Modeling Communication Networks With Hybrid Systems

Junsoo Lee, Stephan Bohacek, *Member, IEEE*, João P. Hespanha, *Senior Member, IEEE*, and Katia Obraczka, *Member, IEEE*

Abstract—This paper introduces a general hybrid systems framework to model the flow of traffic in communication networks. The proposed models use averaging to continuously approximate discrete variables such as congestion window and queue size. Because averaging occurs over short time intervals, discrete events such as the occurrence of a drop and the consequent reaction by congestion control can still be captured. This modeling framework, thus, fills a gap between purely packet-level and fluid-based models, faithfully capturing the dynamics of transient phenomena and yet providing significant flexibility in modeling various congestion control mechanisms, different queueing policies, multicast transmission, etc. The modeling framework is validated by comparing simulations of the hybrid models against packet-level simulations. It is shown that the probability density functions produced by the `ns-2` network simulator match closely those obtained with hybrid models. Moreover, a complexity analysis supports the observation that in networks with large per-flow bandwidths, simulations using hybrid models require significantly less computational resources than `ns-2` simulations. Tools developed to automate the generation and simulation of hybrid systems models are also presented. Their use is showcased in a study, which simulates TCP flows with different roundtrip times over the Abilene backbone.

Index Terms—Congestion control, data communication networks, hybrid systems, simulation, TCP, UDP.

I. INTRODUCTION

DATA communication networks are highly complex systems, thus modeling and analyzing their behavior is quite challenging. The problem aggravates as networks become larger and more complex. Packet-level models are the most accurate network models and work by keeping track of individual packets as they travel across the network. Packet-level models, which are used in network simulators such as ns-2 [1], have two main drawbacks: the large computational requirements (both in processing and storage) for large-scale simulations and the difficulty in understanding how network parameters affect the

J. Lee is with Computer Science Department, Sookmyung Women's University, Seoul, Korea 140-742 (e-mail: junsool@sm.ac.kr).

S. Bohacek is with the Electrical and Computer Engineering Department, University of Delaware, Newark, DE 19716 USA (e-mail: bohacek@eecis.udel.edu).

J. P. Hespanha is with the Electrical and Computer Engineering Department, University of California, Santa Barbara, CA 93106 USA (e-mail: hespanha@ece.ucsb.edu).

K. Obraczka is with the Computer Engineering Department, University of California, Santa Cruz, CA 95064 USA (e-mail: katia@cse.ucsc.edu).

overall system performance. Aggregate fluid-like models overcome these problems by simply keeping track of the average quantities that are relevant for network design and provisioning (such as queue sizes, transmission rates, drop rates, etc). Examples of fluid models that have been proposed to study computer networks include [2] and [3]. The main limitation of these aggregate models is that they mostly capture steady state behavior because the averaging is typically done over large time scales. Thus, detailed transient behavior during congestion control cannot be captured. Consequently, these models are unsuitable for a number of scenarios, including capturing the dynamics of short-lived flows.

Our approach to modeling computer networks and its protocols is to use hybrid systems [4] which combine continuous-time dynamics with event-based logic. These models permit complexity reduction through continuous approximation of variables like queue and congestion window size, without compromising the expressiveness of logic-based models. The "hybridness" of the model comes from the fact that, by using averaging, many variables that are essentially discrete (such as queue and window sizes) are allowed to take continuous values. However, because averaging occurs over short time intervals, one still models discrete events such as the occurrence of a drop and the consequent reaction by congestion control.

In this paper, we propose a general framework for building hybrid models that describe network behavior. Our hybrid systems framework fills the gap between packet-level and aggregate models by averaging discrete variables over a short time scale on the order of a roundtrip time (RTT). This means that the model is able to capture the dynamics of transient phenomena fairly accurately, as long as their time constants are larger than a couple of RTTs. This time scale is appropriate for the analysis and design of network protocols including congestion control mechanisms.

We use TCP as a case-study to showcase the accuracy and efficiency of the models that can be built using the proposed framework. We are able to model fairly accurately TCP's distinct congestion control modes (e.g., slow-start, congestion avoidance, fast recovery, etc.) as these last for periods no shorter than one RTT. One should keep in mind that the timing at which events occur in the model (e.g., drops or transitions between TCP modes) is only accurate up to roughly one RTT. However, since the variations on the RTT typically occur at a slower time scale, the hybrid models can still capture quite accurately the dynamics of RTT evolution. In fact, that is one of the strengths of the models proposed here, i.e., the fact that they do not assume constant RTT.

We validate our modeling methodology by comparing simulation results obtained from hybrid models and packet-level

simulations. We ran extensive simulations using different network topologies subject to different traffic conditions (including background traffic). Our results show that hybrid models are able to reproduce packet-level simulations quite accurately. We also compare the run time of the two approaches and show that hybrid models incur considerably less computational load. We anticipate that speedups yielded by hybrid models will be instrumental in studying large-scale, more complex networks.

Finally, we describe the *network description scripting language* (NDSL) and the *NDSL translator*, which were developed to automate the generation and simulation of hybrid systems models. NDSL is a scripting language that allows the user to specify network topologies and traffic. The NDSL translator automatically generates the corresponding hybrid models in the `modelica` modeling language [5]. We showcase these tools in a simulation study on the effect of the RTT on the throughput of TCP flows over the Abilene backbone [6].

II. RELATED WORK

Several approaches to the modeling and simulation of networks have been widely used by the networking community to design and evaluate network protocols. On one side of the spectrum, there are packet-level simulation models: `ns-2` [1], QualNet [7], SSFNET [8], and Opnet [9] are event simulators where an event is the arrival or departure of a packet. Whenever a packet arrives at the link or node, events are generated and stored in the event list and handled in the appropriate order. These models are highly accurate, but are not scalable to large networks. On the other extreme, static models provide approximations using first principles: [3] and [10] provide simple formulas that model how TCP behaves in steady state. These models ignore much of the dynamics of the network. For example, the RTT and loss probability are assumed constant and the interaction between flows is not considered.

Dynamic model fall between static models and detailed packet-level simulators. By allowing some parameters to vary, these models attempt to obtain more accuracy than static approaches, and yet alleviate some of the computational overhead of packet-level simulations. This modeling approach was followed by [11], where TCP's sending rate is taken as an ensemble average. When averaging across multiple flows, the sending rates do not exhibit the linear increase and divide in half. However, the ensemble average still varies dynamically with queue size and drop probability. [2] proposes a stochastic differential equation (SDE) model of TCP, in which the sending rate increases linearly until a drop event occurs and then it is divided in half.

While the dynamic models above proved very useful for developing a theoretical understanding of networks, their purpose was not to simulate networks. In an effort to simulate networks efficiently, [12] and [13] proposed a fluid-like approach in which bit rates are assumed piecewise constant. This type of network simulator only needs to keep track of rate changes that occur due to queueing, multiplexing, and services. As a result, the computational effort may be reduced with respect to a packet-level simulators. However, the piecewise constant assumption can lead to an explosion of events known as the ripple
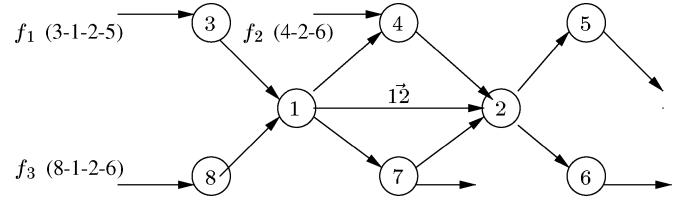

Fig. 1. Example network where $q_{\vec{12}} = q_{\vec{12}}^{f_1} + q_{\vec{12}}^{f_3}$.

effect [14], which can significantly increase the computational load.

Systems that exhibit continuously varying variables whose values are affected by events generated by discrete-logic are known as *hybrid systems* and have been widely used in many fields to model physical systems. The reader is referred to [4] for a general overview of hybrid systems. An early hybrid modeling approach to computer systems appeared in [15], where the author proposes to combine discrete-event models with continuous analytic models. The former are used to capture "rare events," whereas the latter avoid the need to carry out the detailed simulation of very frequent events. This general framework was used in [15] to simulate a central server systems consisting of a CPU and several IO devices serving multiple jobs.

Traffic sampling [16] consists of taking a sample of network traffic, feeding it into a suitably scaled version of the network, and then using the results so obtained to extrapolate the behavior of the original network. This has been proposed as a methodology to efficiently simulate large-scale networks by combining simulation and analytical techniques. However, it loses scalability when packet drops are bursty and correlated, or when packet drops are not accurately modeled by a Poisson process.

The remainder of the paper is organized as follows. Section III presents our hybrid systems modeling framework. In Section IV, we validate our hybrid models by comparing them to packet-level simulations. Section V shows results comparing the computational complexity of hybrid- and packet-level models, and Section VI shows development tools and case study using these tools. Finally, we present our concluding remarks and directions for future work in Section VII.

III. HYBRID MODELING FRAMEWORK

Consider a communication network consisting of a set $\mathcal{N}$ of *nodes* connected by a set $\mathcal{L}$ of *links*. We assume that all links are unidirectional and denote by $\ell := \vec{ij}$ the link from node $i \in \mathcal{N}$ to node $j \in \mathcal{N}$ (cf. Fig. 1). Every link $\ell \in \mathcal{L}$ is characterized by a finite *bandwidth* B^ℓ and a *propagation delay* T^ℓ.

We assume that the network is being loaded by a set $\mathcal{F}$ of *end-to-end flows*. Given a flow $f \in \mathcal{F}$ from node $i \in \mathcal{N}$ to node $j \in \mathcal{N}$, we denote by r_f the *flow's sending rate*, i.e., the rate at which packets are generated and enter node i where the flow is initiated. Given a link $\ell \in \mathcal{L}$ in the path of the f-flow, we denote by r_f^ℓ the rate at which packets from the f-flow are sent through the ℓ-link. We call r_f^ℓ the *ℓ-link/f-flow transmission rate*. At each link, the sum of the link/flow transmission rates must not exceed the bandwidth, i.e., B^ℓ

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

In general, the flow sending rates r_f, $f \in \mathcal{F}$ are determined by congestion control mechanisms and the link/flow transmission rates r_f^ℓ are determined by packet conservation laws to be derived shortly. To account for the fact that not all packets may have the same length, we measure all rates in bytes per second.

Associated with each link $\ell \in \mathcal{L}$, there is a *queue* that holds packets before transmission through the link. We denote by q_f^ℓ the number of bytes in this queue that belong to the f flow. The total number of bytes in the queue is given by

$$q^\ell := \sum_{f \in \mathcal{F}} q_f^\ell, \qquad \forall \ell \in \mathcal{L}. \tag{2}$$

The queue can hold, at most, a finite number of bytes that we denote by q_{max}^ℓ. When q^ℓ reaches q_{max}^ℓ, drops will occur.

For each flow $f \in \mathcal{F}$, we denote by RTT_f the *flow's RTT*, which elapses between a packet is sent and its acknowledgment is received. The value of RTT_f can be determined by adding the propagation delays T^ℓ and queueing times q^ℓ/B^ℓ of all links involved in one roundtrip. In particular

$$\mathrm{RTT}_f = \sum_{\ell \in \mathcal{L}[f]} \left(T^\ell + \frac{q^\ell}{B^\ell} \right)$$

where $\mathcal{L}[f]$ denotes the set of links involved in one roundtrip for flow f.

A. Flow Conservation Laws

Consider a link $\ell \in \mathcal{L}$ in the path of flow $f \in \mathcal{F}$. We denote by s_f^ℓ the rate at which f-flow packets arrive (or originate) at the node where ℓ starts. We call s_f^ℓ the *ℓ-link/f-flow arrival rate*. The link/flow arrival rates s_f^ℓ are related to the flow sending rates r_f and the link/flow transmission rates r_f^ℓ by the following simple *flow-conservation law*: for every $f \in \mathcal{F}$ and $\ell \in \mathcal{L}$

$$s_f^\ell := \begin{cases} r_f, & f \text{ starts at the node where } \ell \text{ starts} \\ r_f^{\ell'}, & \text{otherwise} \end{cases} \tag{3}$$

where ℓ' denotes the previous link in the path of the f-flow. For simplicity, we are assuming single-path routing and unicast transmission. It would be straightforward to derive conservation laws for multipath routing and multicast transmission.

The flow-conservation law (3) implicitly assumes that packets are not dropped "on the fly." For consistency, we will regard packet drops that occur in the transmission medium (e.g., needed to model wireless links) as taking place upon arrival at the destination node. From a traffic modeling perspective, this makes no difference but somewhat simplifies the notation.

B. Queue Dynamics

In this section, we make two basic assumptions regarding flow uniformity that are used to derive our models for the queue dynamics.

Assumption 1 (Arrival Uniformity): The packets of the all flows arrive at each node in their paths roughly uniformly distributed over time. Consequently, the packets of each flow are roughly uniformly distributed along each queue.

Because of packet quantization, bursting, synchronization, etc., this assumption are never quite true over a very small interval of time. However, they are generally accurate over time

intervals of a few RTTs. In fact, we shall see shortly that they are sufficiently accurate to lead to models that match closely packet-level simulations.

1) Queue-Evolution and Drop Rates: Consider a link $\ell \in \mathcal{L}$ that is in the path of the flow $f \in \mathcal{F}$. The queue dynamics associated with this pair link/flow are given by

$$\dot{q}_f^\ell = s_f^\ell - d_f^\ell - r_f^\ell$$

where d_f^ℓ denotes the *f-flow drop rate*. In this equation, s_f^ℓ should be regarded as an input whose value is determined by upstream nodes. To determine the values of d_f^ℓ and r_f^ℓ, we consider three cases separately.

1) *Empty queue* (i.e., $q^\ell = 0$). In this situation, there are no drops and the outgoing rates r_f^ℓ are equal to the arrival rates s_f^ℓ, as long as the bandwidth constrain (1) is not violated. However, when $\sum_{f \in \mathcal{F}} s_f^\ell > B^\ell$, we cannot have $r_f^\ell = s_f^\ell$, and the available link bandwidth B^ℓ must be somehow distributed amount the flows so that $\sum_{f \in \mathcal{F}} r_f^\ell = B^\ell$. To determine how to distribute B^ℓ, we note that a total of $\sum_{f \in \mathcal{F}} s_f^\ell$ bytes arrive at the queue in a single unit of time. Assuming arrival uniformity (Assumption 1) all incoming packets are equally likely to be transmitted so the probability that a packet of flow f is, indeed, transmitted is given by

$$\frac{s_f^\ell}{\sum_{\bar{f} \in \mathcal{F}} s_{\bar{f}}^\ell}. \tag{4}$$

Since a total of B^ℓ bytes will be transmitted, the faction of these that correspond to flow f is given by

$$\frac{s_f^\ell B^\ell}{\sum_{\bar{f} \in \mathcal{F}} s_{\bar{f}}^\ell}.$$

The above discussion can be summarized as follows: For every $f \in \mathcal{F}$

$$d_f^\ell = 0, \quad r_f^\ell = \begin{cases} s_f^\ell, & \sum_{\bar{f} \in \mathcal{F}} s_{\bar{f}}^\ell \le B^\ell \\ \frac{s_f^\ell}{\sum_{\bar{f} \in \mathcal{F}} s_{\bar{f}}^\ell} B^\ell, & \sum_{\bar{f} \in \mathcal{F}} s_{\bar{f}}^\ell > B^\ell. \end{cases}$$

2) *Queue neither empty nor full* (i.e., $0 < q^\ell < q_{max}^\ell$ or $q^\ell = q_{max}^\ell$ but $\sum_{\bar{f} \in \mathcal{F}} s_{\bar{f}}^\ell \le B^\ell$). In this situation, there are no drops, and the available link bandwidth B^ℓ must also be distributed amount the flows so that $\sum_{f \in \mathcal{F}} r_f^\ell = B^\ell$. Assuming that the packets of each flow are uniformly distributed along each queue (Assumption 1), the probability that a packet of flow f is at the head of the queue is given by

$$\frac{q_f^\ell}{\sum_{\bar{f} \in \mathcal{F}} q_{\bar{f}}^\ell}. \tag{5}$$

Since a total of B^ℓ bytes will be transmitted per unit of time, the faction of these that correspond to flow f is given by

$$\frac{q_f^\ell B^\ell}{\sum_{\bar{f} \in \mathcal{F}} q_{\bar{f}}^\ell}.$$

We, thus, conclude that, in this situation, for every $f \in \mathcal{F}$

$$d_f^\ell = 0, \quad r_f^\ell = \frac{q_f^\ell B^\ell}{\sum_{\bar{f} \in \mathcal{F}} q_{\bar{f}}^\ell}.$$

3) *Queue full and still filling* (i.e., $q^\ell = q_{\max}^\ell$ and $\sum_{\bar{f} \in \mathcal{F}} s_{\bar{f}}^\ell > B^\ell$). In this situation, the total drop rate d^ℓ must equal the difference between the total arrival rate and the link bandwidth, i.e., $d^\ell := \sum_{\bar{f} \in \mathcal{F}} s_{\bar{f}}^\ell - B^\ell > 0$. Once again, we must determine how this total drop rate d^ℓ should be distributed among all flows. Assuming arrival uniformity (Assumption 1) all incoming packets are equally likely to be dropped so the probability that a packet of flow f is indeed dropped is given by

$$\frac{s_f^\ell}{\sum_{\bar{f} \in \mathcal{F}} s_{\bar{f}}^\ell}. \tag{6}$$

Since a total of d^ℓ bytes will be dropped, the faction of these that correspond to flow f is given by

$$\frac{s_f^\ell d^\ell}{\sum_{\bar{f} \in \mathcal{F}} s_{\bar{f}}^\ell} = \frac{s_f^\ell \left(\sum_{\bar{f} \in \mathcal{F}} s_{\bar{f}}^\ell - B^\ell \right)}{\sum_{\bar{f} \in \mathcal{F}} s_{\bar{f}}^\ell} = s_f^\ell - \frac{s_f^\ell B^\ell}{\sum_{\bar{f} \in \mathcal{F}} s_{\bar{f}}^\ell}.$$

The rate at which packets are transmitted is the same as when the queue is neither empty not full, which was considered above. This leads to the following model: for every $f \in \mathcal{F}$

$$d_f^\ell = s_f^\ell - \frac{s_f^\ell B^\ell}{\sum_{\bar{f} \in \mathcal{F}} s_{\bar{f}}^\ell}, \quad r_f^\ell = \frac{q_f^\ell B^\ell}{\sum_{\bar{f} \in \mathcal{F}} q_{\bar{f}}^\ell}. \tag{7}$$

To complete the queue dynamics model, it remains to determine when and which flows suffer drops. To this effect, suppose that at time t_1, q^ℓ reached $q_{\max}^\ell$ with $s^\ell := \sum_{f \in \mathcal{F}} s_f^\ell > B^\ell$. Clearly, a drop will occur at time t_1 but, multiple drops may occur. In general, if a drop occurred at time t_k, a new drop is expected at a time $t_{k+1} > t_k$, for which the total drop rate d^ℓ integrates from t_k to t_{k+1} to exactly the packet-size L, i.e., for which

$$z^\ell := \int_{t_k}^{t_{k+1}} \sum_{f \in \mathcal{F}} d_f^\ell = \int_{t_k}^{t_{k+1}} \left(\sum_{f \in \mathcal{F}} s_f^\ell - B^\ell \right) = L. \tag{8}$$

This equation determines t_{k+1}, $k \geq 1$ for all drops after t_1. We call (8) the *drop-count model*.

The question as to which flows suffer drops must be consistent with the drop probability specified by (6), which was a consequence of the arrival uniformity Assumption 1. In particular, the selection of the flow $\mathbf{f}^*$ where a drop occurs is made by drawing the flow randomly from the set $\mathcal{F}$, according to the distribution

$$p_{\mathbf{f}^*}(f) := \mathrm{P}(\mathbf{f}^* = f) = \frac{d_f^\ell}{\sum_{\bar{f} \in \mathcal{F}} d_{\bar{f}}^\ell} = \frac{s_f^\ell}{\sum_{\bar{f} \in \mathcal{F}} s_{\bar{f}}^\ell}, \quad \forall f \in \mathcal{F}. \tag{9}$$
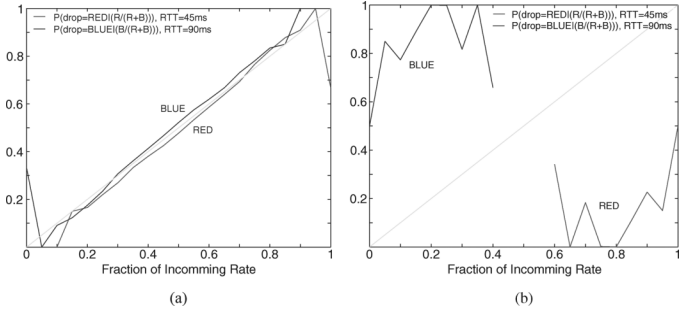

Fig. 2. Drop probability versus fraction of arrival rate. (a) 10% background traffic; (b) packet synchronization.

We assume that the flows $\mathbf{f}^*(t_k)$, $\mathbf{f}^*(t_{k+1})$ that suffer drops at two distinct time instants t_k, t_{k+1} are (conditionally) independent random variables (given that the drops did occur at times t_k and t_{k+1}). We call (9) the *drop-selection model*.

The uniformity Assumption 1 was used in the construction of our queue model to justify the formulas (4), (5) for the packet transmission probabilities and the formula (6) for the packet drop probability. To validate this assumption, we matched these formulas with the results of several ns-2 [1] simulations. Fig. 2 shows the result of one such validation procedure for the formula (6). Fig. 2(a) refers to a simulation in which 2 TCP flows (RED and BLUE) compete for bandwidth on a bottleneck queue (with 10% ON–OFF UDP traffic). The x axis shows the fraction of arrival rate for each flow given by the formula (6) and the y axis shows the corresponding drop probability. A near perfect $45°$ line shows that (6) does provide a very good approximation to the packet drop probability. Fig. 2(b) shows a network with very strong drop synchronization for which Assumption 1 breaks down. We postpone the discussion of this plot to Section III-B3. Similar plots can be made to validate the formulas (4), (5) for the packet transmission probabilities, but we do not include them here for lack of space. However, in Section IV, we present a systematic validation of our overall hybrid model, which includes the queue mode above as a subcomponent.

2) Hybrid Model for Queue Dynamics: The queue model developed above can be compactly expressed by the hybrid automaton represented in Fig. 3. Each ellipse in this figure corresponds to a discrete state (or mode) and the continuous state of the hybrid system consists of the flow byte rate s_f^ℓ, $f \in \mathcal{F}$ and the variable z^ℓ used to track the number of drops in the *queue-full* mode. The differential equations for these variables in each mode are shown inside the corresponding ellipse. The arrows between ellipses represent transitions between modes. These transitions are labeled with their enabling conditions (which can include events generated by other transitions), any necessary reset of the continuous state that must take place when the transition occurs, and events generated by the transition. Events are denoted by $\mathfrak{E}[\cdot]$. We assume here that a jump always occurs when the transition condition is enabled. This model is consistent with most of the hybrid system frameworks proposed in the literature (cf., e.g., [4]). The inputs to this model are the rates $r_f^{\ell'}$, $f \in \mathcal{F}$ of the upstream queues $\ell' \in \mathcal{L}[\ell]$, which determine
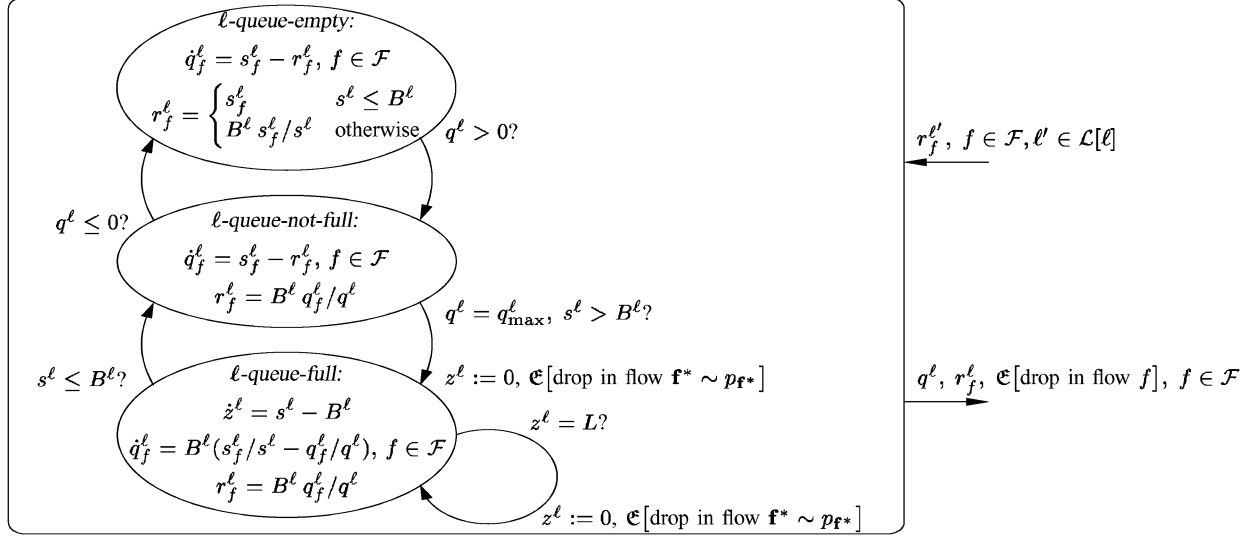
Fig. 3. Hybrid model for the queue at link ℓ. In this figure, q^ℓ is given by (2), the s_f^ℓ, $f \in \mathcal{F}$ are given by (3), and $s^\ell := \sum_{f \in \mathcal{F}} s_f^\ell, \forall \ell \in \mathcal{L}$.

the arrival rates s_f^ℓ, $f \in \mathcal{F}$, and the outputs are the transmission rates r_f^ℓ, $f \in \mathcal{F}$. For the purpose of congestion control, we should also regard the drop events and the queue size as outputs of the hybrid model. Note that the queue sizes will eventually determine packet RTTs.

3) Other Drop Models: For completeness, one should add that the drop-selection model described by (9) is not universal. For example, in dumbbell topologies without background traffic, one can observe synchronization phenomena that sometimes lead to flows with smaller sending rates suffering more drops than flows with larger sending rates. The right plot in Fig. 2 shows an extreme example of this (2-TCP flows in a 5-Mbps dumbbell topology with no background traffic and drop-tail queueing). In this example, the BLUE flow suffers most of the drops, in spite of using a smaller fraction of the bandwidth. In [17], it was suggested that 10% of random delay would remove synchronization between TCP connections. However, this does not appear to be the case when the number of connections is small. To avoid synchronization we mostly used background traffic. In fact, the left plot in Fig. 2 shows results obtained with 10% background traffic, whereas the right plot shows results obtained without any background traffic.

Drop Rotation: The drop model in (9) is not very accurate when strong synchronization occurs. Constructing drop models that remain accurate under packet-drops synchronized is generally very challenging, except under special network conditions. The *drop rotation* model is valid in topologies with drop-tail queueing, when several TCP flows have the roughly the same RTT and there is a bottleneck link with bandwidth significantly smaller than that of the remaining links and there is no (or little) background traffic [18]–[20]. Under this model, when the queue gets full, each flow gets a drop in a round-robin fashion. The rationale for this is that, once the queue gets full, it will remain full until TCP reacts (approximately one RTT after the drop). In the meantime, all TCP flows are in the congestion avoidance mode and each will increase its window size by one. When this occurs each will attempt to send two packets back-to-back and,

under a drop-tail queueing policy, the second packet will almost certainly be dropped.

Although the drop rotation model is only valid for special networks, these networks are very useful to validate congestion control because they lead to essentially deterministic drops. This allows one to compare exactly traces obtained from packet-level models with traces obtained from hybrid models. We will use this feature of drop rotation to validate our hybrid models for TCP in Section IV.

Other drop models, such as *active queueing* and *drop-head*, can be found in [21].

C. TCP Model

So far, our discussion focused on the modeling of the transmission rates r_f^ℓ and the queue sizes q_f^ℓ across the network, taking as inputs the sending rates r_f of the end-to-end flows. We now construct a hybrid model for a single TCP flow $f \in \mathcal{F}$ that should be composed with the flow-conservation laws and queue dynamics presented in Sections III-A and B to describe the overall system. We start by describing the behavior of TCP in each of its main modes and later combine them into a unified hybrid model of TCP.

1) Slow-Start Mode: During *slow-start*, the *congestion window* w_f (cwnd) increases exponentially, being multiplied by 2 every RTT. This can be modeled by

$$\dot{w}_f = \frac{\log 2}{\mathrm{RTT}_f} w_f. \tag{10}$$

because, neglecting the variation of RTT_f during a single RTT, this would lead to

$$w_f(t + \mathrm{RTT}_f) = e^{\log 2 \int_t^{t+\mathrm{RTT}_f} \frac{1}{\mathrm{RTT}_f} d\tau} w_f(t) \approx 2 w_f(t).$$

Since w_f packets are sent each RTT, the instantaneous sending rate r_f should be given by

$$r_f = \frac{w_f}{\mathrm{RTT}_f}. \tag{11}$$

However, this formula needs to be corrected to

$$r_f = \frac{\beta w_f}{\text{RTT}_f} \qquad (12)$$

because slow-start packets are sent in bursts. A detailed justification for the introduction of the parameter $\beta := 1.59$ can be found in [21].

The *slow-start* mode lasts until a drop or a timeout are detected. Detection of a drop leads the system to the *fast-recovery* mode, whereas the detection of a timeout leads the system to the *timeout* mode.

The formulas (10) and (12) hold as long as the congestion window w_f is below the receiver's advertised window size w_f^{adv}. When w_f exceeds this value, the sending rate is limited by w_f^{adv} and (12) should be replaced by

$$r_f = \frac{\min\left\{\beta w_f, w_f^{\text{adv}}\right\}}{\text{RTT}_f}. \qquad (13)$$

When the congestion window reaches the advertised window size, the system transitions to the *congestion-avoidance* mode.

2) Congestion-Avoidance Mode: During the *congestion-avoidance* mode, the congestion window size increases "linearly," with an increase equal to the packet-size L for each RTT. This can be modeled by

$$\dot{w}_f = \frac{L}{\text{RTT}_f}$$

with the instantaneous sending rate r_f given by (11). When the receiver's advertised window size w_f^{adv} is finite, (11) should be replaced by

$$r_f = \frac{\min\left\{w_f, w_f^{\text{adv}}\right\}}{\text{RTT}_f}.$$

The *congestion-avoidance* mode lasts until a drop or timeout are detected. Detection of a drop leads the system to the *fast-recovery* mode, whereas detection of a timeout leads the system to the *timeout* mode.

3) Fast-Recovery Mode: The *fast-recovery* mode is entered when a drop is detected, which occurs some time after the drop actually occurs. When a single drop occurs, the sender leaves this mode at the time it learns that the packet dropped was successfully retransmitted (i.e., when its acknowledgment arrives). When multiple drops occur, the transition out of fast recovery depends on the particular version of TCP implemented. We provide next the model for TCP-Sack.

TCP-Sack: In TCP-Sack, when n_{drop} drops occur, the sender learns immediately that several drops occurred and will attempt to retransmit all these packets as soon as the congestion window allows it. As soon as *fast-recovery* is initiated, the first packet dropped is retransmitted and the congestion window is divided by two. After that, for each acknowledgment received, the congestion window is increased by one. However, until the first retransmission succeeds, the number of outstanding packets is not decreased when acknowledgments arrive.

Suppose that the drop was detected at time t_0 and let $w_f(t_0^-)$ denote the window size just before its division by 2. In practice,

during the first RTT after the retransmission (i.e., from t_0 to $t_0 + \text{RTT}_f$), the number of outstanding packets is $w_f(t_0^-)$; the number of duplicate acknowledgments received is equal to $w_f(t_0^-) - n_{\text{drop}}$ (we are including here the three duplicate acknowledgments that triggered the retransmission), and a single nonduplicate acknowledgment is received (corresponding to the retransmission). The total number of packets sent during this interval will be one (corresponding to the retransmission that took place immediately), plus the number of duplicate acknowledgments received, minus $w_f(t_0^-)/2$. We need to subtract $w_f(t_0^-)/2$ because the first $w_f(t_0^-)/2$ acknowledgments received will increase the congestion window up to the number of outstanding packets but will not lead to transmissions because the congestion window is still below the number of outstanding packets [22]. This leads to a total of $1 + w_f(t_0^-)/2 - n_{\text{drop}}$ packets sent, which can be modeled by an average sending rate of

$$r_f = \frac{1 + w_f(t_0^-)/2 - n_{\text{drop}}}{\text{RTT}_f} \qquad \text{on } [t_0, t_0 + \text{RTT}_f].$$

In case a single packet was dropped, fast recovery will finish at $t_0 + \text{RTT}_f$; otherwise, it will continue until all the retransmissions take place and are successful. However, from $t_0 + \text{RTT}_f$ on, each acknowledgment received will also decrease the number of outstanding packets so one will observe an exponential increase in the window size. In particular, from $t_0 + \text{RTT}_f$ to $t_0 + 2\text{RTT}_f$ the number of acknowledgments received is $1 + w_f(t_0^-)/2 - n_{\text{drop}}$ (which was the number of packets sent in the previous interval) and each will both increase the congestion window size and decrease the number of outstanding packets. This will lead to a total number of packets sent equal to $2(1 + w_f(t_0^-)/2 - n_{\text{drop}})$ and, therefore

$$r_f = \frac{2\left(1 + w_f(t_0^-)/2 - n_{\text{drop}}\right)}{\text{RTT}_f} \quad \text{on } [t_0 + \text{RTT}_f, t_0 + 2\text{RTT}_f].$$

On each subsequent interval, the sending rate increases exponentially until all the n_{drop} packets that were dropped are successfully retransmitted. In k RTTs, the total number of packets retransmitted is equal to
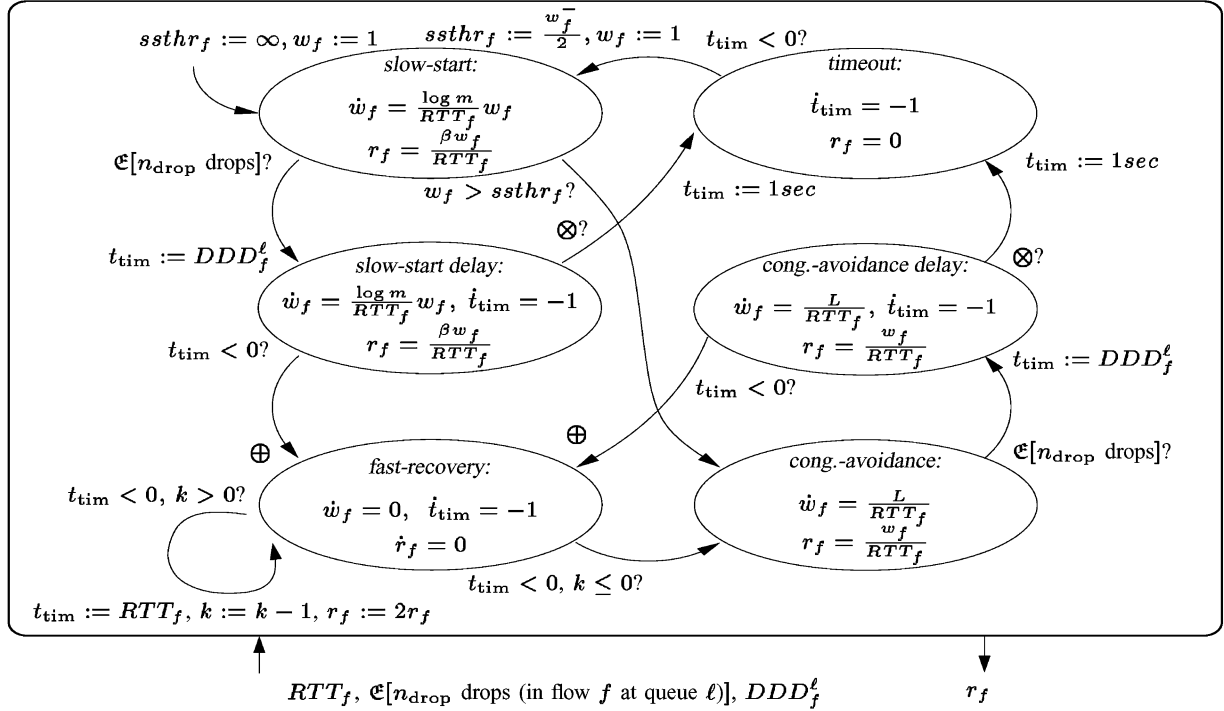
$$\sum_{i=0}^{k-1} 2^i \left(1 + w_f(t_0^-)/2 - n_{\text{drop}}\right)$$

$$= (2^k - 1)\left(1 + w_f(t_0^-)/2 - n_{\text{drop}}\right)$$

and the sender will exit fast recovery when this number reaches n_{drop}, i.e., when

$$(2^k - 1)\left(1 + w_f(t_0^-)/2 - n_{\text{drop}}\right) = n_{\text{drop}}$$

$$\Leftrightarrow \quad k = \log_2 \frac{1 + w_f(t_0^-)/2}{1 + w_f(t_0^-)/2 - n_{\text{drop}}}.$$

In practice, this means that the hybrid model should remain in the fast recovery mode for approximately

$$n\left(w_f(t_0^-), n_{\text{drop}}\right) := \left\lceil \log_2 \frac{1 + \frac{w_f(t_0^-)}{2}}{1 + \frac{w_f(t_0^-)}{2} - n_{\text{drop}}} \right\rceil \qquad (14)$$

Fig. 4. Hybrid model for the flow f under TCP. The meaning of the symbols $\otimes$ and $\oplus$ depend on the version of TCP under consideration and is shown in the table above, where $n(\cdot)$ is defined by (14) and (15).

RTTs. The previous reasoning is only valid when the number of drops does not exceed $w_f(t_0^-)/2$. As shown in [22], when $n_{\mathrm{drop}} > w_f(t_0^-)/2 + 1$, the sender does not receive enough acknowledgments in the first RTT to retransmit any other packets and there is a timeout. When $n_{\mathrm{drop}} = w_f(t_0^-)/2 + 1$, only one packet will be sent on each of the first two RTTs, followed by exponential increase in the remaining RTTs. In this case, the fast recovery mode will last approximately

$$n\left(w_f\left(t_0^-\right), n_{\mathrm{drop}}\right) := 1 + \lceil \log_2 n_{\mathrm{drop}} \rceil \qquad (15)$$

RTTs.

We have also developed *fast-recovery* models for TCP-NewReno, TCP-Reno, and TCP-Tahoe. These are discussed in [21] and [23]. The behavior of the several variants of TCP in the presence of multiple packet losses in the same window is also discussed in [24].

4) Timeouts: Timeouts occur when the timeout timer exceeds a threshold that provides a measure of the current RTT. This timer is reset to zero whenever the number of outstanding packets decreases (i.e., when it has received an acknowledgment for a new packet). Even when there are drops, this should occur at least once every RTT_f, except in the following cases.

1) The number of drops n_{drop} is larger or equal to $w_f - 2$, and, therefore, the number of duplicate acknowledgments received is smaller or equal to 2. These are not enough to trigger a transition to the *fast-recovery* mode.
2) The number of drops n_{drop} is sufficiently large so that the sender will not be able to exit fast recovery because it does not receive enough acknowledgments to retransmit all the packets that were dropped. As seen above, this corresponds to $n_{\mathrm{drop}} \geq w_f/2 + 2$.

These two cases can be combined into the following condition, under which a timeout will occur:

$$w_f \leq \max\{2 + n_{\mathrm{drop}}, 2n_{\mathrm{drop}} - 4\}.$$

When a timeout occurs at time t_0 the variable $ssthr_f$ is set equal to half the congestion window size, which is reset to 1, i.e.,

$$ssthr_f(t_0) = w_f^-(t_0)/2, \qquad w_f(t_0) = 1.$$

At this point, and until w reaches $ssthr$, we have multiplicative increase similar to what happens in slow start, and, therefore, (13) holds. This lasts until w_f reaches $ssthr_f(t_0)$ or a drop/timeout is detected. The former leads to a transition into the
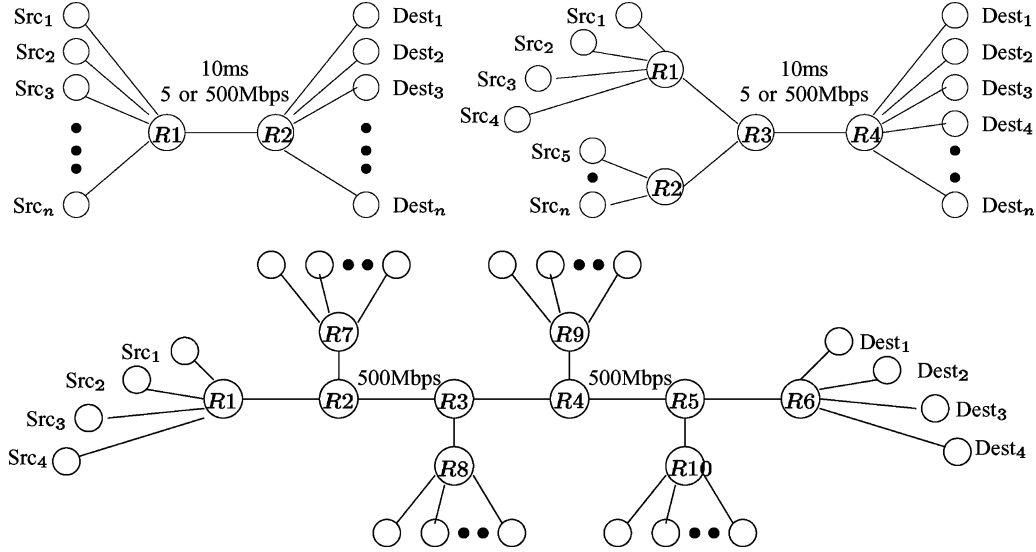
Fig. 5. (Upper left) Dumbbell, (upper right) Y-shape multiqueue with four different propagation delays, and (bottom) parking-lot with four different propagation delays topologies.

congestion avoidance mode, whereas the latter to a transition into the *fast-recovery/timeout* mode.

5) Hybrid Model for TCP-Sack: The model in Fig. 4 combines the modes described in Section III-C1–4 for TCP-Sack. This model also takes into account that there is a delay between the occurrence of a drop and its detection. This *drop-detection delay* is determined by the "roundtrip time" from the queue ℓ where the drop occurred, all the way to the receiver, and back to the sender. It can be computed using

$$\mathrm{DDD}_f^\ell := \sum_{\ell' \in \mathcal{L}[f,\ell]} \left(T^\ell + \frac{q^\ell}{B^\ell} \right)$$

where $\mathcal{L}[f,\ell]$ denotes the set of links between the ℓ-queue and the sender, passing through the receiver (for drop-tail queueing, this set should include ℓ itself). To take this delay into account, we added two modes (*slow-start delay* and *congestion-avoidance delay*), in which the system remains between a drop occurs and it is detected. The congestion controller only reacts to the drop once it exits these modes. The timing variable t_{tim} is used to enforce that the system remains in the *slow-start delay*, *congestion-avoidance delay*, and *fast-recovery* modes for the required time. For simplicity, we assumed an infinitely large advertised window size in the diagram in Fig. 4.

The inputs to the TCP-Sack flow model are the RTTs, the drop events, and the corresponding drop-detection delays (which can be obtained from the flow-conservation law and queue dynamics in Section III-A and B) and its outputs are the sending rates of the end-to-end flows.

The model in Fig. 4 assumes that the flow f is always active. It is straightforward to turn the flow on and off by adding appropriate modes [21]. In fact, in the simulation results described in Section IV-B we used random starting times for the persistent TCP flows. On–off UDP sources with a fixed sending rate during the on-period are also straightforward to construct [21].

IV. VALIDATION

We use the ns-2 (version 2.26) packet-level simulator to validate our hybrid models. Different network topologies subject to a variety of traffic conditions are considered.

A. Network Topologies

We focus our study on the topologies shown in Fig. 5. The topology in the upper left corner is known as the *dumbbell topology* and is characterized by a set of flows from the source nodes in the left to the sink nodes in the right, passing through a *bottleneck link* with 10-ms propagation delay.

While all the flows in a dumbbell topology have the same propagation delays, the flows in the Y-shape topology in the upper right corner of Fig. 5 exhibit distinct propagation delays: 45 ms (Src_1), 90 ms (Src_2), 135 ms (Src_3), and 180 ms (Src_4). In this topology, UDP background traffic is injected at Src_5 and router R2, whereas the TCP flows originate at Src_1 through Src_4. The background traffic model is described in Section IV-B.

We also consider the parking-lot topology depicted at the bottom of Fig. 5. This topology includes two 500 Mbps bottleneck links. The traffic consists of four TCP flows with propagation delays of 45, 90, 135, and 180 ms competing with 10% UDP background traffic. Two sets of background traffic were used: in the first set, traffic was injected into the sources attached to R7 and sent to the sinks attached to R8, while the second set originated at the sources connected to R9 and was sent to the sinks attached to R10. This configuration creates two bottlenecks on the links between R2 and R3 and between R4 and R5.

All queues are 40 packets long for the topologies with 5-Mbps bottleneck links and 11 250 packets for the ones with 500-Mbps bottleneck links. These queues are large enough to hold the bandwidth-delay product.

B. Simulation Environment

The hybrid models in this paper were formally specified using the object-oriented modeling language `modelica` [5]. `Modelica` allows convenient component-oriented modeling of complex physical systems. All `ns-2` simulations use TCP-Sack (more specifically its Sack1 variant outlined in [24]). Each simulation ran for 600 s of simulation time for the 5-Mbps topologies and for 8 000 s for the 500-Mbps one. Data points were obtained by averaging 20 trials for the 5-Mbps topologies and five trials for the 500-Mbps one. TCP flows start randomly between 0 and 2 s.

The background traffic consists of UDP flows with exponentially distributed ON and OFF times, both with average equal to 500 ms. We do not claim that this type of background traffic is realistic, but it suffices to reduce packet synchronization as in [25]. We considered different amounts of background traffic but in all the results reported here the background flows to account for 10% of the traffic. While the exact fraction of short-lived traffic found on the Internet is unknown, it appears that short-lived flows make up for at least 10% of the total Internet traffic [26]. However, it should be emphasized that the accuracy of the hybrid system simulations does not degrade as more short-lived traffic is considered.

As previously mentioned, the drop model is topology dependent. As observed in [20], for the single bottleneck topology with uniform propagation delays, drops are deterministic with each flow experiencing drops in a round-robin fashion. However, when background on/off traffic is considered, losses are best modeled stochastically.

The variables used for comparing the hybrid and the packet-level models include the RTTs, the packet drop rates, the throughput and congestion window size for the TCP flows, and the queue size at the bottleneck links.

C. Results

We start by considering a dumbbell topology with no background traffic for which the drop rotation model in Section III-B3 is valid. As discussed above, in such networks drops are essentially deterministic phenomena and one can directly compare `ns-2` traces with our hybrid model, without resorting to statistical analysis. Fig. 6 compares simulation results for a single TCP flow (no background traffic). These plots show traces of TCP's congestion window size and the bottleneck queue size over time. The plots show a nearly perfect match and one can easily identify the slow-start, congestion-avoidance, and fast-recovery modes discussed in Section III-C. While most previous models of TCP are able to capture TCP's steady-state behavior, TCP slow-start is typically harder to model because it often results in a large number of drops within the same window. We can observe in Fig. 6 that after the initial drops, the congestion window is divided by two and maintains this value for about half a second before it begins to increase linearly. This is consistent with the basic slow-start behavior of TCP Sack1 when the number of losses is around cwnd/2. In this case, TCP Sack1 eventually leaves fast-recovery but only after several multiples of the RTT (cf. Section III-C3 and [22]).

In the next set of experiments, we simulate four TCP flows on the dumbbell topology with and without background traffic.
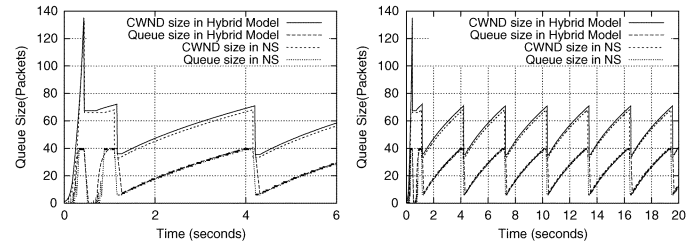

Fig. 6. Comparison of the congestion window and queue sizes over time for the dumbbell topology with one TCP flow and no background traffic.
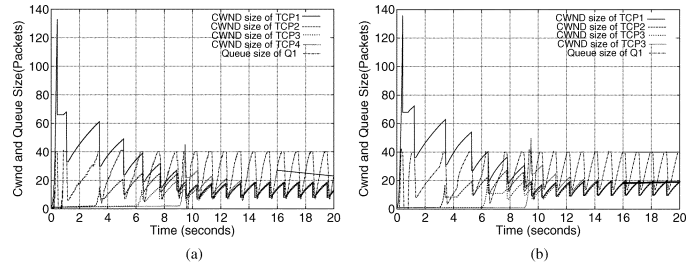

Fig. 7. Congestion window and queue size over time for the dumbbell topology with four TCP flows and no background traffic. (a) `ns-2`; (b) hybrid model.
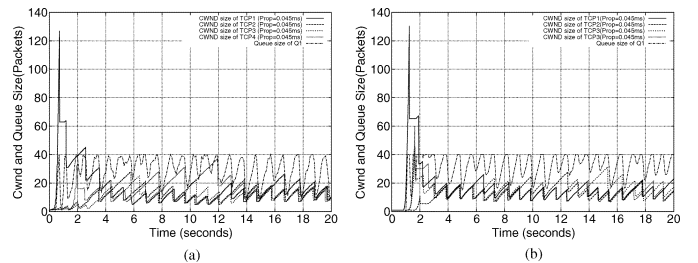

Fig. 8. Congestion window and queue size over time for the dumbbell topology with four TCP flows and 10% background traffic. (a) `ns-2`; (b) hybrid model.

Fig. 7 shows the simulation results without background traffic. As observed in previous studies, TCP connections with the same RTT get synchronized and this synchronization persists even for a large number of connections [17], [27]. This synchronization is modeled using drop rotation. Similar to the single-flow case, the two simulations coincide almost exactly. Specifically, in steady state, all flows synchronize to a saw-tooth pattern with period close to 1 s.

Simulation results for four TCP flows with background traffic are shown in Fig. 8. Even a small amount of background traffic breaks packet-drop synchronization and the stochastic drop-selection model (9) becomes valid. We can see that the traces obtained with `ns-2` are qualitatively very similar to those obtained with the hybrid model. A quantitative comparison between `ns-2` and a hybrid model is summarized in Table I, which presents average throughput and RTT for each flow for both hybrid system and `ns-2` simulations. These statistics confirm that the hybrid model reproduces accurately the results obtained with the packet-level simulation.

To validate our hybrid models, we also use the Y-shape, multiqueue topology with different RTTs shown on the right-hand side of Fig. 5. We consider the drop-count and drop-selection models described by (8) and (9), respectively, which generate

TABLE I
AVERAGE THROUGHPUT AND RTT FOR THE DUMBBELL TOPOLOGY WITH 4 TCP FLOWS AND 10% BACKGROUND TRAFFIC

	$Thru_1$	$Thru_2$	$Thru_3$	$Thru_4$	RTT_1	RTT_2	RTT_3	RTT_4
ns-2	1.13	1.14	1.15	1.14	0.097	0.097	0.097	0.097
hybrid system	1.14	1.14	1.13	1.14	0.096	0.096	0.096	0.096
relative error	0.9%	0%	1.3%	0%	1%	1%	1%	1%

TABLE II
AVERAGE THROUGHPUT AND RTT FOR THE Y-SHAPE TOPOLOGY (5 Mbps) WITH FOUR TCP FLOWS AND 10% BACKGROUND TRAFFIC

	$Thru_1$	$Thru_2$	$Thru_3$	$Thru_4$	RTT_1	RTT_2	RTT_3	RTT_4
ns-2	1.913	1.134	0.817	0.668	0.091	0.136	0.182	0.225
hybrid model	1.816	1.162	0.876	0.680	0.093	0.138	0.183	0.228
relative error	5.0%	2.4%	6.7%	1.0%	2.1%	1.5%	0.5%	1.3%

stochastic drops. Since losses are random, no two simulations will be exactly equal so one cannot expect the hybrid model to exactly reproduce the results from ns-2. Table II summarizes the simulation results obtained with ns-2 and the hybrid model for four TCP flows with 10% background traffic on the Y-shape topology under a drop-tail queueing discipline. This table presents the mean throughput and mean RTTs for each competing TCP flow. The relative error is always less than 10% and in most cases well under this value. Similar results hold for variations of the Y-shape topology with different RTTs and different numbers of competing flows. However, for the stochastic drop model to hold, there must be background traffic and/or enough complexity in the topology and flows such that synchronization does not occur. When synchronization does occur, then a deterministic model for drops needs to be used. As described in Section III-B3, in single bottleneck topologies drop-rotation provides an accurate model. In more complex settings, the construction of drop models for synchronized flows appears to be quite challenging. This is one direction of future work we plan to pursue.

To accurately compare stochastic processes one should examine their probability density functions. Fig. 9 plots the probability density functions corresponding to the time-series used to generate the results in Table II. We observe that the hybrid model reproduces fairly well the probability densities obtained with ns-2. For the congestion window, three of the flows closely agree, while one shows a slight bias towards larger values. The density function of the queue is similar for both models. One noticeable difference is that the peak near the queue-full state is sharper for the hybrid model. This is due to the fact that the queue in ns-2 can only take integer values, while in the hybrid model it can take fractional values. Thus, the probability that the queue is nearly full is represented by a probability mass at $q_{max} - \varepsilon$ for the hybrid model, while it is represented by a probability mass at $q_{max} - 1$ in ns-2. This results in a more smeared probability mass around queue-full for ns-2.

We also validate the hybrid models in high bandwidth networks with drop-tail queueing. These networks are especially challenging because, due to the larger window sizes, they are more prone to synchronized losses even when the drop rates are small [28]. Also, TCP's unfairness towards connections with higher propagation delays is more pronounced in high bandwidth-delay networks where synchronization occurs [29]. We simulate dumbbell, Y-shape, and parking-lot topologies with a
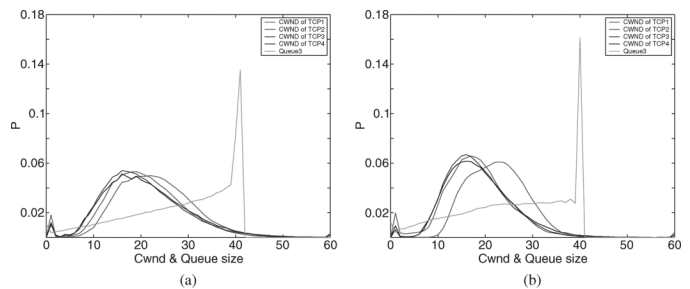

Fig. 9. Probability density functions for the congestion window and queue size for the Y-shape topology with four TCP flows and 10% background traffic. (a) ns-2; (b) hybrid model.

bottleneck of 500 Mbps and 10% background traffic. The bottleneck queues are set to be large enough to hold the bandwidth-delay product.

Table III presents the mean throughput and mean RTT for each competing TCP flow for the dumbbell, Y-shape, and parking-lot topologies with 500-Mbps bottleneck(s). The relative errors between the results obtained with ns-2 and the hybrid models are always smaller than 10%. The corresponding probability density functions for the congestion window and queue size for the Y-shape and parking-lot topologies are given in Figs. 10 and 11, respectively. In both cases, the probability density functions match fairly well.

It is interesting to compare the distributions of the bottleneck queue and congestion window sizes for the low-bandwidth Y-shape topology in Fig. 9 with those obtained for the high bandwidth Y-shape topologies in Fig. 10. The explanation for the significant differences observed lie in the frequent synchronized losses that occur in the high bandwidth networks [28]. Note that, when the probability of synchronized loss is higher, the bottleneck queue size exhibits larger variations because more flows are likely to backoff approximately at the same time. It is, thus, not surprising to observe that in high-speed networks the queue size distribution is less concentrated around the queue-full state [25].

Fig. 11 shows the probability density functions for the congestion window and queue sizes for the 500-Mbps bottleneck parking-lot topology. Unlike in the 5-Mbps dumbbell or Y-shape topologies where bottleneck queues are not empty most of the time, in this high-speed, multiple bottleneck topology, queues become empty more frequently producing

TABLE III
AVERAGE THROUGHPUT AND AVERAGE RTT FOR Y-SHAPE AND PARKING-LOT TOPOLOGY FOR 500 Mbps BOTTLENECK

		$Thru_1$	$Thru_2$	$Thru_3$	$Thru_4$	RTT_1	RTT_2	RTT_3	RTT_4
Dumbbell	ns-2	113.6	113.37	112.47	110.31	0.085	0.085	0.085	0.085
	hybrid model	113.6	114.08	112	110.08	0.0819	0.0819	0.0819	0.0819
	relative error	0%	0.6%	0.4%	0.2%	3.6%	3.6%	3.6%	3.6%
Y-shape	ns-2	201.99	116.90	76.80	54.38	0.159	0.204	0.249	0.295
	hybrid model	196.40	113.76	81.44	58.96	0.158	0.203	0.248	0.293
	relative error	2.8%	2.7%	6.0%	8.4%	0.6%	0.5%	0.4%	0.6%
parking-lot	ns-2	197.8	117.5	78	55.2	0.158	0.203	0.248	0.293
	hybrid model	194	126.4	74.92	53.24	0.162	0.206	0.252	0.301
	relative error	1.95%	7.6%	4.1%	3.68%	2.5%	1.5%	1.6%	2.78%

TABLE IV
L^1-DISTANCE BETWEEN HISTOGRAMS COMPUTED FROM SIMULATIONS USING ns-2 AND HYBRID MODEL

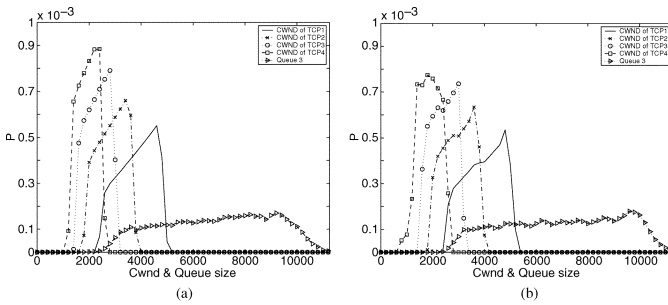|	cwnd1	cwnd2	cwnd3	cwnd4	bottleneck queue 1	bottleneck queue 2
dumbbell (5Mbps)	0.1951	0.1764	0.1513	0.1771	0.2625	-
Y-shape (5Mbps)	0.1040	0.0783	0.2213	0.0416	0.3333	-
parking-lot (5Mbps)	0.1128	0.1054	0.1749	0.1546	0.1691	02061
Dumbbell (500Mbps)	0.2138	0.2277	0.2354	0.1707	0.1836	-
Y-shape (500Mbps)	0.2226	0.1950	0.1935	0.1852	0.1276	-
parking-lot (500Mbps)	0.1206	0.0682	0.0511	0.0873	0.1313	0.1238

Fig. 10. Probability density functions for the congestion window and queue size for the Y-shape topology with four TCP flows and 10% background traffic (500 Mbps bottleneck). (a) ns-2; (b) hybrid model.
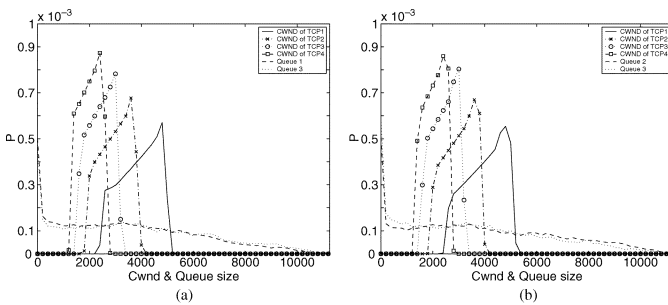

Fig. 11. Probability density functions for the congestion window and queue size for the parking-lot topology (500 Mbps bottleneck) with four TCP flows and 10% background traffic. These were computed from simulations using ns-2 (left) and a hybrid model (right).

a more chaotic behavior. However, the hybrid model still reproduces well the probability densities obtained from ns-2 simulations.

While visually comparing two density functions provides a qualitative understanding of their similarity, there are several techniques to compare density functions quantitatively. One well-established metric is the L^1-distance $|\cdot|_1$ [30], which

has the desirable property that when f is a density and $\hat{f}$ an estimate of f

$$|f - \hat{f}|_1 = 2\sup_A \left| \int_A f - \int_A \hat{f} \right|.$$

Thus, if the probability of an event A is to be predicted using $\hat{f}$, the prediction error is never larger than half of the L^1-distance between f and $\hat{f}$. Table IV shows the L^1-distance between all the distributions compared in Figs. 9–11. The largest L^1-distance is 0.3333, which corresponds to a maximum error of 0.1667 in probability.

V. COMPUTATIONAL COMPLEXITY

Modern ordinary differential equation (ODE) solvers are especially efficient when the state variables are continuous functions. However, the state variables of hybrid systems exhibit occasional discontinuities, which requires special care and can lead to significant computational burden. In fact, the simulation time of hybrid systems typically grows linearly with the number of discontinuities in the state variables because each discontinuity typically requires the integration step of the ODE solver to be interrupted so that the precise timing of the discontinuity can be determined. Between discontinuities, the integration step typically grows rapidly and the simulation is quite fast, as long as the ODEs are not-stiff. In our models, these discontinuities are mainly caused by two types of discrete events: drops and queues becoming empty. Drops typically cause TCP to abruptly decrease the congestion window, whereas a queue becoming empty forces the outgoing bit-rates to switch from a fraction of the outgoing link bandwidth to the incoming bit-rates. In practice, the frequencies at which these events occur are essentially determined by the drop-rates of the active flows and the rate at which flows start and stop.
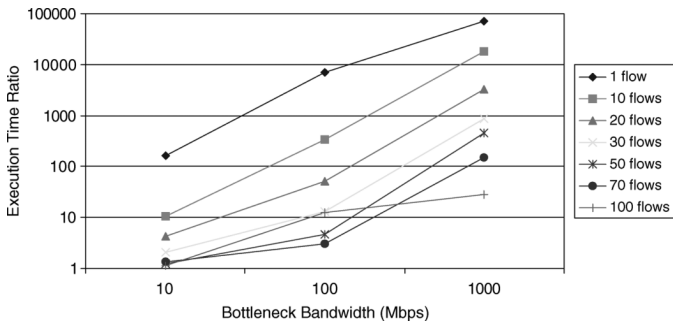
Fig. 12. Execution time speedup of hybrid model over `ns-2` simulation for dumbbell topology with 100-ms propagation delay bottleneck.
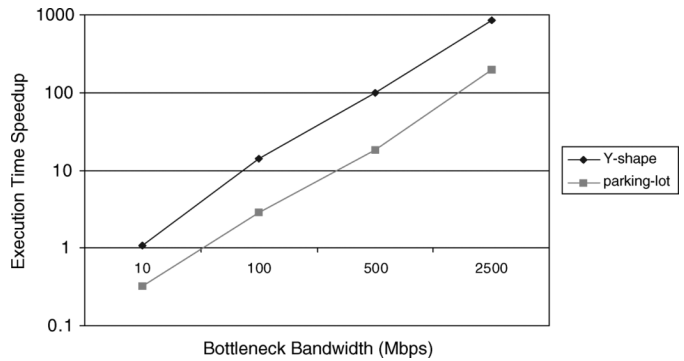

Fig. 13. Execution time speedup of hybrid model over `ns-2` simulation for Y-shape and parking-lot topologies with background traffic.

Fig. 14. Abilene backbone network.

Since the main factor that determines the simulation speed is the drop rate, it is informative to study how it scales with the number of flows. To this effect consider the well-known equation $T = (c/\mathrm{RTT}\sqrt{p})$, which relates the per-flow throughput T, the average RTT, and the drop probability p. According to this formula the total drop rate for n competing flows, which is equal to nTp, is given by $(n/T)(c^2/\mathrm{RTT}^2)$. This suggests that the computational complexity is of order $O(n/T)$, scaling linearly with the number of flows when the per-flow throughput is maintained constant and is actually inversely proportional to the per-flow throughput when the number of flows remains constant. This is in sharp contrast with event-based simulators for which the computational complexity is essentially determined by the total number of packets transmitted, which is of order $O(nT)$.

This analysis is confirmed by the data in Fig. 12. This figure shows the execution time speedup defined as the ratio between the execution time of a `ns-2` packet-level simulation divided by the execution time of the corresponding hybrid model simulation in `modelica` [5]. These results correspond to a single-bottleneck topology where the bottleneck link's propagation delay is 100 ms and its bandwidth varied among 10 Mbps, 100 Mbps, and 1 Gbps. We simulate from 1 to 100 long-lived TCP flows competing for the bottleneck bandwidth for 30 min of simulation time. Simulations ran on a 1.7-GHz PC with 512-MB memory. We can see that the hybrid model simulation is especially attractive for large per-flow throughput, for which the speedup can reach several orders of magnitude.

The execution time speedups for the Y-shape and parking-lot topologies with background traffic are shown in Fig. 13. The speedup for the Y-shape topology is larger than that of the parking-lot topology due to the fact that queues empty more frequently in the latter, resulting in more discontinuities.

Memory usage can also be a concern when simulating large, complex networks. Hybrid systems require one state variable for each active flow and one state variable for each flow passing through a queue. Hence, the memory usage scales linearly with the number of flows and the number of queues. For `ns-2`, the memory usage depends on the number of packets in the system and, hence, scales with the bandwidth-delay product.

VI. TOOLS AND CASE STUDY

Hybrid systems modeling languages such as `modelica` [5] are special-purpose languages designed to model complex physical systems. To simplify the use of hybrid modeling by networking researchers, we developed the NDSL to specify succinctly large, complex networks using a syntax similar to object oriented TCL (OTCL) in `ns-2`.

NDSL primitives include node and link definitions, as well as the parameters that define the different end-to-end congestion control protocols. Details of NDSL primitives can be found in [23]. A *NDSL translator* automatically converts a network NDSL specification into a hybrid model expressed in `modelica`. In the remainder of this section, we illustrate the use of these tools in the simulation of TCP flows over a realistic high-bandwidth network for which packet-level simulations would be prohibitively long.

A. Case Study: Abilene Backbone Network

The Abilene Network (shown in Fig. 14) is an Internet-2 high-performance backbone network connecting research institutions to enable the development of advanced Internet applications and protocols [6]. Recently, its has been upgraded to 10-Gbps backbone links using OC-192 circuits. The links propagation delays considered are shown in Table V. All links have a bandwidth of 10 Gbps and we assume droptail queues with size equal to 25 000 packets with 1-K packet size. In this experiment, we simulate the three sets of ten flows described in Table VI. Each flow starts randomly between 0 and 1 s and terminates at time 40 000 s.

TABLE V
TWO-WAY PROPAGATION DELAY BETWEEN NODES IN THE ABILENE BACKBONE

source	destination	prop. delay
Seattle (STTL)	Denver (DNVR)	25.608ms
Sunnyvale (SNVA)	Denver (DNVR)	25.010ms
Denver (DNVR)	Kansas City (KSCY)	10.674ms
Kansas City (KSCY)	Indianapolis (IPLS)	9.340ms
Indianapolis (IPLS)	Chicago (CHIN)	3.990ms
Chicago (CHIN)	New York (NYCM)	20.464ms
Sunnyvale (SNVA)	Los Angeles (LOSA)	7.772ms
Los Angeles (LOSA)	Houston (HSTN)	31.624ms
Houston (HSTN)	Atlanta (ATLA)	19.756ms
Atlanta (ATLA)	Washington (WASH)	15.938ms
Washington (WASH)	New York (NYCM)	4.412ms
Sunnyvale (SNVA)	Seattle (STTL)	16.852ms
Houston (HSTN)	Kansas City (KSCY)	15.504ms
Atlanta (ATLA)	Indianapolis (IPLS)	10.950ms

TABLE VI
TCP FLOWS SIMULATED OVER ABILENE BACKBONE

sets	number of flows	prop. delay	source/destination
set one	10	15 ms	ATLA to CHIN
set two	10	28.8 ms	HSTN to CHIN
set three	10	69.5 ms	SNVA to NYCM

B. Results

We use our hybrid model of the Abilene network to study how queue size impacts throughput fairness. To this effect we vary the queue sizes from 25 000 to 150 000 packets in increments of 25 000 and measure the throughput obtained. We ran 11 h of simulation time. In this network, one needs simulations this large if one wants to obtain steady-state throughput. Note that for a 10-Gbps backbone with 70-ms RTT and 1000-byte packet size, the bandwidth-delay product is 87 500 packets. When the queue size is as large as the bandwidth-delay product, the maximum window size before a packet is drops is 175 000 packets. If the sender detects a congestion loss at this time, the window size reduces from 175 000 to 87 500. Thus, it takes 87 500 RTTs to get another drop, which amounts to 1 h and 42 min. Simulation times as long as this are not feasible in `ns-2` with our 512-MB memory PC. However, the hybrid systems simulation requires no more than 20 min of execution time. It should be noted that versions of TCP adapted to high-bandwidth networks, such as FAST-TCP and HSTCP, reach steady-state much faster than this, and, in fact, we currently have hybrid models for these. However, due to space limitations we do not describe those here.

Fig. 15 shows the fairness ratio between flows in different sets (cf. Table VI). The *fairness ratio* $FR_{i,j}$ is defined as the ratio between the average throughput of the flows in sets i divided by the average throughput of the flows in j. When the queue size is 25 000, the average throughput of set one is 3.1 times the average throughput of set three, but when the queue size increases to 150 000, the throughput ratio becomes only 1.5. This is consistent with the expectation that, when the queueing delay increases considerably, it will dominate the RTT, thus decreasing the RTT ratio between the two flows with different propagation delays. However, in topologies like this one, the precise dependence of the fairness ratio with the buffer size is difficult to predict without resorting to simulations.
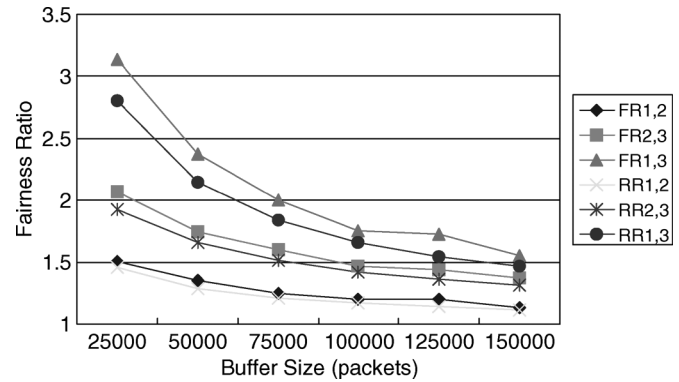

Fig. 15. Average throughput fairness between the three different TCP-Sack flow sets simulated on the Abilene network.

Fig. 15 also shows the ratio $RR_{i,j}$ between the average RTTs of the flows in sets i and j (in the reciprocal order). Since all the flows go through the same bottleneck (Chicago-Indianapolis), based on the TCP-friendly formula one could expect the fairness ratio to match the reciprocal of the RTT ratio. The simulations reveal that this generally underestimates the fairness ratio, especially when the ratio is far from one. This phenomena has been confirmed by `ns-2` simulations in smaller networks.

VII. CONCLUSION AND FUTURE WORK

This paper proposes a general framework for building hybrid models to describe network behavior. This framework fills the gap between packet-level and aggregate models by averaging discrete variables over very short time scales. This means that the models are able to capture the dynamics of transient phenomena fairly accurately, as long as their time constants are larger than a couple of RTTs. This is quite appropriate for the analysis and design of network protocols including congestion control mechanisms.

To validate our hybrid systems modeling framework, we compare hybrid model against packet-level simulations and show that the probability density functions match very closely. We also briefly describe the software tools that we developed to automate the generation of hybrid models for complex networks. We showcased their use with a case study involving the Abilene backbone network.

Our results indicate that simulations using hybrid models should be preferred over packet-level simulators in the study of *networks with large per-flow bandwidths* when one wants to *accurately capture traces of individual flows and the evolution of buffer sizes.* For networks with small bandwidth, the computational saving introduced by hybrid model are small and one might as well rely on packet-level simulators.

REFERENCES

[1] *"The ns Manual (Formerly ns Notes and Documentation)"* Oct. 2000 [Online]. Available: http://www.isi.edu/nsnam/ns/ns-documentation.html, The VINT Project.

[2] V. Misra, W. Gong, and D. Towsley, "Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED," presented at the ACM SIGCOMM, Sep. 2000.

[3] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based congestion control for unicast applications," in *Proc. ACM SIGCOMM*, Aug. 2000, pp. 43–56.

[4] A. V. D. Schaft and H. Schumacher, *An Introduction to Hybrid Dynamical Systems*, ser. Lecture Notes Contr. Inf. Sci.. London, U.K.: Springer-Verlag, 2000, 251.

[5] "Modelica—A unified object-oriented language for physical systems modeling: Tutorial," Modelica Assoc. [Online]. Available: http://www.modelica.org/

[6] Abilene [Online]. Available: http://abilene.internet2.edu

[7] QualNet [Online]. Available: http://www.scalable-networks.com

[8] Scalable Simulation Framework [Online]. Available: http://www.ssfnet.org/

[9] F. Desbrandes, S. Bertolotti, and L. Dunand, "Opnet 2.4: An environment for communication network modeling and simulation," in *Proc. Eur. Simulation Symp.*, Oct. 1993, pp. 609–614.

[10] M. Mathis, J. Semke, J. Mahdavi, and T. Ott, "The macroscopic behavior of the TCP congestion avoidance algorithm," *ACM Comput. Commun. Rev.*, vol. 27, no. 3, July 1997.

[11] S. H. Low, F. Paganini, J. Wang, S. Adlakha, and J. C. Doyle, "Dynamics of TCP/RED and a scalable control," presented at the IEEE INFOCOM, Jun. 2002.

[12] Y. Guo, W. Gong, and D. Towsley, "Time-stepped hybrid simulation (TSHS) for large scale networks," presented at the IEEE INFOCOM, Mar. 2000.

[13] K. Kumaran and D. Mitra, "Performance and fluid simulations of a novel shared buffer management system," in *Proc. IEEE INFOCOM*, Mar. 1998, pp. 1449–1461.

[14] B. Liu, D. R. Figueiredo, J. K. Yang Guo, and D. Towsley, "A study of networks simulation efficiency: Fluid simulation vs. packet-level simulation," in *Proc. IEEE INFOCOM*, Apr. 2001, vol. 3, pp. 1244–1253.

[15] D. Schwetman, "Hybrid simulation models of computer systems," *Commun. ACM*, vol. 19, pp. 718–723, Sep. 1978.

[16] R. Pan, B. Prabhakar, K. Psounis, and D. Wischik, "Shrink: A method for scaleable performance prediction and efficient network simulation," presented at the IEEE INFOCOM, Mar. 2003.

[17] L. Qiu, Y. Zhang, and S. Keshav, "Understanding the performance of many TCP flows," *Comput. Netw.*, vol. 37, pp. 277–306, Nov. 2001.

[18] S. Floyd, "Connections with multiple congested gateways in packet-switched networks Part 1: One-way traffic," *ACM Comput. Commun. Rev.*, vol. 21, no. 5, pp. 30–47, Oct. 1991.

[19] S. Shenker, L. Zhang, and D. Clark, "Some observations on the dynamics of a congestion control algorithm," *ACM Comput. Comm. Rev.*, pp. 30–39, Oct. 1990.

[20] S. Bohacek, J. P. Hespanha, J. Lee, and K. Obraczka, "A hybrid systems modeling framework for fast and accurate simulation of data communication networks," presented at the ACM SIGMETRICS, 2003.

[21] J. Lee, S. Bohacek, J. P. Hespanha, and K. Obraczka, "Modeling data communication networks using hybrid systems: Extended version," Tech. Rep., Univ. California, Santa Barbara, Apr. 2006.

[22] B. Sikdar, S. Kalyanaraman, and K. Vastola, "Analytic models for the latency and steady-state throughput of TCP Tahoe, Reno and SACK," in *Proc. IEEE GLOBECOM*, 2001, pp. 25–29.

[23] Hybrid systems modeling framework web site [Online]. Available: http://cs.sookmyung.ac.kr/~jslee/hybrid

[24] K. Fall and S. Floyd, "Simulation-based comparisons of Tahoe Reno and SACK TCP," *ACM Comput. Comm. Rev.*, vol. 27, no. 3, pp. 5–21, Jul. 1996.

[25] Y. Joo, V. Ribeiro, A. Feldmann, A. Gilbert, and W. Willinger, "The impact of variability on the buffer dynamics in IP networks," presented at the 37th Annu. Allerton Conf. Communications, Control, Computing, Sep. 1999.

[26] F. Hernández-Campos, J. S. Marron, G. Samorodnitsky, and F. D. Smith, "Variable heavy tail duration in internet traffic," presented at the IEEE/ACM MASCOTS, 2002.

[27] L. Zhang, S. Shenker, and D. D. Clark, "Observations on the dynamics of a congestion control algorithm: The effects of two-way traffic," presented at the ACM SIGCOMM, Sep. 1991.

[28] L. Xu, K. Harfoush, and I. Rhee, "Binary increase congestion control for fast long-distance networks," presented at the IEEE INFOCOM, Mar. 2004.

[29] T. V. Lakshman and U. Madhow, "The performance of TCP/IP for networks with high bandwidth-delay products and random loss," *IEEE/ACM Trans. Netw.*, vol. 5, no. 3, pp. 336–350, Jul. 1997.

[30] L. Devroye, *A Course in Density Estimation*. Boston, MA: Birkhauser, 1987.

Junsoo Lee received the B.S. and M.S. degrees in computer science from Seoul National University, Seoul, Korea, and the Ph.D. degree in computer science from University of Southern California, Los Angeles, in 2004.

He is currently an Assistant Professor of computer science at Sookmyung Women's University of Seoul. Before joining Sookmyung Women's University, he was a Postdoctoral scholar at the University of California, Santa Barbara, and the University of California, Santa Cruz. He was also with Samsung Electornics as a Senior Researcher in the Network and Computer Division before pursuing the Ph.D. degree.

Stephan Bohacek (M'00) received the B.S. degree in electrical engineering from the University of California, Berkeley, in 1989, and the Ph.D. degree in electrical engineering from the University of Southern California, Los Angeles, in 1999.

He is currently an Assistant Professor in the Department of Electrical and Computer Engineering, University of Delaware, Newark . His research focuses on the design, analysis, and control of data networks. His current interests include congestion control and routing for wireless and wireline networks, modeling mobile wireless networks, and cross-layer design for wireless networks.

João P. Hespanha (SM'95–A'98–SM'02) received the Ph.D. degrees in electrical engineering and applied science from Yale University, New Haven, CT, in 1998.

He currently holds an Associate Professor position with the Department of Electrical and Computer Engineering, University of California, Santa Barbara. His current research interests include hybrid and switched systems, the modeling and control of communication networks, distributed control over communication networks, stochastic modeling in biology, and game theory. More information about his research can be found at http://www.ece.ucsb.edu/~hespanha.

Katia Obraczka (S'90–M'92) received the Ph.D. degree in computer science from the University of Southern California (USC), Los Angeles, in 1994.

She is currently an Associate Professor of computer engineering at the University of California, Santa Cruz (UCSC). Before joining UCSC, she held a reserach scientist position at USC's Information Sciences Institute and a research faculty appointment at USC's Computer Science Department. Her research interests include computer networks, more specifically, network protocol design and evaluation in wireline, as well as wireless (in particular, multihop ad hoc) networks, distributed systems, and Internet information systems. More information on her research can be found at http://www.soe.ucsc.edu/~katia.