior of a network, modeling TCP protocols has been our first goal since not only is a significant portion of Internet traffic controlled by the TCP protocol but also it potentially possesses rich dynamics due to the concept of varying the sending rate in response to the network conditions.

The modes of TCP-New Reno policy include slow-start, fast-recovery/fast-retransmit (fr/fr), congestion-avoidance and time-out. The TCP-New Reno algorithm chooses a certain mode based on the limited information it holds and has received from the destination host.

Data transfer starts in the slow-start mode with congestion window size, w, equal to 1. The source limits the number of sending packets below w minus the number of outstanding (unacknowledged) packets. Once the transmitted packet arrives at the destination host, the receiver returns a small acknowledgment packet (ACK) with information containing the sequence number of the next expected packet. If the ACK is successfully received by the sender, the sender increases w by ACK so that it has more room to send a new packet. If the packet is lost or the order of the packets are altered during transmission, the sender may receive duplicate acknowledgments (dupACKs) or may not receive any ACKs.

If the sender receives triple dupACKs, the source enters fr/fr mode. Initial congestion window size in fr/fr mode is given by $w^-/2+3$ where w^- is the congestion window just before fr/fr mode is entered. A variable called slow-start threshold ($ssthr$) is set to $w^-/2$ which is the maximum congestion window size allowed in the next slow-start mode, if there is any. In TCP-New Reno policy, the fr/fr mode ends if all the lost packets are successfully retransmitted. Subsequently, the congestion avoidance mode follows.

In the congestion avoidance mode, unlike in slow-start mode, every ACK increases the congestion window size by a small portion of the congestion window, usually it takes a form of $w^{\text{new}} = w^{\text{old}} + 1/w^{\text{old}}$. Normally, a source honors the maximum advertised congestion window value satisfying the following inequality: $w \le w^{\max}$. From the congestion avoidance mode, depending on number of dupACKs, the subsequent mode is either fr/fr or time-out mode which is discussed next.

If the sender has not received any ACKs, it will not be able to send any new packets and the time-out timer will eventually expire. The transmission effectively ceases during this waiting period. After the time-out period is over a new slow-start mode is re-initiated with $w = 1$ and $ssthr = w^-/2$ where w^- is the congestion window after the last ACK was received.

C. Source Dynamics

In the following subsection, we revisit Bohacek *et. al.*'s hybrid TCP source model [14]. A new model for fr/fr mode is introduced along with a more accurate sending rate formula. Transition to time-out mode is naturally embedded without further modeling effort. We start with the continuous time (CT) domain model of slow-start mode. Note that our primary unknown in CT domain of source dynamics is the congestion window size. The sending rate is a derived quantity from the congestion window size variable.

1) Slow-Start:

The congestion window size w_f of a given flow $f \in \mathcal{F}$ is governed by the following ordinary differential equation:

$$w_f'(t) = \frac{\log m_{ss}}{RTT_f(t)} w_f(t)\ , \tag{4}$$

where $'$ denotes the time derivative d/dt, m_{ss} is a multiplicative constant such that w_f is being multiplied by m_{ss} every RTT, and RTT_f is computed from Eqn. (3). w_f is initialized to 1 whenever the source enters slow-start mode. Hence, in slow-start CT domain we solve the initial value ODE problem. Derivation of Eqn. (4) is given in the Appendix section.

In TCP flow control w_f is used to limit the number of packets to be sent out from the sender. Because the sender is able to send more packets only when the ACKs arrive, effectively, only w_f number of packets are sent in RTT_f time

window. With the assumption that the packets (or flow f) are sent over RTT_f period with evenly spaced pattern, the instantaneous sending rate s_f should be

$$s_f(t) = \frac{w_f(t)}{RTT_f(t)} \ . \tag{5}$$

Here we should note that the evenly spaced packets over RTT_f time period is a rather strong assumption. It may be a good model when traffic flow is up and running for some time so that ACKs become spread over time due to a certain random process. However, when traffic is at its early stage, like the very first slow-start mode, TCP tends to burst out traffic. In such cases the sending rate model, Eqn. (5) might be poor. We revisit this question in the discussion section.

Bohacek *et. al.* used a coefficient β to adjust the sending rate formula as follows:

$$s_f(t) = \frac{\beta w_f(t)}{RTT_f(t)} \ , \tag{6}$$

where $\beta = 1.45$ is obtained from the trace comparison against **ns-2** simulation. However, we argue that it might be an artifact from the hybrid simulation paradigm employed in Bohacek *et. al.*'s paper [14]. In the discussion section we will show how a CT-FSM (*i.e.* hybrid) model may introduce this error by deducing the value of β based on certain assumptions.

Though the congestion window size evolves with Eqn (4) within the slow-start CT domain, w_f should not grow over the advertised maximum window size $w_f^{\rm adv}$, which is assigned to the source host by the destination host at the initial hand-shake stage. Therefore we use Eqn. (5) with a slight modification. The model we use for instantaneous flow sending rate is

$$s_f(t) = \frac{\max\left\{w_f(t), w_f^{\rm adv}\right\}}{RTT_f(t)}. \tag{7}$$

2) Congestion Avoidance:
Congestion avoidance mode increases congestion window size much more cautiously. During the congestion avoidance mode every ACK increases congestion window size by a small portion of the current window size,

$$w_f^{\rm new} = w_f^{\rm old} + \frac{L}{w_f^{\rm old}} \ , \tag{8}$$

where L is a real positive number (normally $L = 1$). Since w_f packets are sent every RTT, if the congestion avoidance mode is maintained during this time window, w_f packets will be returned. Therefore w_f is increased by 1 every RTT if $L = 1$. The ODE model that governs the congestion window size evolution in the congestion avoidance mode is

$$w_f'(t) = \frac{L}{RTT_f(t)} \ . \tag{9}$$

It is straight forward to deduce Eqn. (9) (see Appendix).

The model shows a linear increase of congestion window size in congestion avoidance mode as long as no drop is detected and $w_f \leq w_f^{\max}$ is satisfied. Occurrence of dupACKs or no ACK ends the congestion avoidance mode and is followed by fr/fr or time-outs.

3) Fast Recovery/Fast Retransmit:
In TCP-New Reno policy fr/fr mode is initiated whenever the sender receives triple dupACKs. These dupACKs do not increase the congestion window size in real TCP flow control. Therefore, it is not accurate if the state remains at either slow-start or congestion avoidance mode since they are governed by their own dynamic discipline which evolves the congestion window size in time.

In order to model congestion window evolution correctly over the state transition, the source should enter fr/fr once a drop is detected. However the actual transition should be delayed by the right amount of time to account for elapsed time between the drop event and the arrival of dupACKs in the real system.

Since each dupACK increases w_f by 1, the speed of window size increments is the same as that in the slow-start mode, *i.e.*, an exponential increase, unless there is a drop. Hence Eqn. (4) might well be applied as long as no drop occurs. However, in a real packet network a packet drop is usually followed by another drop because it takes a certain amount of time to send buffered packets out onto the out-bound link from a congested queue. In that event, a hybrid system may experience too many state transitions in a short time epoch and the model becomes inefficient.

To overcome this modeling difficulty within the hybrid system framework, we propose an algebraic model for fr/fr mode. Along with it we provide a formula for the duration of fr/fr mode.

Let w_f^- be the congestion window size of flow f just before the source enters fr/fr mode and $t_0^{\rm fr}$ be the time when the source mode becomes fr/fr. Suppose $a_f^D(t)$ is the instantaneous arrival rate of flow f at destination node D at time t, $\zeta(t)$ is size of outstanding flow at time t and $ack(t)$ is the acknowledged flow size since $t = t_0^{\rm fr}$. Then we have following relations,

$$\begin{aligned} ack(t) &= \int_{t_0^{\rm fr}}^{t} a_f^D(t' - RTT_f/2)\ dt' \ , \\ \zeta(t) &= (w_f^- - w_f(t_0^{\rm fr} - RTT_f)) + \eta(t) - H(t) \ , \end{aligned} \tag{10}$$

for $t \geq t_0^{\rm fr}$. Here $(w_f^- - w_f(t_0^{\rm fr} - RTT_f))$ represents the initial outstanding flow size when the source enters fr/fr, $\eta(t)$ denotes size of flow newly transmitted since the initiation of fr/fr mode. If there is no drop in $[t - RTT_f(t), t]$, $H(t)$ is equal to $ack(t)$. Otherwise, if the source learns about the drop at $t_{\rm drop} \leq t$, H returns the value of $ack(t')$ provided no drop is detected in time period of $[t' - RTT_f(t'), t']$ for $t' \leq t_{\rm drop}$. Hence, function H does not count dupACKs in the events of drop during fr/fr mode. We remark that ζ approximates the outstanding flow size for the drop-tail queuing policy.

In fr/fr mode every ACK (either normal ACK or dupACK) increases the congestion window by 1 so the model for congestion window evolution is

$$w_f(t) = w_f^-/2 + ack(t) \ . \tag{11}$$

The instantaneous sending rate is different from Eqn. (5) since we do not allow the drop event to trigger state transition.

Our flow sending rate model is

$$s_f(t) = \frac{\max\left\{(w_f(t) - \zeta(t)), w_f^{\text{adv}}\right\}}{RTT_f(t)}. \tag{12}$$

Let n_f^{drop} be the cumulated flow size that are lost during the transmission. The increment of n_f^{drop} is then given by

$$n_f^{\text{drop}}(t) = \int_{t_0^{\text{fr}}}^{t} d_f(t')\ dt'\ , \tag{13}$$

where d_f is a drop rate which will be discussed in detail in the next section, Sec. III-D. Since TCP New-Reno only retransmits one probably-lost packet at one RTT, the decrement of n_f^{drop} is 1 at every RTT unless all the flow sent were lost. In the event of total loss n_f^{drop} should not be decreased. More precisely, for every RTT_f, if $ack(t) \neq ack(t - RTT_f)$, n_f^{drop} should be decreased as follows:

$$n_f^{\text{drop}}(t) = n_f^{\text{drop}}(t - RTT_f) - 1\ . \tag{14}$$

Here $ack(t) \neq ack(t - RTT_f)$ ensures that not all the flows were dropped during time interval of $[t - RTT_f, t]$.

In our TCP New-Reno implementation, the source exits fr/fr mode if n_f^{drop} becomes 0. Other TCP variants should differ in fr/fr dynamic behavior and the duration of fr/fr. For example, TCP Reno policy exits as soon as the fast retransmit of the first lost packet is successful. If two packets were dropped, another fr/fr mode follows immediately after the recovery (fr/fr mode) of the first lost packet. Therefore, multiple packet losses might lead the source mode to time-out by multiple steps of halving the congestion window size.

4) Time-out:

In our implementation, whenever a drop is detected, the source enters fr/fr. In fr/fr, if ACKs cease to return, the congestion window stays at a constant value and the number of outstanding packets is not decreased resulting in no new packets being sent for some time period.

Let t_{out} be a timer which is set to 0 when the source enters fr/fr. t_{out} keeps on increasing with the simulation clock. If t_{out} exceeds the given time-out period, say, RTO, then the source enters slow-start mode with $w_f = 1$ and $ssthr = w_f^-/2$. Note that if time-out occurs in the simulation, the fr/fr state entered exhibits no effect on congestion window dynamics because no new packets were sent and no ACKs were received by the sender.

D. Queue Dynamics

Suppose ω and ν are in-bound (west) and out-bound (east) links, respectively, in the traffic path of a flow f. Let $\mathcal{F}_\alpha$ be a set of such flows sharing the links ω and ν. We have the following equation describing the queue changing rate

$$\frac{d}{dt} q_f^\omega = a_f^\omega - s_f^\nu - d_f^\omega\ , \tag{15}$$

where q_f^ω is the size of flow f buffered at q^ω. a_f^ω is flow arrival rate from the link ω. s_f^ν denote flow sending rate into the link ν.

Suppose there is no packet loss during transmission over the links in a network, then incoming flow into a link ω must all exit from that link, that is, $a_f^\omega(t) = s_f^\omega(t - \tau_\omega)$ where τ_ω is the link propagation delay. Here $s_f^\omega(t - \tau_\omega)$ is readily available at time t.

Let $q_{\max}^\omega$ represent maximum queue size associated with the in-bound link ω. B^ν (B^ω) is a given fixed bandwidth of the link ν (ω).

Bohacek *et. al.* determined the values of s_f^ν and d_f^ω based on queue state. In their model, the FSM for a queue consists of three states, they are:

- Empty queue ($q^\omega = 0$)

$$d_f^\omega = 0, \quad s_f^\nu = \begin{cases} a_f^\omega \sum_{\bar{f} \in \mathcal{F}_\alpha} a_{\bar{f}}^\omega \leq B^\omega \\ a_f^\omega B^\omega / (\sum_{\bar{f} \in \mathcal{F}_\alpha} a_{\bar{f}}^\omega) \end{cases} \tag{16}$$

- Queue neither empty nor full ($0 < q^\omega < q_{\max}^\omega$ or $q^\omega = q_{\max}^\omega$ but $\sum_{\bar{f} \in \mathcal{F}_\alpha} a_{\bar{f}}^\omega \leq B^\omega$)

$$d_f^\omega = 0, \quad s_f^\nu = \frac{q_f^\omega}{\sum_{\bar{f} \in \mathcal{F}_\alpha} q_{\bar{f}}^\omega} B^\nu \tag{17}$$

- Queue full and still filling ($q^\omega = q_{\max}^\omega$ and $\sum_{\bar{f} \in \mathcal{F}_\alpha} a_{\bar{f}}^\omega > B^\omega$)

$$d_f^\omega = \frac{a_f^\omega (\sum_{\bar{f} \in \mathcal{F}_\alpha} a_{\bar{f}}^\omega - B_\omega)}{\sum_{\bar{f} \in \mathcal{F}_\alpha} a_{\bar{f}}^\omega}, \quad s_f^\nu = \frac{q_f^\omega}{\sum_{\bar{f} \in \mathcal{F}_\alpha} q_{\bar{f}}^\omega} B^\nu \tag{18}$$

There are two assumptions made to obtain the above queue model. They are, i) packet arrival rate does not change rapidly and is assumed to be constant in a short time interval, namely, *Arrival Uniformity* ii) packets of different flows are uniformly distributed, *Queue Uniformity*.

The governing equation (15) of the queue size and the finite queue state description (16) - (18) is a mathematically well-defined hybrid system. However, the number of state transitions can be numerous specially if $q^\omega \approx q_{\max}^\omega$ to detect zero-crossing for the state transition [15] in the hybrid system. Furthermore, a large system of ODE for q_f^ω for all $f \in \mathcal{F}_\alpha$ needs to be solved.

In fact, to find $q_f^\omega(t)$ from the previously known queue size $q_f^\omega(t_0)$ for $t_0 < t$, all that is required is the net increase of queue size. If we do not impose $q_f^\omega \leq q_{\max}^\omega$, we have

$$q_f^\omega(t) = q_f^\omega(t_0) + \int_{t_0}^{t} (a_f^\omega(t') - s_f^\nu(t'))\ dt' \tag{19}$$

It is simply an integral form of Eqn. (15) without the drop term. Here s_f^ν is the sending rate into the link ν, *i.e.*, the sending rate from the q^ω.

We now discuss our queue model. Unlike Bohacek *et. al.*'s model, the states are not determined by the queue size q_f^ω. Rather they depend on the bandwidth of the out-bound link, B^ν, and the size of the arriving and departing packet. For the

flow sending s_f^ν, we have,

$$s_f^\nu = \begin{cases} \frac{q_f^\omega}{\sum_{\bar{f}\in\mathcal{F}_\alpha} q_{\bar{f}}^\omega}, & \text{if} \quad \sum_{\bar{f}\in\mathcal{F}_\alpha} q_{\bar{f}}^\omega \geq B^\omega \\ a_f^\omega + q_f^\omega, & \text{if} \quad \sum_{\bar{f}\in\mathcal{F}_\alpha} (a_{\bar{f}}^\omega + q_{\bar{f}}^\omega) \leq B^\omega \\ q_f^\omega + \frac{a_f^\omega}{\sum_{\bar{f}\in\mathcal{F}_\alpha} a_{\bar{f}}^\omega} (B^\omega - \sum_{\bar{f}\in\mathcal{F}_\alpha} q_{\bar{f}}^\omega), & \text{o.w.} \end{cases} \tag{20}$$

Equations (19) and (20) completely describe the queue size and the flow sending rate changes in time. However, the sum of solutions $\sum q_f^\omega$ may be greater than $q_{\max}^\omega$ which should be understood as buffer overflow or flow drop.

The drop model should describe which flows suffer losses at the time a drop occurs. For example, one can drop a larger flow first or vice versa. A probability distribution $p(f)$ may be used to select a flow to drop, for instance, $p(f)$ can be

$$p(f) = \frac{a_f^\omega}{\sum_{\bar{f}\in\mathcal{F}_\alpha} a_{\bar{f}}^\omega} \ . \tag{21}$$

Another question is, if a flow is decided to be dropped, what is the right size of the flow that should be dropped. We propose the following model. Let $D_f^\omega(t)$ denote the size of dropped flow f, we model $D_f^\omega(t)$ as,

$$D_f^\omega(t) = \max\left\{0, \left\lceil \frac{a_f^\omega \left(\sum_{\bar{f}\in\mathcal{F}_\alpha} q_{\bar{f}}^\omega(t) - q_{\max}^\omega\right)}{\sum_{\bar{f}\in\mathcal{F}_\alpha} a_{\bar{f}}^\omega} \right\rceil\right\} \tag{22}$$

Eqns. (19) - (20) are not entirely different from the Eqn. (15) and Eqns. (16) - (18). Eqns. (19) - (20) take an integral form and s_f^ν is based on packet arrival rate a_f^ω and queue size q_f^ω instead of the queue state. Note that Bohacek *et. al.*'s drop rate d_f^ω is determined by queue state only, however, in our model, the value of the drop size (22) is a deduced quantity.

E. Solution Techniques

We have created the simulator without relying on any existing simulation frameworks, such as, Stateflow/Simulink [17], Modelica [18], Ptolemy [19], [20] etc. Therefore we have full control of the solution method of the hybrid system, numerical scheme to be used in CT domain, and scope of component model. Furthermore, it is easy to identify any computational bottleneck and the sources of inaccuracy in the model specific to the implementation.

In general FSM is not a timed model, however, its composition with CT domain requires detection of any events that may be triggered on the time axis. Each state in FSM is being refined during the continuous evolution of the sources and the queues in our model. At each time abscissa, which may be a priori chosen or dynamically updated, active states are evaluated. If the state transition condition is enabled, then the FSM makes corresponding transitions according to the condition and the system continues to evolve.

Assume that CT-FSM composition refers to a master clock for their time. Consider one tick of that master clock to be the time step Δt of our hybrid system simulation. It is trivial that a smaller Δt introduces small error in state transition. In our simulation, we do not allow FSM state transitions during an epoch of length Δt. Now, CT domain simulation can be performed as accurately as possible, say, over an interval $[n\Delta t, (n+1)\Delta t]$ where $n = 0, 1, \ldots$. At each time point, $n\Delta t$, current states are evaluated, checked against transition enablers, and changes are made to the states, as necessary. The tasks described above can be summarized in the following algorithm.

Algorithm for the Hybrid Simulation

Input:
- A network $(\mathcal{N}, \mathcal{L})$,
- Bandwidth of all the links $l \in \mathcal{L}$,
- Maximum queue size at all the queues,
- Set of source nodes $\mathcal{S}$,
- Set of destination nodes $\mathcal{D}$,
- Source wake up time and interval,
- Time-step h,
- End of simulation time T_{end}.

Construct $\mathcal{L}_f$ *for each* $f \in \mathcal{F}$
Set $t = 0$
while $t \leq T_{end}$ **do**
  $t = t + h$
  Wake-up sources as scheduled
  Solve equations for all $s \in \mathcal{S}$ *and* $q \in \mathcal{N}$
  for *all* $s \in \mathcal{S}$ **do**
    switch *mode(s):*
    case *(mode(s)=slow start):*
      if *cwnd(s)* $\geq$ *ssthr* **then**
        mode(s) $\overset{t+\text{delay}}{\longrightarrow}$ *c.a.*
      else if *flow f from s suffer drop* **then**
        mode(s) $\overset{t+\text{delay}}{\longrightarrow}$ *fr/fr*
      else
        remain in slow start mode
    case *(mode(s)=fr/fr):*
      if $n_f^{\text{drop}} = 0$ **then**
        mode(s) $\overset{t}{\longrightarrow}$ *c.a.*
      else
        remain in fr/fr
    case *(mode(s)=c.a.):*
      if *flow f from s suffer drop* **then**
        mode(s) $\overset{t+\text{delay}}{\longrightarrow}$ *fr/fr*
      else
        remain in c.a.
  end for
  for *all queue* **do**
    evaluate and set queue state for the next iteration
  end for
end while

Now we comment on a numerical scheme for Eqns. (4) - (9) which describes dynamics of the congestion window evolution. We have, in their general form,

$$w' = g(t, w), \quad w(0) = w_0$$

where w_0 is given.

It is well known that Euler's method [16] is not practical due to the requirements of small step size for reasonably accurate numerical solutions. Runge-Kutta methods do not require higher order derivative by evaluating $g(t, w)$ at selected points on each interval of computation. Higher order methods can be obtained by paying extra for the evaluation of g.

For example, Runge-Kutta method of order 2 employs the following recursive formula:

$$w_{n+1} = w_n + \tfrac{1}{2}(k_1 + k_2),$$
$$\text{where} \quad k_1 = hg(t_n, w_n),$$
$$k_2 = hg(t_n + h, w_n + k_1)\ .$$

Here $t_n = nh$ and $w_n = w(t_n)$. The local error of this method is of order h^3 while Euler's method is of order h^2. Henceforth, we can use larger step size or save computation time with higher order numerical scheme. As one might expect, in terms of computational efficiency, we found that the larger the hybrid simulation time step, the better it is with a higher order method.

Finally, we remark that the proposed solution approach is somewhat similar to the Time-stepped Hybrid Simulation (TSHS) [11] and shares the idea of packet smoothing, however, the concept of hybrid system of CT-FSM composition is fundamentally different from TSHS.

IV. Hybrid Model & Simulation Validation

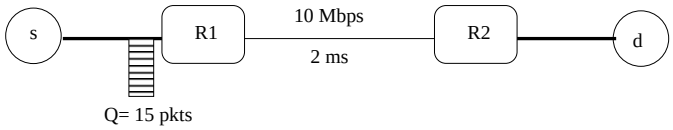

Fig. 1. A Simple Topology with 1 TCP New-Reno Source.

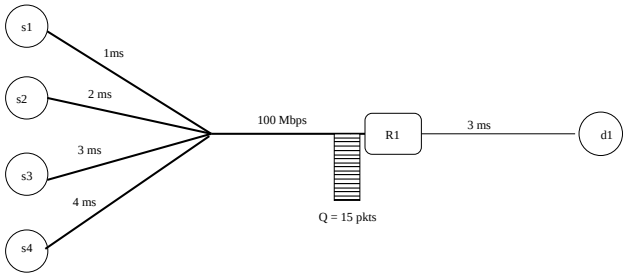

Fig. 2. A Dumbbell topology with 4 TCP New-Reno Sources.

A. Simulation Set Up & Environment

Our simulation studies are comprised of multiple experiments using the packet-based **ns-2** simulator and the hybrid model-driven simulator that we have developed. Since we are interested in comparing the two simulators we had to use the same traffic characteristics for both simulators. Furthermore the specific choice of the traffic pattern is dictated by how well it models network traffic. Experimental studies have shown that individual sources of traffic can be approximated by ON/OFF processes, where the On and Off periods have sub-exponential distributions. Within the ON state, a source transmits at peak rate. We use this model for our experiments with both the **ns-2** and the hybrid simulations. Packets are sent at a fixed rate during ON state while no packets are sent during OFF state. Both on and off periods are taken from a Pareto distribution with constant size packets. These traffic sources can be used to generate aggregate network traffic that exhibits the long range dependency seen in the IP network traffic.

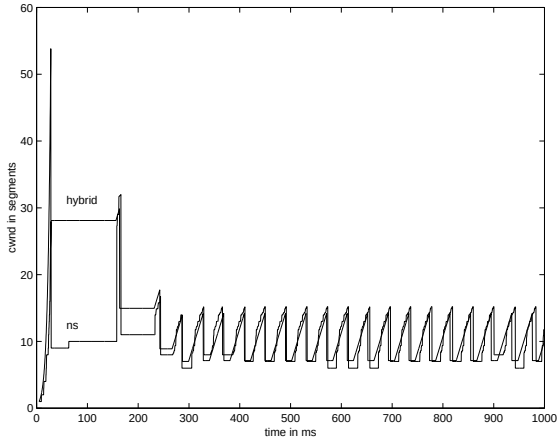

Fig. 3. Comparison of congestion window evolution from the hybrid model and **ns2** simulation. TCP New-Reno Sources.

B. Simulation Validation & Results

The first goal of our experimental set up was to baseline the performance of our hybrid model using well known packet level simulators such as **ns-2**. For this purpose we used two topologies. A simple topology as shown in Figure 1 with a single TCP New-Reno source and a dumbbell topology as shown in Figure 2 with 4 TCP New-Reno sources.

For the **ns-2** simulations we used an identical TCP source model with no background traffic. The simple topology is used to evaluate the behavior of the queue dynamics as well as TCP source behavior. It consists of a single edge and two network nodes. The link propagation delay was 2ms. In the dumbbell topology, we have 4 TCP sources that are aggregated at the queue for router 1 and each of these sources have link delays of 1, 2, 3, and 4ms each. All four TCP sources are destined for the same destination node **d**. The queue size is set to 15 packets with each packet consisting of 512 bytes. The run time for each of the simulations was 1000ms. The TCP flows were triggered to start at 5ms. The metrics used for comparing the hybrid simulator with the **ns-2** simulator are, the queue sizes and the congestion window sizes of the TCP source.

Figure 3 shows a comparison of the congestion window evolution of **ns-2** versus the hybrid model. Observe that the behavior of the two simulators is more closely matched in the congestion avoidance phase of the simulation while there is some mismatch in the slow start phase as well as the fr/fr phases. In the case of the hybrid model the exponential increase in the slow start phase reaches a value of 55 while in **ns-2** the maximum value of the congestion window in the slow start phase is only about 27. Also the increase in congestion window size in the hybrid model is smooth due to the sending rate model that is used. In the hybrid model we use a time averaged sending rate for the packets. The plausible explanation for the deviation in the behavior of **ns-2** is that **ns-2** employs a burst model sending rate in the initial slow start

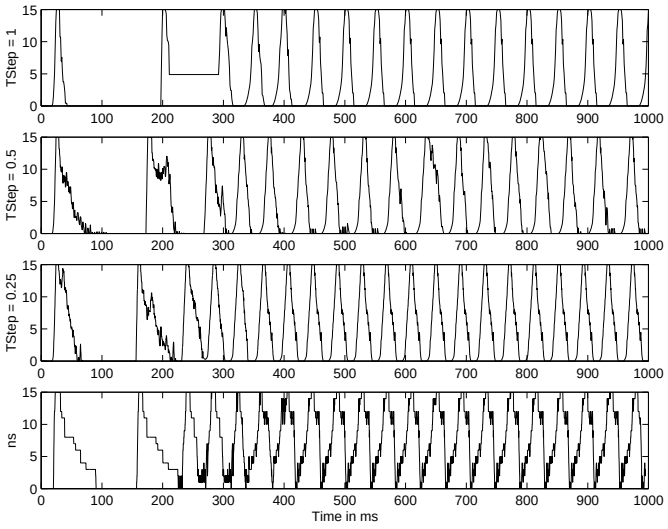

Fig. 4. Comparison of queue size changes using the hybrid model and **ns-2**. Queue size changes are evaluated using different time step values (0.25, 0.5, 1.0) in the hybrid simulator.

stages of the TCP flow. Also in the fr/fr mode we observe that the hybrid model is more graceful in evolving the congestion window while in **ns-2** there is an abrupt change in the window size.

The graceful congestion window evolution in the hybrid simulation is attributed to the wait time that is incorporated into the model after a drop occurs to account for the arrival of dupACKs. Furthermore, in the congestion avoidance phase we observe that the saw tooth behavior is more well behaved in the hybrid model as compared to ns. This smooth behavior is attributed to the lack of randomness in the arrival rates at the individual queues. In **ns-2** the randomness arises from the sending rate model that is used by the TCP sources.

Figure 4 shows the queue size changes using different time steps in the hybrid simulator as well as in **ns-2**. The queue sizes computed using the finer time steps in the hybrid model are much closer to the actual behavior. However using a finer time step increases the computational complexity. The queue size changes obtained by the hybrid simulator using a time step of 0.25 provides the closest match to the queue size change observed in **ns-2**.

In the dumbbell topology we simulate 4 TCP flows each with links of different delay values but having the same destination node. We do not assume any background traffic.

Figure 5 shows the evolution of the congestion window of the 4 different sources. Note that as expected in steady state all of the TCP sources are synchronized to a saw tooth pattern with a period of 100ms. The **ns-2** simulation for a similar topology is shown in Figure 6.

Note that in the steady state a good match is obtained for congestion window evolution in the hybrid as well as the **ns-2** models. The average values of the steady state congestion window sizes obtained using the hybrid model and **ns-2** are in Table I.

Figure 7 compares the queue size changes between the hybrid and the **ns-2** simulations. As the simulation time

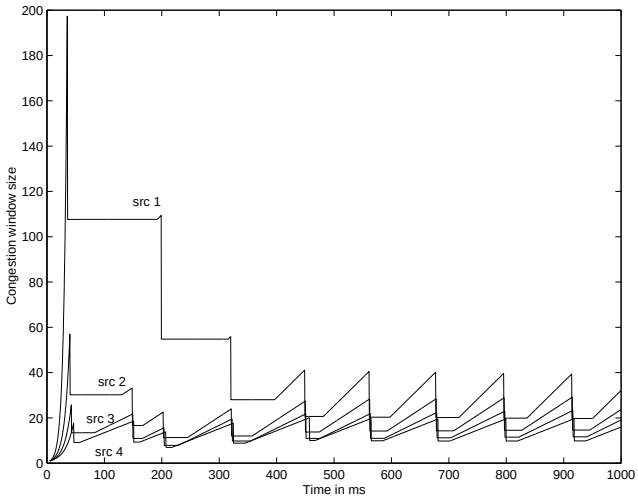

Fig. 5. Congestion window sizes evolution over time: 4 sources in the dumbbell topology using the hybrid simulation.

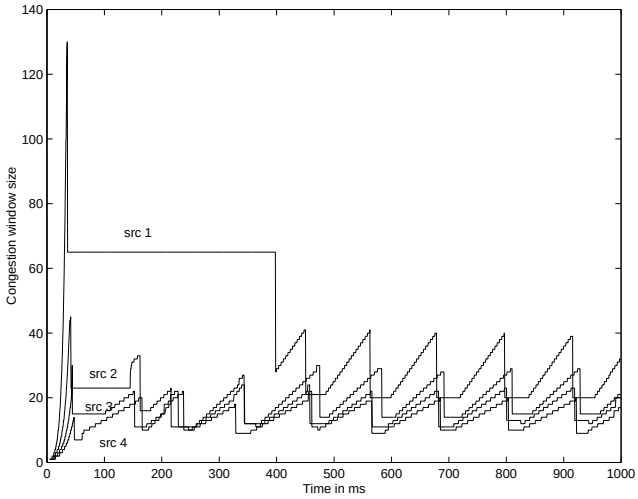

Fig. 6. Congestion window sizes evolution over time: 4 sources in the dumbbell topology using **ns-2**.

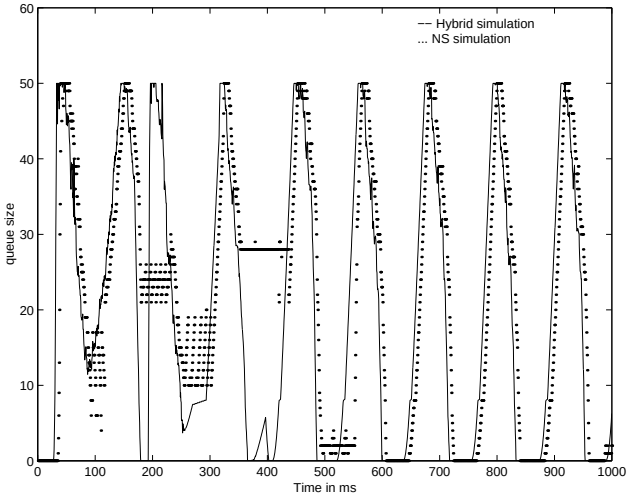

Fig. 7. Comparison of the queue size changes: the hybrid model *vs.* **ns-2**.

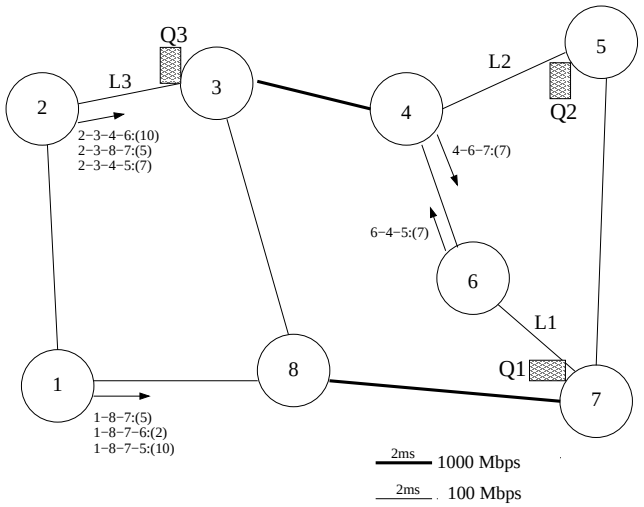

Fig. 8. US Topology loaded with 53 sources: A dashed-numbers (*e.g.* 1-8-7-6) denotes the path of a flow. The parenthesized number represents how many flows are in that path.

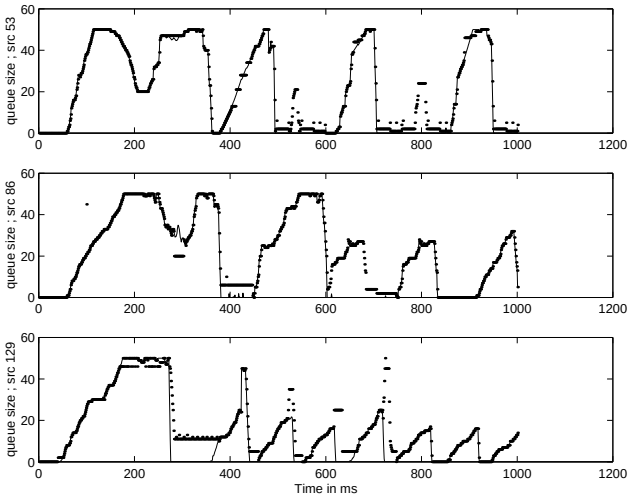

Fig. 9. Comparison of queue size changes using the hybrid model and **ns-2** for US topology. Queue size changes are evaluated using different number of sources. Legend: ··· **ns-2** and – hybrid

progresses and reaches a steady state we obtain a close match between the queue behavior of the **ns-2** and the hybrid models.

	$source_1$	$source_2$	$source_3$	$source_4$
hybrid	26.92	19.92	15.89	13.84
ns-2	26.92	19.82	16.02	13.94

TABLE I

DUMBBELL TOPOLOGY, STEADY STATE ($t \geq 500$MS) AVERAGE CONGESTION WINDOW SIZE.

V. SIMULATION OF LARGE NETWORK TOPOLOGIES

In order to study the effectiveness of the hybrid simulator we evaluated its performance using a larger topology. We compare the behavior of queue sizes under different TCP source configurations for both the hybrid simulator and **ns-2**.

A. Network Topology

Figure 8, called to be *US Topology*, shows a network of 8 nodes connected by duplex-links of 100 Mbps bandwidth and 2ms link-delay. Links 2-3 and 8-7 have higher, 1000 Mbps, bandwidth with 2ms link-delay also. For all the nodes, maximum queue size is set to 50 packets. Queue sizes are traced for the link 6-7 (L1), the link 4-5 (L2), and the link 2-3 (L3). We have used 53, 86, and 129 TCP sources where destination nodes are spread over the network.

B. Results

Figure 9 shows the change in queue (Q2) size under three different source configurations (53, 86 and 129 sources each) for both the **ns-2** as well as the hybrid simulator. In our simulation scenarios, of all the queues in the network this particular queue was the one that had the most discrepancy with the **ns-2** simulation. However note that even this worst case deviation from **ns-2** is not very significant as can be seen from Table II.

	Q_1	Q_2	Q_3
53 connections	(23.7:24.8)	(21.0:22.3)	(8.7:10.5)
86 connections	(24.4:23.8)	(20.5:22.5)	(11.8:13.0)
129 connections	(28.5:28.5)	(13.9:16.4)	(21.0:20.6)

TABLE II

US TOPOLOGY AVERAGE QUEUE LENGTH FOR Q1, Q2, Q3 WRITTEN FOR (HYBRID:NS-2)

VI. DISCUSSION & CONCLUSION

In this section we discuss the sending rate model (5) and the hybrid model implementation of TCP and complexity issues. We conclude the section with a brief summary of the paper.

A. Transport Layer & Application Layer

In Sec. III-C we pointed out that the sending rate model (5) might be inappropriate due to the bursty nature of the TCP protocol and the dependency on packet availability from applications. For example, in **ns-2** simulation the queue overflows much earlier than the hybrid simulation (see Figure 3). It is because the application sent all the packets in a much shorter period of time than the sending rate modeling equation (5).

The model in equation (5) sends out packets (flow) over a time interval of length RTT. More specifically, if congestion window size is 30 and RTT is 6ms then 5 packets are sent per 1ms in our model. **ns-2** is more likely to send all the 30 packets almost instantaneously. Hence if the bandwidth of the outbound link is not capable of processing all the incoming traffic, the queue overflows. However, with the model in equation 5,

the queue has enough time to process the flows that are arriving spread out over time.

The model in equation (5) is quite appropriate at the asymptotic stage, when the packets are more evenly spread out. One can observe that in Figure 3, the plots from the hybrid simulation and **ns-2** match well for $t \geq q$ 300ms.

B. Implementation Issues

Our current hybrid system simulator employs a fixed time-step approach. More specifically, we use the notion of a global clock and advance time by a fixed time-step till the end of simulation time. After each time-step, all the sources and queues are recomputed and checked against the transition enablers. This approach allows us to accurately incorporate propagation delays without further modeling effort. On the other hand, because the checks are carried out regardless of the states and the events at any time interval, its performance is not optimal. In future work, we plan to implement more intelligent time marching techniques — the resulting simulator would provide a better basis for studying the performance characteristics of hybrid based simulations.

C. Conclusions

In this paper, we have demonstrated the strength of the hybrid system modeling and simulation method for TCP network. New models, enhancements, and revisions were proposed and studied. We have used the time-stepped solution method to solve the hybrid system model of a TCP network. The extensive experiments showed good agreements with the results from Network Simulator **ns-2**. However, we have not yet studied the stability and the convergence issues of the numerical methods we have employed. The robust simulation tool which we have developed can be used to analyze complex network behavior so that service providers are better equipped to support the required quality of service (QoS) needs of their applications.

APPENDIX

A. Equation for Congestion Window Size

Consider slow-start mode in TCP source dynamics. In slow-start mode, the congestion window size increases exponentially in time. Suppose that the window size is multiplied by $m_{ss} \in R^+$ for every RTT, where R^+ is the set of real positive number. Usually $m_{ss} = 2$ since most of the TCP protocols double the congestion window size for every RTT. Let $RTT = n\Delta t$ for $\Delta t \in R^+$ and $n = 0, 1, \ldots$.

We assume that congestion window w uniformly increases over RTT time period $[t, t + RTT]$, that is, $w(t + \Delta t) = \sqrt[n]{m_{ss}}w(t)$. Then we have,

$$\frac{w(t+\Delta t) - w(t)}{\Delta t} = w(t)\frac{(m_{ss}^{\Delta t/RTT} - 1)}{\Delta t} \quad . \qquad (23)$$

Take $\lim_{\Delta t \to 0}$ on both sides of Eqn. (23) and applying L'Hospital's rule. We obtain,

$$w'(t) = \ln(m_{ss})\frac{w(t)}{RTT} \quad .$$

The governing equation for congestion avoidance mode, Eqn. (9) can be obtained in a similar manner.

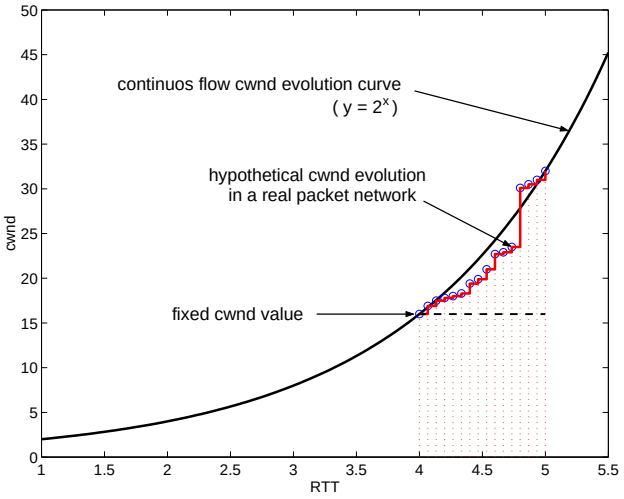

Fig. 10. Schematic description congestion window size changes for the TCP controlled flow in slow-start mode. With infinitesimal packets the congestion window size supposedly follows the solid curve. The step-wise increase may represent what happens in a real network. The dashed line indicates a fixed value of window size during RTT time period.

B. Correction Coefficient β

Bohacek *et. al.* used a correction factor $\beta = 1.45$ for the sending rate formula (6) to best match their simulation results to the ones from **ns-2**. Without the specifics of the numerical method used in Bohacek *et. al.*'s work, we conjecture that the presence of β is necessary if time points for event checking and triggering is RTT apart.

Let $m_{ss} = 2$ for the following discussion. Consider slow-start mode which is continued from $t = t_0$. Then congestion window w becomes 2^n at $t = t_0 + nRTT$. If a fixed w is used throughout the time interval $[nRTT, (n+1)RTT]$, the total number of packets sent during this time period is exactly w provided that Eqn. (6) is used for the packet sending process.

However, in reality, the congestion window size increases during this time interval due to arriving ACKs, therefore the source is able to send more packets. It can be explained schematically in Figure 10. The dashed line represents fixed window size ($w = 2^n$) over $[nRTT, (n+1)RTT]$, the step-wise variation may represent what happens in reality, and the solid curve is a plot of $y = 2^{x/RTT}$.

With the sending rate model (6), *i.e.* $r = w/RTT$, the area between the dashed line and the time axis represents the size of flow sent in $[nRTT, (n+1)RTT]$, where $n = 4$ in the figure. It is always smaller than the area below the step-wise function provided the slow-start mode persists. Suppose that the congestion window size variation can be closely approximated by $y = 2^{x/RTT}$ curve, *i.e.* with infinitesimal Δt, the area under the $y = 2^{x/RTT}$ curve for $[nRTT, (n+1)RTT]$ is

$$\int_{nRTT}^{(n+1)RTT} 2^{x/RTT} dx = 2^n \frac{RTT}{\ln(2)} \quad .$$

Note that $(\ln(2))^{-1} \approx 1.4424$ which is close to the value of $\beta = 1.45$ reported in [14].

REFERENCES

[1] D.D. Clark and D. L. Tennenhouse, "Architectural Considerations for a New Generation of Protocols", ACM SIGCOMM '90 (Philadelphia, PA, September 1990)

[2] The VINT Project on NS2, "The Network Simulator", ns2: http://www.isi.edu/nsnam/ns

[3] SSFNet, "The Scalable Simulation Framework", http://www.ssfnet.org/homePage.html

[4] J-Sim, "A component-based simulation environment", http://www.j-sim.org

[5] INET, "Network simulation in the J-Sim environment", http://www.j-sim.org/drcl.inet/index.html

[6] V. Misra, W-B. Gong, and D. Towsley, "Stochastic Differential Equation Modeling and Analysis of TCP Windowsize Behavior", Performance '99 (Istanbul, Turkey, October 1999)

[7] V. Misra, W-B. Gong, and D. Towsley, "A Fluid-based Analysis of a Network of AQM Routers Supporting TCP Flows with an Application to RED", Proceedings of ACM SIGCOMM'00, (Stockholm, Sweden, September 2000)

[8] E. Altman, K. Avrachenkov, and C. Barakat, "A Stochastic Model of TCP/IP with Stationary Random Losses", ACM SIGCOMM Computer Communication Review, 30: 231-242, 2000

[9] T. Bonald, "Comparison of TCP Reno and TCP Vegas via fluid approximation", Technical Report RR-3563, INRIA, 1998

[10] V. Jacobson, "Congestion Avoidance and Control", Computer Communication Review, vol. 18, 1988

[11] Y. Guo, W-B. Gong, and D. Towsley, "Time-stepped Hybrid Simulation (TSHS) for Large Scale Networks", Proceedings of IEEE Infocom 2000, 441-450, 2000

[12] Y. Gu, Y. Li, and D. Towsley, "On Integrating Fluid Models with Packet Simulation", Proceedings of IEEE Infocom 2004, 2004

[13] A. Yan and W-B. Gong, "Time-Driven Fluid Simulation for High-Speed Networks", IEEE Trans. on Information Theory, vol. 45, no. 5, July 1999

[14] S. Bohacek, J.P. Hespanha, J. Lee, and K. Obraczka, "A Hybrid Systems Modeling Framework for Fast and Accurate Simulation of Data Communication Networks", ACM SIGMETRICS'03, June 2003

[15] A.J. Van Der Schaft and J.M. Schumacher, An Introduction to Hybrid Dynamical Systems (Lecture Notes in Control and Information Sciences, 251) Springer-Verlag Telos, 1999

[16] S.D. Conte and C. Boor, "Elementary Numerical Analysis (An Algorithmic Approach)" McGraw-Hill, 1981

[17] The MathWorks Inc., "Simulink", http://www.mathworks.com/products/simulink

[18] The Modelica Association, "Modelica: ", http://www.modelica.org

[19] Dept. of EECS, University of California at Berkeley, "The Ptolemy Project", http://ptolemy.eecs.berkeley.edu

[20] E. Lee "Overview of the Ptolemy Project", Technical Memorandum No. UCB/ERL M03/25, University of California, Berkeley, CA, 94720