A Hybrid Systems Modeling Framework for Fast and Accurate Simulation of Data Communication Networks[*]

Stephan Bohacek
Dept. Electrical &
Comp. Eng.
Univ. of Delaware
Newark

bohacek@eecis.udel.edu

João P. Hespanha
Dept. Electrical &
Comp. Eng.
Univ. of California
Santa Barbara

hespanha@ece.ucsb.edu

Junsoo Lee
Dept. Comp. Science
Univ. of Southern
California
Los Angeles

junsoole@usc.edu

Katia Obraczka
Computer Engineering
Dept.
Univ. of California
Santa Cruz

katia@cse.ucsc.edu

ABSTRACT

In this paper we present a general hybrid systems modeling framework to describe the flow of traffic in communication networks. To characterize network behavior, these models use averaging to continuously approximate discrete variables such as congestion window and queue size. Because averaging occurs over short time intervals, one still models discrete events such as the occurrence of a drop and the consequent reaction (e.g., congestion control). The proposed hybrid systems modeling framework fills the gap between packet-level and fluid-based models: by averaging discrete variables over a very short time scale (on the order of a round-trip time), our models are able to capture the dynamics of transient phenomena fairly accurately. This provides significant flexibility in modeling various congestion control mechanisms, different queuing policies, multicast transmission, etc. We validate our hybrid modeling methodology by comparing simulations of the hybrid models against packet-level simulations. We find that the probability density functions produced by `ns-2` and our hybrid model match very closely with an L^1-distance of less than 1%. We also present complexity analysis of `ns-2` and the hybrid model. These tests indicate that hybrid models are considerably faster.

Categories and Subject Descriptors

C.2.5 [**Computer-Communication Networks**]: Local and Wide-Area Networks

General Terms

Algorithms, Performance

Keywords

Data Communication Networks, Congestion Control, TCP, UDP, Simulation, Hybrid Systems

1. 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. The most accurate network models are packet-level models that keep track of individual packets as they travel across the network. These are used in network simulators such as `ns-2` [32]. Packet-level models 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 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 [21, 22, 9]. 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. For instance, 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 [29] which combine both continuous time dynamics and discrete-time 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 (e.g., 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 very short time scale (on the order of a round-trip time). 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 round-trip times. This time scale is quite 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 round-trip time. 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 round-trip time. However, since the variations on the round-trip time typically occur at a slower time scale, the hybrid models can still capture quite accurately the dynamics of round-trip time evolution. In fact, that is one of the strengths of the models proposed here, i.e., the fact that they do not assume constant round-trip time.

We validate our hybrid modeling methodology by comparing results from hybrid models against packet-level simulations. We run extensive simulations using different network topologies subject to different traffic conditions (including background traffic). Our results show that the model is able to reproduce packet-level simulations quite accurately. We also compare the efficiency 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.

2. RELATED WORK

Several approaches to modeling and simulating networks exist, some of which have been widely used by the networking community in the design and evaluation of network protocols. On one side of the spectrum, there are packet-level simulation models. For example, ns [32], QualNet [27], SSFNET [28] are event simulators where an event is the arrival or departure of a packet. These models are highly accurate, but are not scalable to large networks. On the other extreme, static models provide approximations using first principles. For example, in [8, 24, 25, 20, 17], simple formulas are derived that model how TCP behaves. This approach has been extended to the case of short-lived flows [5]. These models ignore much of the dynamics of the network. For example, the round-trip time and loss probability are assumed constant and the interaction of flows is not considered.

Between static models and detailed packet level simulators are dynamic fluid flow models. 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 more dynamic modeling approach was followed by [19, 16, 12] where TCP's sending rate is taken as an ensemble average. Specifically, the sending rates do not suffer the linear increase and divide in half. However, this ensemble average dynamically varies with queue size and drop probability. For example, [21, 22] present a stochastic differential equation (SDE) model of TCP where the sending rate linearly increases until a drop event occurs and then divides in half. Along these lines, [2] developed an SDE model that allows the round-trip time to vary and includes more accurate loss models. While these SDE approaches make sense from an end-to-end perspective, they are difficult to justify for network models. The main source of the problem is that, from the network perspective, drops among flows are highly dependent. Such interdependence is difficult to efficiently incorporate into the SDE approach.

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, [33, 10] develop a fluid approach for efficient network simulation, which assumes bit rates to be piecewise constant. This piecewise constant assumption seems to have a major impact and can lead to an explosion of events known as the ripple effect [18]. A similar direction is taken in [1] where packets are aggregated. Again, during a single time step, the behavior of the set of packets is uniform across all packets in the set. The approach we present is somewhat similar to the one in [14, 15], where sending rates vary *continuously*. However, our approach also allows for discrete jumps in the sending rates.

Systems that contain both continuous-time state variable and discrete-time events causing discontinuities are known as hybrid systems [29]. Such modeling approach has been widely used in other fields, but is new to networking. A hybrid modeling approach is taken in [30]. In that model, both discrete-event simulation and analytic techniques are combined to produce efficient, yet accurate models of job queues and processing on a multi-user computer system. Our approach of applying hybrid modeling to networking, in particular congestion control, is conceptually close to the work in [30]. [13] applied hybrid simulation techniques to perform large-scale multicast simulations at low computational cost.

The remainder of the paper is organized as follows. Section 3 presents our hybrid systems modeling framework, including hybrid models for a number of TCP variants (i.e., SACK, New Reno, and Reno), and UDP flows. In Section 4, we validate our hybrid models by comparing them to packet-level simulations. Section 5 shows results comparing the computational complexity of hybrid- and packet level models. Finally, we present our concluding remarks and directions for future work in Section 6. The reader is referred to the Technical report [4] for additional details that were not included due to space restrictions.

3. HYBRID MODELING FRAMEWORK

Consider a communication network consisting of a set $\mathcal{N}$ of *nodes* connected by a set $\mathcal{L}$ of *links*. We consider all links as unidirectional and denote by $\ell := \vec{ij}$, the link from node $i \in \mathcal{N}$ to node $j \in \mathcal{N}$ (cf. Figure 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 f-*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 go through the ℓ-link. We call r_f^ℓ the ℓ-*link/f-flow*
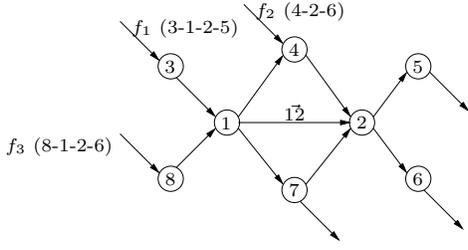
Figure 1: Example network where $q_{\vec{12}} = q_{\vec{12}}^{f_1} + q_{\vec{12}}^{f_3}$.

transmission rate. At each link, the sum of the link/flow transmission rates must not exceed the bandwidth B^ℓ, i.e.,

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

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.

Associated with each link $\ell \in \mathcal{L}$, there is a *queue* that temporarily holds packets before transmission. 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 then 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 $q_{\max}^\ell$. When q^ℓ reaches $q_{\max}^\ell$, drops will occur.

For each flow $f \in \mathcal{F}$, we denote by RTT_f the *f-flow round-trip-time* that elapses between a packet is sent and its acknowledgment is received. The round-trip-time can also be determined by adding the propagation delays T^ℓ and queuing times q^ℓ/B^ℓ of all links involved in one round-trip. In particular,

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

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

3.1 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 are related to the flow sending rates and the link/flow transmission rates 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 here single-path routing and unicast transmission. It would be straightforward to derive conservation laws for multi-path routing and multi-cast transmission.

3.2 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). *On a short time-interval over which the arrival rates can be assumed approximately constant, the packets of the different flows arrive at each node in their paths uniformly distributed over time.*

ASSUMPTION 2 (QUEUE UNIFORMITY). *Packets of the different flows are uniformly distributed in each queue.*

Clearly, because of packet quantization, bursting, synchronization, etc., these assumptions are never quite true. However, they are sufficiently accurate to lead to models that match closely packet-level simulations. We will show this in Section 4.2.

Consider a link $\ell \in \mathcal{L}$ that is in the path of 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., $q^\ell = 0$). 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. In case $r_f^\ell = s_f^\ell$, $\forall f \in \mathcal{F}$ would violate (1), the available link bandwidth B^ℓ is distributed among all flows proportionally to their arrival rates s_f^ℓ, which is justified by the Arrival Uniformity Assumption 1. This can be summarized as follows: for every $f \in \mathcal{F}$,

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

2. *Queue neither empty nor full* (i.e., $0 < q^\ell < q_{\max}^\ell$ or $q^\ell = q_{\max}^\ell$ but $\sum_{\bar{f} \in \mathcal{F}} s_{\bar{f}}^\ell \le B^\ell$). There are no drops and because of the Queue Uniformity Assumption 2, the available link bandwidth B^ℓ is distributed among the flows proportionally to their percentage of bytes in the queue, i.e., for every $f \in \mathcal{F}$,

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

3. *Queue full and still filling* (i.e., $q^\ell = q_{\max}^\ell$ and $\sum_{\bar{f} \in \mathcal{F}} s_{\bar{f}}^\ell > B^\ell$). The total drop rate d^ℓ must equal the difference between the total arrival rate and the link bandwidth, i.e., $d^\ell = \sum_{\bar{f} \in \mathcal{F}} s_{\bar{f}}^\ell - B^\ell > 0$. From the Arrival Uniformity Assumption 1, we conclude that this total drop rate d^ℓ should be distributed among all flows proportionally to their arrival rates s_f^ℓ. Moreover, from the Queue Uniformity Assumption 2, we conclude that the available link bandwidth B^ℓ is distributed among the flows proportionally to their the

percentage of bytes in the queue. This can be summarized as follows: for every $f \in \mathcal{F}$,

$$d_f^{\ell} = \frac{s_f^{\ell}\left(\sum_{\bar{f} \in \mathcal{F}} s_{\bar{f}}^{\ell} - B^{\ell}\right)}{\sum_{\bar{f} \in \mathcal{F}} s_{\bar{f}}^{\ell}}, \quad r_f^{\ell} = \frac{q_f^{\ell} B^{\ell}}{\sum_{\bar{f} \in \mathcal{F}} q_{\bar{f}}^{\ell}}. \quad (4)$$

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}^{ℓ} with $s^{\ell} := \sum_{f \in \mathcal{F}} s_f^{\ell} > B^{\ell}$. Clearly, a drop will occur at time t_1 but, in general, 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 the packet-size L, i.e., for which

$$z^{\ell} := \int_{t_k}^{t_{k+1}} \sum_{f \in \mathcal{F}} d_f^{\ell} = \int_{t_k}^{t_{k+1}} \left(\sum_{f \in \mathcal{F}} s_f^{\ell} - B^{\ell}\right) = L. \quad (5)$$

We call (5) the *drop-count model*.

The question as to which flows suffer drops must be consistent (on the average) with the drop rates specified by (4). 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

$$p_{\mathbf{f}^*}(f) = \frac{d_f^{\ell}}{\sum_{\bar{f} \in \mathcal{F}} d_{\bar{f}}^{\ell}} = \frac{s_f^{\ell}}{\sum_{\bar{f} \in \mathcal{F}} s_{\bar{f}}^{\ell}}, \qquad \forall f \in \mathcal{F}. \quad (6)$$

We assume that the flows $\mathbf{f}^*(t_k)$, $\mathbf{f}^*(t_{k+1})$ where drops occur 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 (6) the *drop-selection model*.

We validated the drop-selection model defined by (6) by matching it with `ns-2` [32] simulations. The top plot in Figure 2 shows the outcome of a simulation where 2 TCP flows (RED and BLUE) compete for bandwidth on a bottleneck queue (with 10% on-off CBR traffic). The x-axis shows the fraction of arrival rate for each flow and the y-axis shows the corresponding drop probability. A 45 degree line would be exactly consistent with (6). We can see in the figure that the probabilities of drop measured experimentally match well with the theoretical 45 degree line.

3.2.1 Hybrid model for queue dynamics

The queue model developed above can be compactly expressed by the hybrid automaton represented in Figure 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 in the figure represent transitions between modes. These transitions are labeled with their enabling conditions (that 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., [29]). The inputs to this model are the rates $r_f^{\ell'}$, $f \in \mathcal{F}$ of the upstream queues
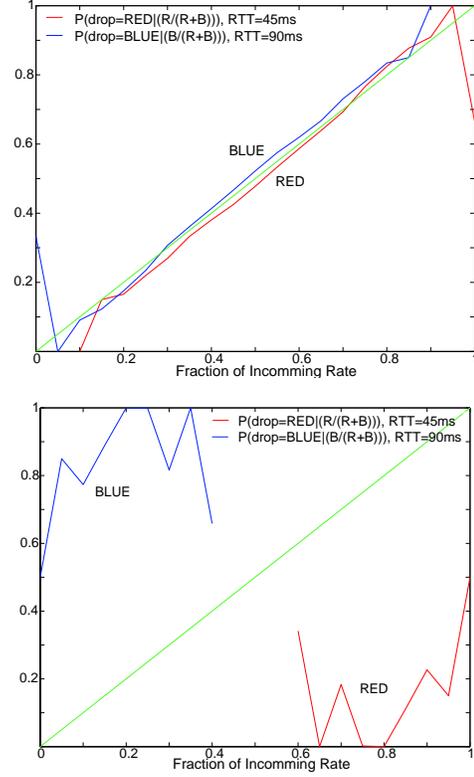
Figure 2: **Drop probability vs. fraction of arrival rate for 10% background traffic (top) and packet synchronization (bottom).**

$\ell' \in \mathcal{L}[\ell]$, which determine 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 round-trip-times.

3.2.2 Other drop models

For completeness one should add that the drop-selection model described by (6) is not universal. For example, in dumbbell topologies without background traffic, one can observe synchronization phenomena that sometimes lead to flows with small sending rates suffering more drops than flows with larger sending rates. The bottom plot in Figure 2 shows an extreme example of this (2 TCP flows in a 5Mbps dumbbell topology with no background traffic and drop-tail queuing). In this example, the BLUE flow gets most of the drops, in spite of using a smaller fraction of the bandwidth. In [26], it was suggested that 10% of random delay would remove synchronization for many TCP connections. We used background traffic because this suggested delay is not sufficient when the number of connections is small. The top plot in Figure 2 shows results obtained with background traffic. In the rest of this section, we briefly discuss another drop model that is useful in specific situations. The reader is referred to [4] for additional models.

Figure contents:

ℓ-queue-empty:
$$\dot{q}_f^\ell = s_f^\ell - r_f^\ell,\ f \in \mathcal{F}$$
$$r_f^\ell = \begin{cases} s_f^\ell & s^\ell \le B^\ell \\ \frac{s_f^\ell}{s^\ell} B^\ell & \text{otherwise} \end{cases}$$

$q^\ell > 0?$

$q^\ell \le 0?$

ℓ-queue-not-full:
$$\dot{q}_f^\ell = s_f^\ell - r_f^\ell,\ f \in \mathcal{F}$$
$$r_f^\ell = \frac{q_f^\ell}{q^\ell} B^\ell$$

$q^\ell = q_{\max}^\ell,\ s^\ell > B^\ell?$

$z^\ell := 0,\ \mathfrak{E}\big[\text{drop in flow } \mathbf{f}^* \sim p_{\mathbf{f}^*}\big]$

$s^\ell \le B^\ell?$

ℓ-queue-full:
$$\dot{z}^\ell = s^\ell - B^\ell$$
$$\dot{q}_f^\ell = (s_f^\ell/s^\ell - q_f^\ell/q^\ell)B^\ell,\ f \in \mathcal{F}$$
$$r_f^\ell = \frac{q_f^\ell}{q^\ell} B^\ell$$

$z^\ell = L?$

$z^\ell := 0,\ \mathfrak{E}\big[\text{drop in flow } \mathbf{f}^* \sim p_{\mathbf{f}^*}\big]$

$r_f^{\ell'},\ f \in \mathcal{F}, \ell' \in \mathcal{L}[\ell]$

$q^\ell,\ r_f^\ell,\ \mathfrak{E}\big[\text{drop in flow } f\big],\ f \in \mathcal{F}$

$dN > 0?$ (with $E[dN] = \gamma(q^\ell)s^\ell dt$)
$\mathfrak{E}\big[\text{drop in flow } \mathbf{f}^* \sim p_{\mathbf{f}^*}\big]$
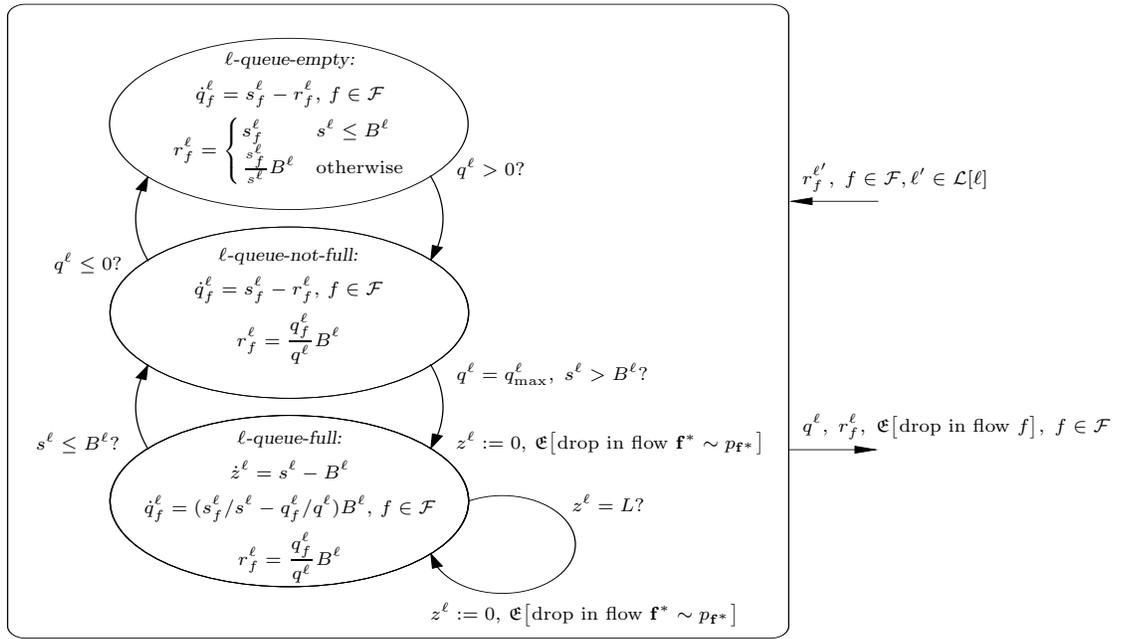
Figure 3: Hybrid model for the queue at link ℓ. In this figure, q^ℓ is given by (2), the s_f^ℓ, $f \in \mathcal{F}$ are given by (3), and $s^\ell := \sum_{f \in \mathcal{F}} s_f^\ell$, $\forall \ell \in \mathcal{L}$.

Drop rotation. The drop model in (6) is not valid when packet synchronization occurs. This effect is particularly noticeable under TCP congestion control (without cross traffic), when all the flows have the same round-trip time and there is a bottleneck link with bandwidth significantly smaller than that of the remaining links. The corresponding bottleneck queue should be operating under drop-tail [3]. A more accurate model for this situation is *drop rotation*. According to 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 round-trip-time after the drop). In the meanwhile, 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 queuing policy, the second packet will almost certainly be dropped. Although the drop-count model (5) would predict the correct number of drops, the drop-selection model (6) would not predict drop rotation because of the independence assumption associated with it.

3.3 TCP model

So far our discussion focused on the modeling of the transmission rates r_f^ℓ and queue sizes q_f^ℓ across the network, taking as inputs the sending rates r_f of the end-to-end flows. In this section, we construct a hybrid model for TCP that should be composed with the flow-conservation law and queue dynamics presented in Sections 3.1 and 3.2, in order to construct a model that describes the overall system. We start by describing the behavior of TCP in each of its main modes, which we later combine into a hybrid model of TCP. We concentrate here on a single flow $f \in \mathcal{F}$.

3.3.1 Slow-start mode

During *slow-start*, the *congestion window* w_f (cwnd) increases exponentially, being multiplied by 2 every round-trip time RTT_f. This can be modeled by

$$\dot{w}_f = \frac{\log m}{RTT_f} w_f, \tag{7}$$

for an appropriately defined constant m. If RTT_f was constant, we would get

$$w_f(t + RTT_f) = e^{\log m \int_t^{t+RTT_f} \frac{1}{RTT_f} d\tau} w_f(t) \approx m w_f(t),$$

Since w_f packets are sent each round-trip time, the instantaneous sending rate r_f should be given by

$$r_f = \frac{w_f}{RTT_f}. \tag{8}$$

However, in this mode the round-trip time RTT_f tends to increase rapidly because of variations on the queue sizes and therefore this formula needs to be corrected to

$$r_f = \frac{\beta w_f}{RTT_f}, \tag{9}$$

where the best match with `ns-2` traces is obtained for $\beta = 1.45$. The formulas (7) and (9) 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 (9) should be replaced by

$$r_f = \frac{\min\{\beta w_f, w_f^{\text{adv}}\}}{RTT_f}. \tag{10}$$

If congestion window reaches the advertised window, *slow-start* mode enters *congestion-avoidance* mode. The *slow-start* mode lasts drop or timeout is 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.

3.3.2 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 round-trip time RTT_f. This can be modeled by

$$\dot{w}_f = \frac{L}{RTT_f},$$

with the instantaneous sending rate r_f given by (8). When the receiver's advertised window size w_f^{adv} is finite, (8) should be replaced by

$$r_f = \frac{\min\{w_f, w_f^{\text{adv}}\}}{RTT_f}.$$

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 the detection of a timeout leads the system to the *timeout* mode.

3.3.3 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 and briefly discuss the differences with respect to TCP-Reno and TCP-NewReno. Due to lack of space we do not provide here the formal model for the later two versions of TCP.

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, and 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 round-trip time after the retransmission (i.e., from t_0 to $t_0 + RTT_f$) 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 3 duplicate acknowledgments that triggered the retransmission), and a single non-duplicate 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 [31]. 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

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

In case a single packet was dropped, fast recovery will finish at $t_0 + RTT_f$, but otherwise it will continue until all the retransmissions take place and are successful. However, from $t_0 + RTT_f$ 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 $t_0 + RTT_f$ to $t_0 + 2RTT_f$ 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

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

On each subsequent interval, the sending rate increases exponentially until all the n_{drop} packets that were dropped are successfully retransmitted. In k round-trip times, the total number of packets retransmitted is equal to

$$\sum_{i=0}^{k-1} 2^i (1 + w_f(t_0^-)/2 - n_{\text{drop}}) =$$
$$= (2^k - 1)(1 + w_f(t_0^-)/2 - n_{\text{drop}}),$$

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

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

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

$$n\big(w_f(t_0^-), n_{\text{drop}}\big) := \left\lceil \log_2 \frac{1 + \frac{w_f(t_0^-)}{2}}{1 + \frac{w_f(t_0^-)}{2} - n_{\text{drop}}} \right\rceil \qquad (11)$$

round-trip times. The previous reasoning is only valid when the number of drops does not exceed $w_f(t_0^-)/2$. As shown in [31], when $n_{\text{drop}} > w_f(t_0^-)/2 + 1$ the sender does not receive enough acknowledgments in the first round-trip time to retransmit any other packets and there is a timeout. When $n_{\text{drop}} = w_f(t_0^-)/2 + 1$ only one packet will be sent on each of the first two round-trip times, followed by exponential increase in the remaining round-trip times. In this case, the fast recovery mode will last approximately

$$n\big(w_f(t_0^-), n_{\text{drop}}\big) := 1 + \lceil \log_2 n_{\text{drop}} \rceil \qquad (12)$$

round-trip times. The behavior of the several variants of TCP in the presence of multiple packet losses in the same window is also discussed in [7].

In `ns-2`, the value of the congestion window variable (cwnd) is actually not changed inside the *fast-recovery* mode. Instead, a variable (`pipe`) is used to emulate the congestion window of standard TCP-Sack algorithm described

above. For compatibility with `ns-2`, in our model we actually keep the congestion window w_f constant throughout the whole duration of fast recovery but adjust the sending rates according to the previous formulas.

TCP-NewReno. TCP-NewReno differs from TCP-Sack in that the sender will only learn about the existence of each additional drop when the retransmission for the previous drop was successful. This means that it must remain in the *fast-recovery* mode for as many round-trip times as the number of drops. Thus, our duration of fast recovery increases linear to the number of dropped packets.

TCP-Reno. In TCP-Reno, the sender leaves the *fast-recovery* mode as soon as the acknowledgment of the first retransmitted packet is received, regardless of the occurrence of more drops. In case more drops occur, these will be detected right after the *fast-recovery* mode and the system re-enters fast recovery again. The net result of each time the *fast-recovery* mode is entered is a division by two of the congestion window size. Thus, three dropped packets in a window often leads to a packet timeout [7].

3.3.4 Timeouts

Timeouts occur when the timeout timer exceeds a threshold that provides a measure of the current round-trip time. 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 any of 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 (10) hold. This lasts until w_f reaches $ssthr_f(t_0)$ or a drop/timeout is detected. The former leads to a transition into the *congestion avoidance* mode, whereas the latter to a transition into the *fast-recovery/ timeout* mode.

3.3.5 Hybrid model for TCP-Sack

The model in Figure 4 combines the modes described in Sections 3.3.1, 3.3.2, 3.3.3, and 3.3.4 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 "round-trip time" from the queue ℓ where the drop occurred, all the way to the receiver, and back to the sender. It can be computed by

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

where $\mathcal{L}[f,\ell]$ denotes the set of links between the ℓ-queue and the sender, passing through the receiver (for drop-tail queuing, 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 exists 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 Figure 4.

The inputs to the TCP-Sack flow model are the round-trip time, the drop events, and the corresponding drop-detection delays (which can be obtained from the flow-conservation law and queue dynamics in Sections 3.1, 3.2) and its outputs are the sending rates of the end-to-end flows.

The model in Figure 4 assumes that the flow f is always active. It is straightforward to turn the flow on and off by adding appropriate modes (similar to what is done in Section 3.4 for UDP flows). In fact, in the simulation results described in Section 4.2 we used random starting times for the persistent TCP flows.

3.4 UDP model

UDP sources differ from TCP sources in that the former do not perform congestion control. The diagram in Figure 5 represents a simple hybrid model for an on-off UDP source with peak rate equal to $r_{\max}$ and exponential distributions for the on and the off times with means τ_{on} and τ_{off}, respectively. The average sending rate for this source is given by $\frac{\tau_{\mathrm{on}} r_{\max}}{\tau_{\mathrm{on}} + \tau_{\mathrm{off}}}$. This model could be generalized to other distributions.
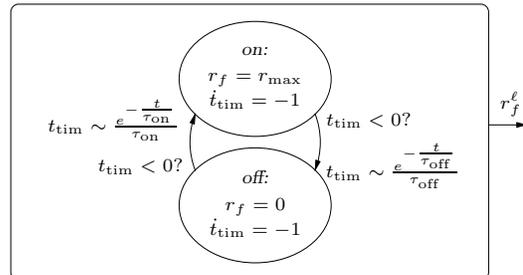
Figure 5: Hybrid model for a UDP flow with exponential on/off-times.

4. VALIDATION

We use the `ns-2` (version 2.1b9a) packet-level simulator to validate our hybrid models. Different network topologies subject to a variety of traffic conditions are considered.
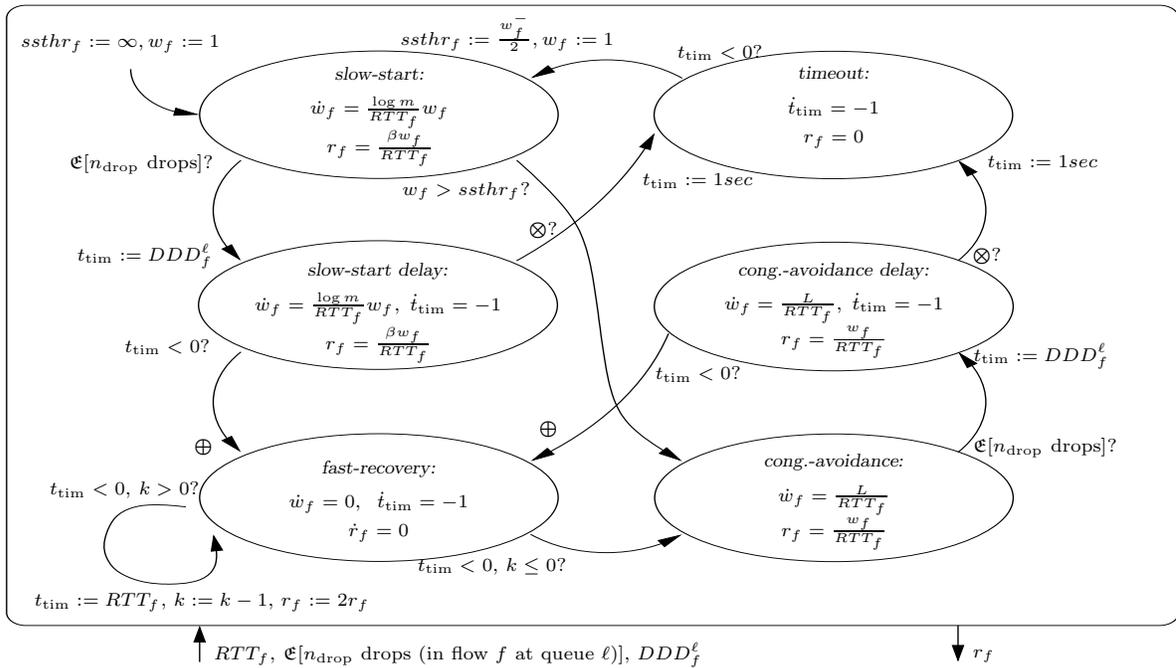
Figure 4: Hybrid model for flow f under TCP-Sack congestion control. The symbol $\otimes$ stands for $w_f \leq \max\{2 + n_{\text{drop}}, 2n_{\text{drop}} - 4\}$ and $\oplus$ stands for $t_{\text{tim}} := RTT_f$, $k := n(w_f^-, n_{\text{drop}})$, $w_f := \frac{w_f^-}{2}$, $r_f = \frac{1 + w_f^-/2 - n_{\text{drop}}}{RTT_f}$, where $n(\cdot)$ is defined by (11)–(12).

4.1 Network Topology

We focus our study on the topologies shown in Figure 6. The topology on the left is known as the *dumbbell topology* and is commonly used to analyze TCP congestion control algorithms. The dumbbell topology is characterized by a set of flows that originate at a source node (node 1) and are directed to a sink node (node 2) through a *bottleneck link*. In more realistic networks, a path with several links (and intermediate nodes) would connect the source and destination. However, to analyze congestion control mechanisms, one often ignores the existence of all the intermediate links, except for the bottleneck, i.e., the most congested link.

While the dumbbell topology only considers TCP flows with uniform propagation delays, the flows in the "Y-shape" topology on the right of Figure 6 exhibit distinct propagation delays: 45ms (Src_1), 90ms (Src_2), 135ms (Src_3), 180ms (Src_4). In our simulations, UDP flows make up for the background traffic injected. Background flows originate from Src_5 and router R2 while TCP flows originate from Src_1 to Src_4. The background traffic model we employ is described in detail in the Section 4.2 below. For the results presented here all queues were chosen to be 40 packets long.

4.2 Simulation Environment

All `ns-2` simulations use TCP-Sack (more specifically its Sack1 variant outlined in [7]). Each simulation ran for 600 seconds of simulation time and data points were obtained by averaging out 20 trials. TCP flows start randomly between 0 and 2 seconds. We model background traffic as (UDP) on/off CBR flows which turn on/off after being off/on for an exponentially distributed amount of time. We considered different amounts of background traffic in the form of short-lived flows whose on and off time is 0.5 seconds on average. In particular, the results presented in this paper were obtained by injecting 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 [11]. We should point out that the quality of the hybrid system simulations do not degrade as more short-live traffic is considered. As previously mentioned, the drop model is topology dependent. As observed in [3], 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 metrics used for comparing the hybrid system and packet-level models include throughput, round trip time, loss rate, and congestion window size for the TCP flows. We also measure queue size at the bottleneck link.

4.3 Results

Figure 7 compares simulation results for the dumbbell topology with a single TCP flow (no background traffic) obtained with `ns-2` and our hybrid model. These plots show TCP's congestion window size and bottleneck queue size over time. As discussed in Section 3.2.2, we use drop rotation to model drops in dumbbell topologies. The plots show a nearly perfect match. While most existing models of TCP congestion control 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 developed our model to capture the basic slow-start behavior of TCP Sack1: When more
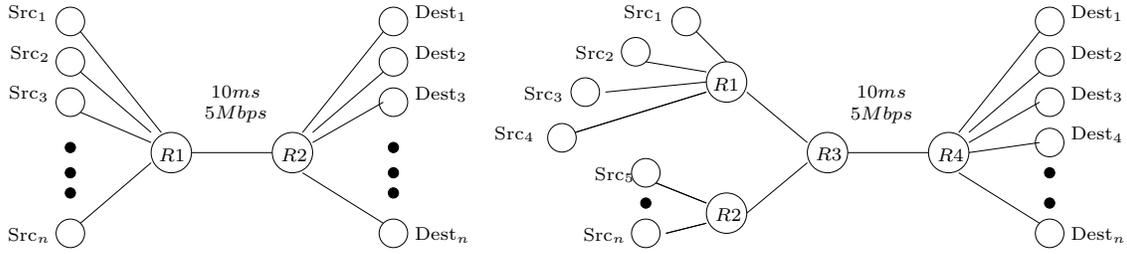
Figure 6: Dumbbell (left) and Y-shape multi-queue topology with 4 different propagation delays (right).
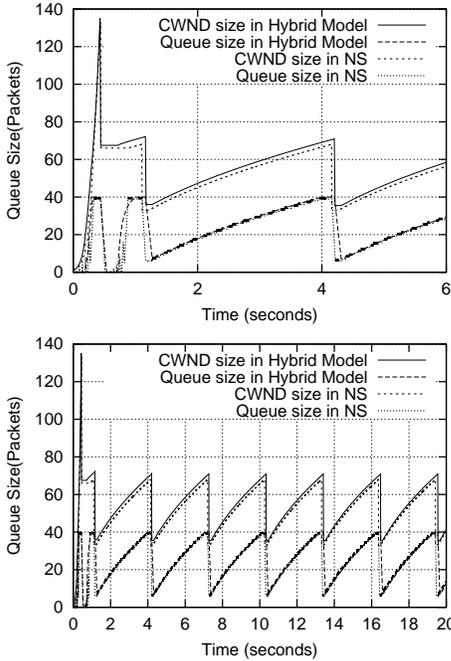
Figure 7: Comparison of the congestion window and queue sizes over time for the dumbbell topology with one TCP flow and no background traffic.
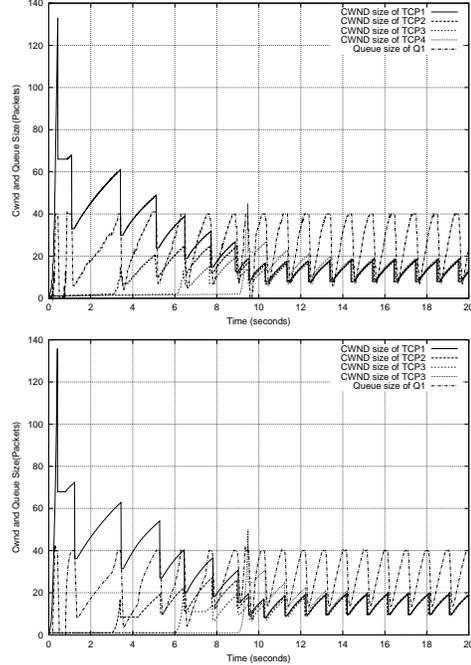
Figure 8: Congestion window and queue size over time for the dumbbell topology with 4 TCP flows and no background traffic. Simulations using `ns-2` (top) and a hybrid model (bottom).

than cwnd/2+1 packets are lost, a timeout occurs because there are not enough acknowledgments to open the congestion window [31]; and when the number of losses is around cwnd/2, Sack1 eventually leaves fast-recovery but only after a few multiples of the round-trip time (cf. Section 3.3.3). This is consistent with Figure 7, where we see that, after the initial drops, the congestion window is divided by two and maintains this value for about half a second before it begins increasing linearly.

In the next set of experiments, we simulate 4 TCP flows on the dumbbell topology with and without background traffic. Figure 8 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 [34, 26]. This synchronization is modeled using drop rotation. Similarly 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 around 1 second.

Simulation results for 4 TCP flows with background traffic are summarized in Table 1, which presents average throughput and round-trip time for each flow for both hybrid system and `ns-2` simulations. These statistics confirm that the hybrid model nearly reproduces the results obtained from the packet-level model.

To validate our hybrid models, we also use the Y-shape, multi-queue topology with different round-trip-times shown on the right-hand side of Figure 6. As discussed in Section 3.2, we consider the drop-count and drop-selection models described by Equations (5) and (6), respectively. Unlike in the drop rotation model in which losses are deterministic, (6) generates stochastic drops. Since losses are random, no two simulations will be exactly the same so it is not possible for the hybrid model to exactly reproduce the results from `ns-2`. Table 2 summarizes simulation results obtained with `ns-2` and the hybrid model for 4 TCP flows with 10% background traffic on the Y-shape topology under drop tail discipline. This table presents the mean throughput and

	Thru$_1$	Thru$_2$	Thru$_3$	Thru$_4$	RTT$_1$	RTT$_2$	RTT$_3$	RTT$_4$
`ns-2`	1.14	1.13	0.13	1.12	0.094	0.094	0.094	0.094
hybrid system	1.14	1.15	1.15	1.15	0.096	0.096	0.096	0.096
relative error	0%	0.01%	0.01%	0.02%	0.02%	0.02%	0.02%	0.02%

Table 1: Average throughput and round-trip time for the dumbbell topology with 4 TCP flows and 10% background traffic.

mean round-trip time for each competing TCP flow. Note that the relative error is less than 10% and in most cases well under 10%. Similar results hold for variations of the Y-shape topology, e.g., different round-trip times, number of competing flows. However, for the stochastic drop model to hold, there must be either 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 should be employed. As described in Section 3.2.2, in single bottleneck topologies drop-rotation provides an accurate model. However, in more complex settings, deriving the deterministic drop model is quite challenging. This is one direction of future work we plan to pursue.

A more accurate methodology to compare stochastic processes is to examine their probability density function. Figure 9 plots the probability density functions corresponding to the time-series used to generate the results in Table 2. We observe that the hybrid model can reproduce similar probability densities. Regarding the density function of the congestion window, three of the flows closely agree, while one shows a slight bias towards the larger value. The density function of the queue is similar for both models. One noticeable discrepancy is that the peak near the queue-full state is sharper in the case of the hybrid model. This difference is due to the fact that the queue in `ns-2` can only take integer values, while the queue in the hybrid model 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 probability mass at $q_{max} - 1$ in `ns-2`. This results in a more smeared probability mass around queue-full in the case of `ns-2`.

While visually comparing two density functions provides a general understanding of their similarity, there are several quantitative metrics to compare density functions. One good metric is the L^1-distance [6], which has the desirable property that when f and g are densities,

$$\int |f - g| = 2\sup_A \left| \int_A f - \int_A g \right|.$$

Thus, if the probability of an event A is to be predicted, the error of the prediction is half of the L^1-distance between the density functions. Table 3 shows the L^1-distance between the hybrid model and `ns-2` histograms. The distances indicate that the hybrid model provides a good prediction of the actual probability density function.

5. COMPUTATIONAL COMPLEXITY

While hybrid systems allow for a theoretical understanding of networks, they are also amenable to simulation.

In [18] it was shown that a *ripple effect* may greatly degrade the efficiency of fluid models. Since a hybrid model is to some extent a fluid model, one might expect an increase
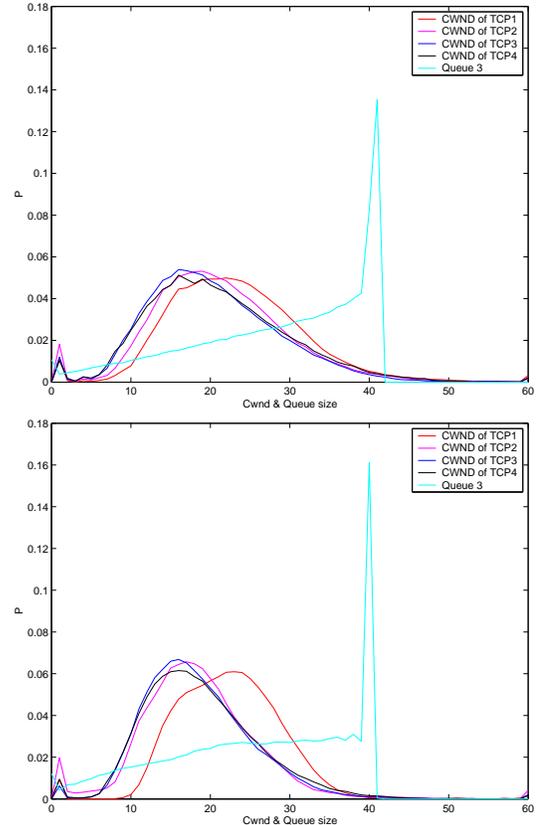
Figure 9: Probability density functions for the congestion window and queue size for the Y-shape topology with 4 TCP flows and 10% background traffic. These were computed from simulations using `ns-2` (top) and a hybrid model (bottom).

in computation speed when compared to a packet level simulator such as `ns-2`. However, in [18] the flow rates are held constant between events, whereas here we utilize ordinary differential equations (ODEs) to describe the bit-rates, accommodating complex variations in the sending rates between events.

Modern ODE solvers are especially efficient while the continuous variables are well modeled by polynomials. However, in networks the bit-rates have occasional discontinuities. This requires special care and can lead to significant computational burden. In fact, the simulation time grows essentially linearly with the number of discontinuities in the continuous variables. In our models, these discontinuities are essentially caused by two types of discrete events: drops and queues emptying. Drops typically cause TCP to abruptly decrease the congestion window, whereas a queue

	Thru$_1$	Thru$_2$	Thru$_3$	Thru$_4$	RTT$_1$	RTT$_2$	RTT$_3$	RTT$_4$
`ns-2`	1.873	1.184	0.836	0.673	0.0969	0.141	0.184	0.227
hybrid model	1.824	1.091	0.823	0.669	0.0879	0.132	0.180	0.223
relative error	2.6%	7.9%	1.5%	0.7%	9.3%	5.9%	3.6%	2.1%

Table 2: Average throughput and round-trip time for the Y-shape topology with 4 TCP flows and 10% background traffic.

	cwnd1	cwnd2	cwnd3	cwnd4	bottleneck queue
4 flows dumbbell with background traffic	0.0071	0.0067	0.0071	0.0066	0.0108
4 flows y-shape with background traffic	0.0034	0.0044	0.0025	0.0033	0.0054

Table 3: L^1-distance between histograms computed from simulations using `ns-2` and a hybrid model.

becomes empty forces the outgoing bit-rates to switch from a fraction of the outgoing link bandwidth to the incoming bit-rates[1]. It turns out that 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. These issues are illustrated in the comparison in Table 4 between `ns-2` and an hybrid model. This table shows execution time (in seconds) of `ns-2` and the hybrid simulator Modelica [23], where a single bottleneck topology with 20ms propagation delay was utilized. The bandwidth of the bottleneck link is 5Mbps, 50Mbps or 500Mbps as shown. There were either one or three long-lived TCP flows competing for the bottleneck bandwidth. Each simulation ran for 10 minutes of simulations time. The table displays the number of seconds required to complete the simulation on a 1.2GHz Pentium PC with 512MB memory.

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 $T = \frac{c}{RTT\sqrt{p}}$, which relates the per-flow throughput T, the average round-trip time RTT, and the drop probability p for dumbbell topology [24]. According to this formula the total drop-rate for n competing flows, which is equal to nTp, is given by $\frac{n}{T}\frac{c^2}{RTT^2}$. 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, of order $O(nT)$. This is confirmed by the results in Table 4, which show that the hybrid simulator is especially competitive for large per-flow throughputs. However, when many flows share a low-bandwidth link packet-level simulators have the advantage. E.g., when 3 flows share a 5Mbps link `ns-2` is only twice as slow and could actually be faster than our hybrid simulator if we increased the number of flows.

Memory usage is also a concern when simulating networks. Hybrid systems requires one state variable for each active flow and one state variable for each flow passing through a queue. Hence, memory usage scales linearly with the number of flows and 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.

6. CONCLUSION AND FUTURE WORK

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 very short time scale. 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 round-trip times. This is quite appropriate for the analysis and design of network protocols including congestion control mechanisms.

One direction for future research is to perform very large scale simulation. We found that the hybrid model accurately matches discrete packet level simulation for small simulations. However, the real benefit of this approach will become apparent after a study of very large scale simulations is complete.

7. REFERENCES

[1] J. S. Ahn and P. B. Danzig. Packet network simulation: speedup and accuracy versus timing granularity. *IEEE/ACM Trans. on Networking*, 4(5):743–757, Oct. 1996.

[2] S. Bohacek. Fair pricing of video transmissions using best-effort and purchased bandwidth. In *Proc. of the 40th Annual Allerton Conf. on Comm., Contr., and Computing*, 2002.

[3] S. Bohacek, J. P. Hespanha, J. Lee, and K. Obraczka. Analysis of a TCP hybrid model. In *Proc. of the 39th Annual Allerton Conf. on Comm., Contr., and Computing*, Oct. 2001.

[4] S. Bohacek, J. P. Hespanha, J. Lee, and K. Obraczka. A hybrid systems modeling framework for fast and accurate simulation of data communication networks: Extended version. Technical report, University of California, Santa Barbara, Nov. 2002.

[5] N. Cardwell, S. Savage, and T. Anderson. Modeling TCP latency. In *Proc. of the IEEE INFOCOM*, Tel-Aviv, Israel, Mar. 2000.

[6] L. Devroye. *A Course in Density Estimation*. Birkhauser, Boston, 1987.

---

[1]Neither of these events exhibits the type of explosion described in [18] because none of them instantaneously generates further events of the same type downstream. Actually, a queue emptying can eventually cause other queues to empty downstream but not before some time has elapsed.

	1 TCP (5M)	3 TCPs (5M)	1 TCP (50M)	3 TCPs (50M)	1 TCP (500M)	3 TCPs (500M)
Hybrid model	0.37	2.78	0.15	0.511	.87	1.39
ns-2	5.4	5.5	107	107	978	1094

Table 4: Execution time (in seconds) of `ns-2` and Modelica for 10 minutes of simulation time.

[7] K. Fall and S. Floyd. Simulation-based comparisons of Tahoe Reno and SACK TCP. *ACM Comput. Comm. Review*, 27(3):5–21, July 1996.

[8] S. Floyd. Connections with multiple congested gateways in packet-switched networks part1: One-way traffic. *ACM Comput. Comm. Review*, 21(5):30–47, Oct. 1991.

[9] S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation-based congestion control for unicast applications. In *Proc. of the ACM SIGCOMM*, pages 43–56, Aug. 2000.

[10] Y. Guo, W. Gong, and D. Towsley. Time-stepped hybrid simulation (TSHS) for large scale networks. In *Proc. of the IEEE INFOCOM*, Mar. 2000.

[11] F. Hernández-Campos, J. S. Marron, G. Samorodnitsky, and F. D. Smith. Variable heavy tail duration in Internet traffic,. In *Proc. of IEEE/ACM MASCOTS 2002*, 2002.

[12] H. Hisamatsu, H. Ohsaki, , and M. Murata. On modeling feedback congestion control mechanism of TCP using fluid flow approximation and queuing theory. In *Proc. of 4th Asia-Pacific Symp. on Information and Telecommunication Technologies*, 2001.

[13] P. Huang, D. Estrin, and J. Heidemann. Enabling large-scale simulations: selective abstraction approach to the study of multicast protocols. In *Proc. of the Int. Symp. on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, pages 241–248. IEEE, July 1998.

[14] G. Kesidis, A. Singh, D. Cheung, and W. Kwok. Feasibility of fluid-event-driven simulation for ATM networks. In *Proc. IEEE GLOBECOM*, volume 3, pages 2013–2017, Nov. 1996.

[15] K. Kumaran and D. Mitra. Performance and fluid simulations of a novel shared buffer management system. In *Proc. of the IEEE INFOCOM*, pages 1449–1461, Mar. 1998.

[16] S. Kunniyur and R. Srikant. Analysis and design of an adaptive virtual queue (AVQ) algorithm for active queue management. In *Proc. of the ACM SIGCOMM*, San Diego, California, USA, 8 2001.

[17] T. V. Lakshman, U. Madhow, and B. Suter. Window-based error recovery and flow control with a slow acknowledgment channel: A study of TCP/IP performance. In *Proc. of the IEEE INFOCOM*, Apr. 1997.

[18] 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. of the IEEE INFOCOM*, volume 3, pages 1244–1253, Apr. 2001.

[19] S. H. Low, F. Paganini, J. Wang, S. Adlakha, and J. C. Doyle. Dynamics of TCP/RED and a scalable control. In *Proc. of the IEEE INFOCOM*, June 2002.

[20] M. Mathis, J. Semke, J. Mahdavi, and T. Ott. The macroscopic behavior of the TCP congestion avoidance algorithm. *ACM Comput. Comm. Review*, 27(3), July 1997.

[21] V. Misra, W. Gong, and D. Towsley. Stochastic differential equation modeling and analysis of TCP-windowsize behavior. In *Proc. of PERFORMANCE'99*, Istanbul, Turkey, 1999.

[22] V. Misra, W. Gong, and D. Towsley. Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED. In *Proc. of the ACM SIGCOMM*, Sept. 2000.

[23] Modelica Association. *Modelica — A Unified Object-Oriented Language for Physical Systems Modeling: Tutorial*. Available at http://www.modelica.org/.

[24] T. Ott, J. H. B. Kemperman, and M. Mathis. Window size behavior in TCP/IP with constant loss probability. In *Proc. of the DIMACS Workshop on Performance of Realtime Applications on the Internet*, Nov. 1996.

[25] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP throughput: a simple model and its empirical validation. In *Proc. of the ACM SIGCOMM*, Sept. 1998.

[26] L. Qiu, Y. Zhang, and S. Keshav. Understanding the performance of many TCP flows. *Computer Networks*, 37:277–306, Nov. 2001.

[27] QualNet user manual. Available at *http://www.scalable-networks.com*.

[28] Scalable simulation framework. Available at *http://www.ssfnet.org/*.

[29] A. V. D. Schaft and H. Schumacher. *An Introduction to Hybrid Dynamical Systems*. Number 251 in Lect. Notes in Contr. and Inform. Sci. Springer-Verlag, London, 2000.

[30] D. Schwetman. Hybrid simulation models of computer systems. *Communications of the ACM*, 19:718–723, Sept. 1978.

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

[32] The VINT Project, a collaboration between researchers at UC Berkeley, LBL, USC/ISI, and Xerox PARC. *The ns Manual (formerly ns Notes and Documentation)*, Oct. 2000. Available at *http://www.isi.edu/nsnam/ns/ns-documentation.html*.

[33] A. Yan and W. Gong. Fluid simulation for high speed networks. *IEEE Trans. on Inform. Theory*, June 1999.

[34] L. Zhang, S. Shenker, and D. D. Clark. Observations on the dynamics of a congestion control algorithm: The effects of two-way traffic. In *Proc. of the ACM SIGCOMM*, Sept. 1991.