Hybrid Control and Switched Systems

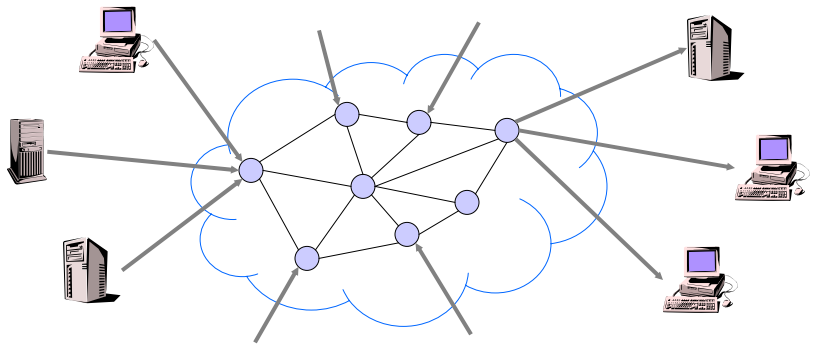
Lecture #17
Hybrid Systems Modeling of Communication Networks

João P. Hespanha

University of California
at Santa Barbara

UCSB

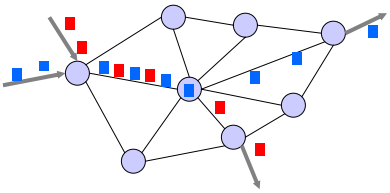
Motivation



Why model network traffic?

- to validate designs through simulation (scalability, performance)
- to analyze and design protocols (throughput, fairness, security, etc.)
- to tune network parameters (queue sizes, bandwidths, etc.)

Types of models

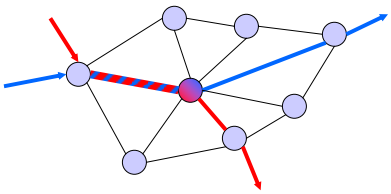


Packet-level modeling

- tracks individual data packets, as they travel across the network
- ignores the data content of individual packets
- sub-millisecond time accuracy
- computationally very intensive

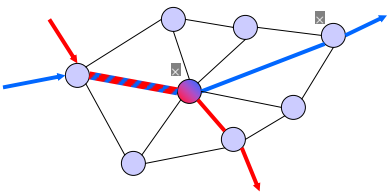
Fluid-based modeling

- tracks time/ensemble-average packet rates across the network
- does not explicitly model individual events (acknowledgments, drops, queues becoming empty, etc.)
- time accuracy of a few seconds for time-average
- only suitable to model many similar flows for ensemble-average
- computationally very efficient (at least for first order statistics)



Types of models

captures fast dynamics
even for a small number of flow



Hybrid modeling

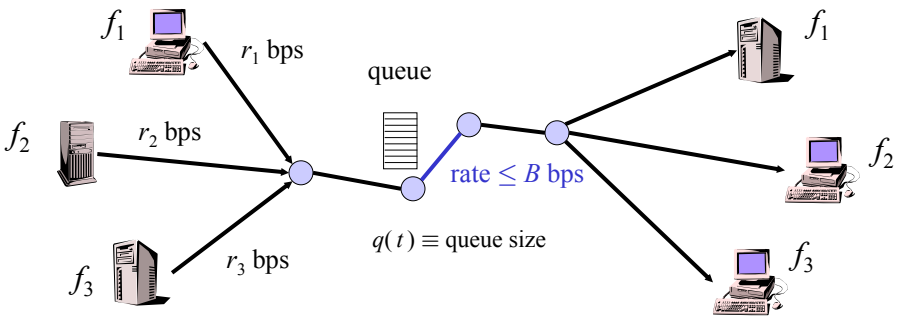
- keeps track of packet rates for each flow averaged over small time scales
- explicitly models some discrete events (drops, queues becoming empty, etc.)
- time accuracy of a few milliseconds (round-trip time)
- computationally efficient

provide information about both
average, peak, and "instantaneous"
resource utilization
(queues, bandwidth, etc.)

Summary

- Modeling 1st pass: Dumbbell topology & simplified TCP
- Modeling 2nd pass: General topology, TCP and UDP models
- Validation
- Simulation complexity

1st pass – Dumbbell topology



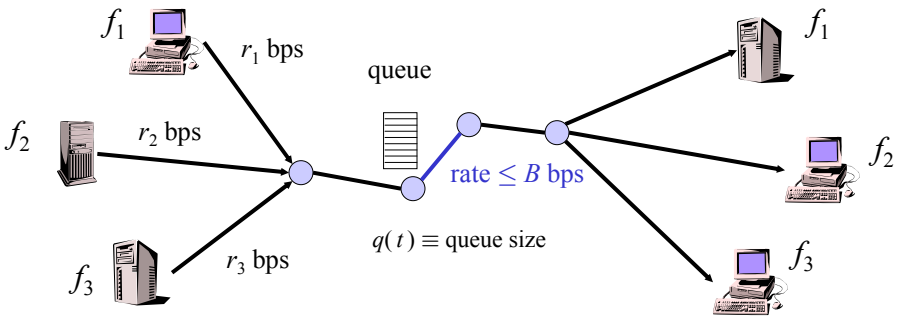
Several flows follow the same path and compete for bandwidth in a single bottleneck link

Prototypical network to study congestion control

routing is trivial — single queue

B is unknown to the data sources and possibly time-varying

Queue dynamics



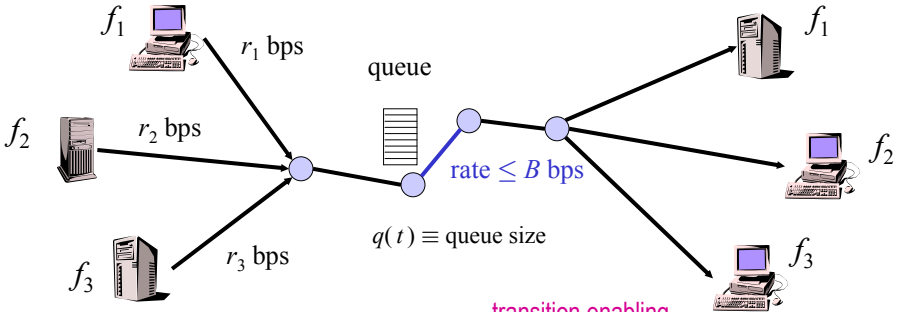
When $\sum_f r_f$ exceeds B the queue fills and data is lost (drops)

$$\dot{q} = \begin{cases} \sum_f r_f - B & 0 \leq q \leq q_{\max} \\ 0 & \text{otherwise} \end{cases}$$

$$q = q_{\max}, \quad \sum_f r_f > B \quad \Rightarrow \textit{drop}$$

(discrete event – relevant for congestion control)

Queue dynamics



transition enabling condition

Hybrid automaton representation:

$$\dot{q} = \begin{cases} \sum_f r_f - B & 0 \leq q \leq q_{\max} \\ 0 & \text{otherwise} \end{cases}$$

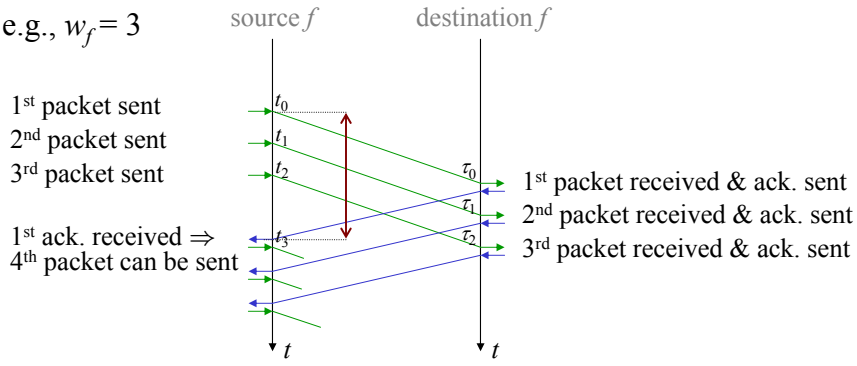
$$q = q_{\max}, \ \sum_f r_f > B \ ?$$

$\mathfrak{E}$[drop] ⟵ exported discrete event

Window-based rate adjustment

w_f (window size) ≡ number of packets that can remain unacknowledged for by the destination

e.g., $w_f = 3$



w_f effectively determines the sending rate r_f:

$$r_f(t) = \frac{w_f(t)}{RTT(t)}$$

round-trip time

Window-based rate adjustment

w_f (window size) ≡ number of packets that can remain unacknowledged for by the destination

$r_f(t) = \dfrac{w_f(t)}{RTT(t)}$ ≡ sending rate

per-packet transmission time

$$RTT(t) = T_{\mathrm{p}} + \frac{1}{B}q(t)$$

total round-trip time

propagation delay

time in queue until transmission

queue gets full → longer *RTT* → rate decreases → queue gets empty

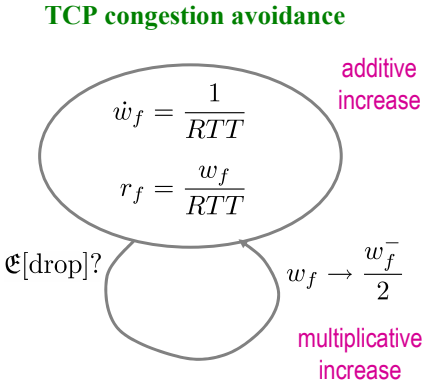
negative feedback

This mechanism is still not sufficient to prevent a catastrophic collapse of the network if the sources set the w_f too large

TCP congestion avoidance

1. While there are no drops, increase w_f by 1 on each RTT (additive increase)
2. When a drop occurs, divide w_f by 2 (multiplicative decrease)

(congestion controller constantly probes the network for more bandwidth)

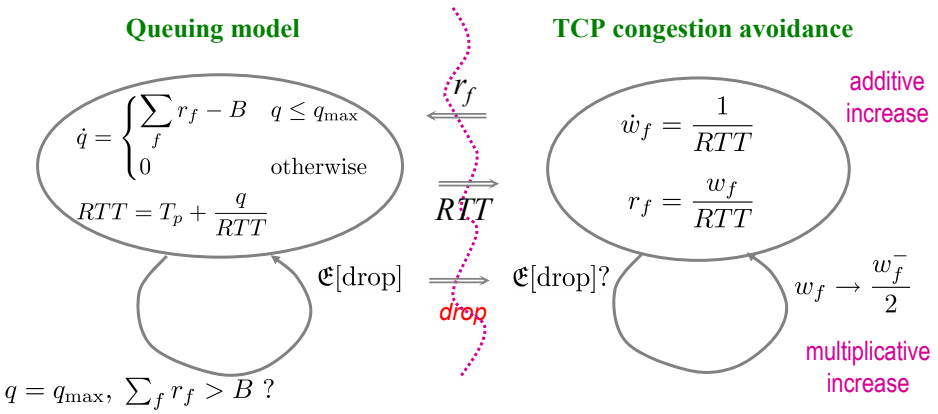


disclaimer: this is a very simplified version of TCP Reno, better models later…

TCP congestion avoidance

1. While there are no drops, increase w_f by 1 on each RTT (additive increase)
2. When a drop occurs, divide w_f by 2 (multiplicative decrease)

(congestion controller constantly probes the network for more bandwidth)



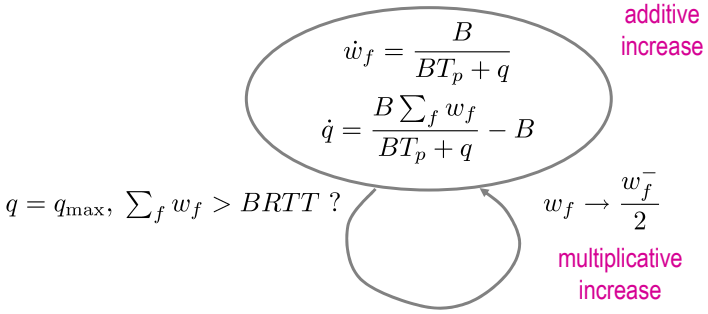
disclaimer: this is a very simplified version of TCP Reno, better models later…

TCP congestion avoidance

1. While there are no drops, increase w_f by 1 on each RTT (additive increase)
2. When a drop occurs, divide w_f by 2 (multiplicative decrease)

(congestion controller constantly probes the network for more bandwidth)

TCP + Queuing model



disclaimer: this is a very simplified version of TCP Reno, better models later…

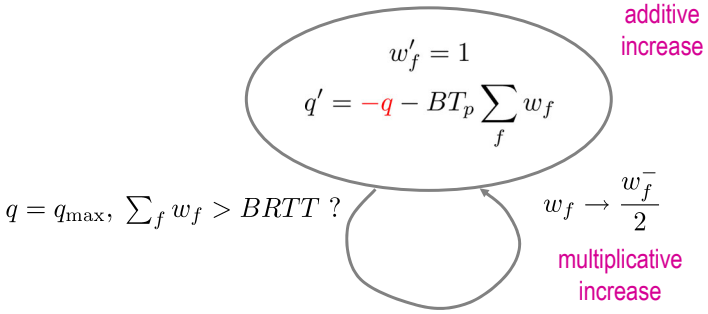
Linearization of the TCP model

Time normalization ≡ define a new "time" variable τ by

$$\frac{\mathrm{d}\tau}{\mathrm{d}t} = \frac{1}{RTT} = \frac{B}{BT_p + q}, \qquad \tau(0) = 0$$

1 unit of τ ≡ 1 round-trip time

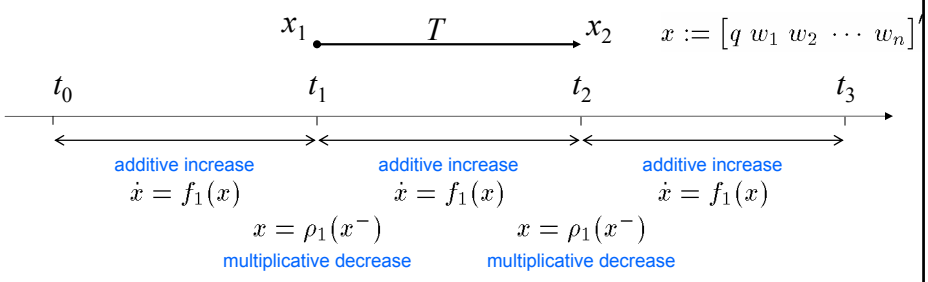
TCP + Queuing model



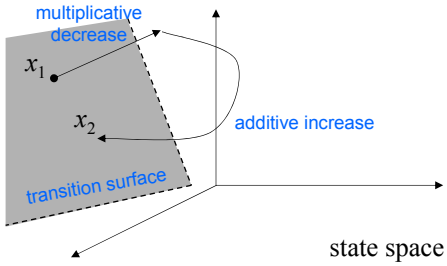
$$(\cdot)' := \frac{\mathrm{d}}{\mathrm{d}\tau}$$

In normalized time, the continuous dynamics become **linear**

Switching-by-switching analysis



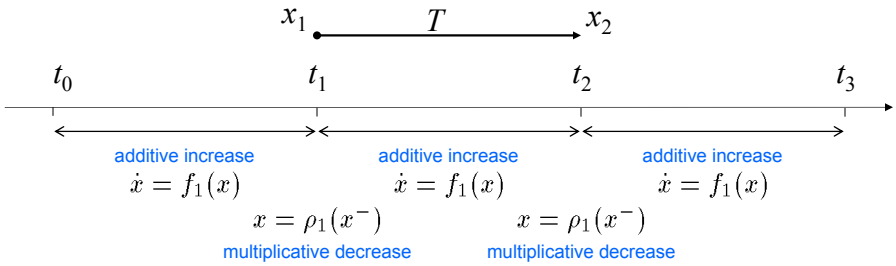
$x_k := x(t_k)$ ≡ continuous state before the k^{th} multiplicative decrease



$$x_{k+1} = T(x_k)$$

impact map

Switching-by-switching analysis



$x_k := x(t_k)$ ≡ continuous state before the k^{th} multiplicative decrease

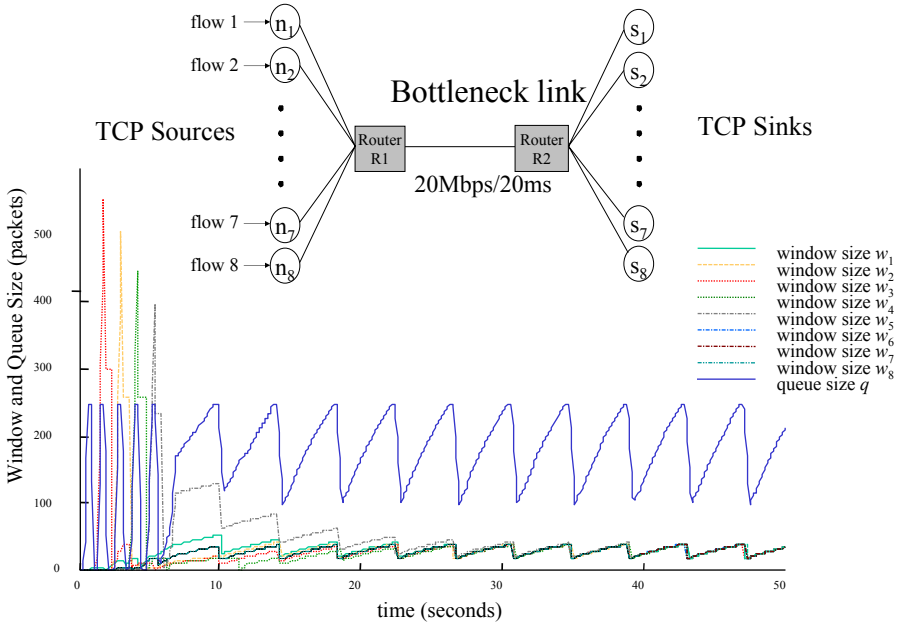
Theorem. The function T is a contraction. In particular,

$$||T(a) - T(b)||_* \leq \frac{1}{2}||a - b||_*, \quad \forall a, b$$

Therefore

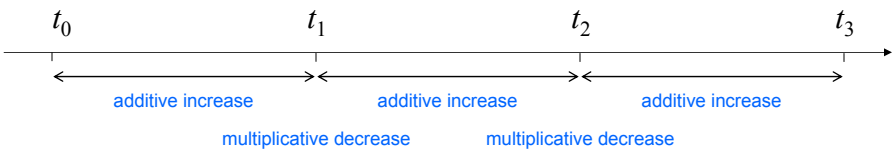
- $x_k \to x_\infty$ as $k \to \infty$ — $x_\infty \equiv$ constant
- $x(t) \to x_\infty(t)$ as $t \to \infty$ — $x_\infty(t) \equiv$ periodic limit cycle

NS-2 simulation results



Results

(valid for $n \ll \frac{q_{\max}+BT_p}{2}$)



Window synchronization:

$$w_f(t_k) \rightsquigarrow \frac{q_{\max} + BT_p}{n}, \quad \text{as } k \to \infty \qquad \forall f$$

convergence is exponential, as fast as $.5^k$

Steady-state formulas:

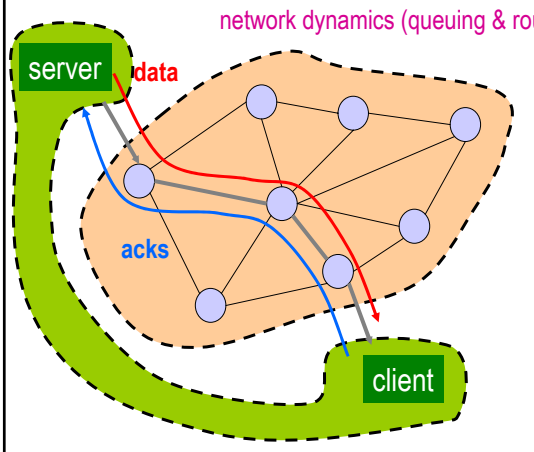
average drop rate $$p \approx 2.67 \left(\frac{n}{q_{\max} + BT_p} \right)^2$$

average RTT $$\overline{RTT} \approx .78 \left(T_p + \frac{q_{\max}}{B} \right)$$

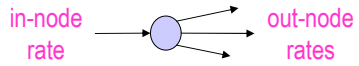
average throughput $$T \approx \frac{1.27}{\overline{RTT}\sqrt{p}}$$ (well known TCP-friendly formula)

2nd pass – general topology

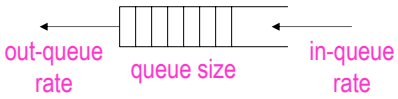
A communication network can be viewed as the interconnection of several blocks with specific dynamics



a) Routing:



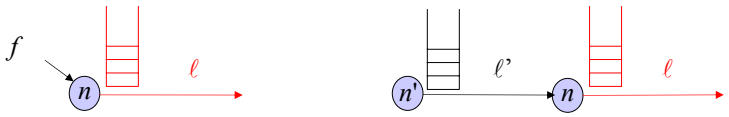
b) Queuing:



c) End2end cong. control

Routing

determines the sequence of links followed by each flow



Conservation of flows:

end2end sending rate of flow f

$$s_f^\ell = \begin{cases} r_f & \text{flow } f \text{ starts in this node} \\ r_f^{\ell'} & \text{flow } f \text{ comes from link } \ell' \end{cases}$$

in-queue rate of flow f

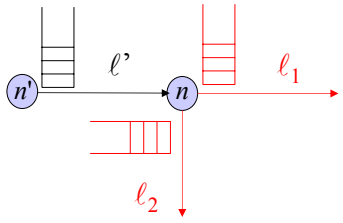
upstream out-queue rate of flow f

indexes ℓ and ℓ' determined by routing tables

Routing

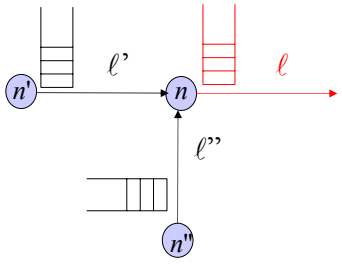
determines the sequence of links followed by each flow

Multicast



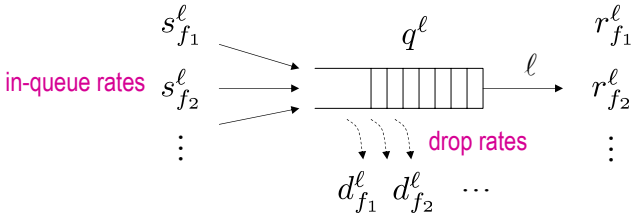
$$s_f^{\ell_1} = s_f^{\ell_2} = r_f^{\ell'}$$

Multi-path routing



$$s_f^{\ell} = r_f^{\ell'} + r_f^{\ell''}$$

Queue dynamics



$$\sum_f r_f^{\ell} \le B^{\ell}$$

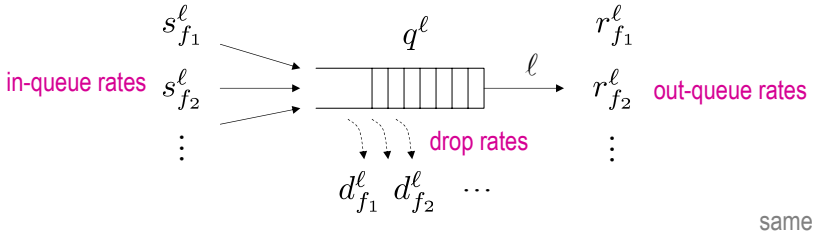
link bandwidth

total queue size → $q^{\ell} = \sum_{f \in \mathcal{F}} q_f^{\ell}$ ← queue size due to flow f

the packets of each flow are assumed uniformly distributed in the queue

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

Queue dynamics



same in and out-queue rates

no drops

$d^\ell_f = 0$

queue empty

$q^\ell = 0$

$(\sum_f s^\ell_f \le B^\ell)$

$r^\ell_f = s^\ell_f$

queue not empty/full

$q^\ell \in (0, q^\ell_{\max})$

queue full

$q^\ell = q^\ell_{\max}$

$(\sum_f s^\ell_f > B^\ell)$

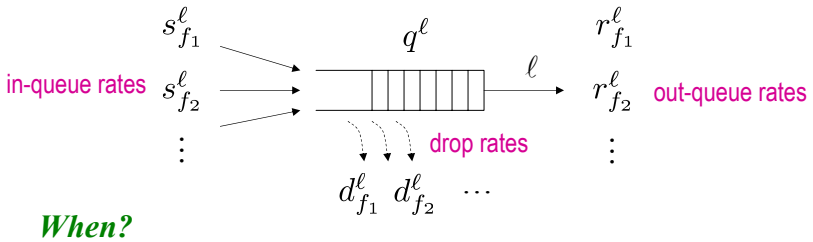
$r^\ell_f = \frac{q^\ell_f}{q^\ell} B^\ell$

out-queue rates proportional to fraction of packets in the queue

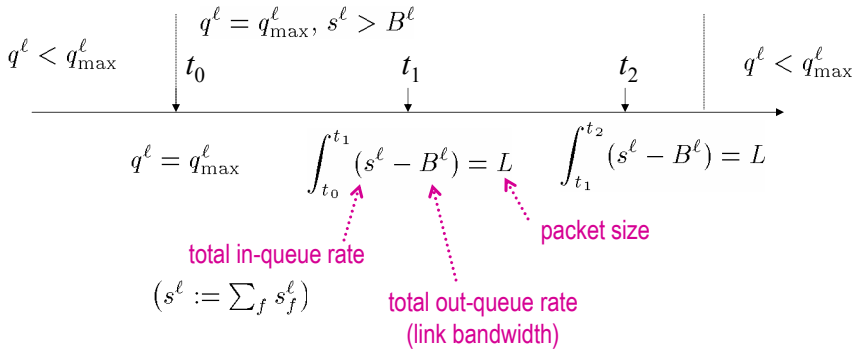
$d^\ell_f = \frac{s^\ell_f}{q^\ell}\left(\sum_{f\in\mathcal{F}} s^\ell_f - B^\ell\right)$

drops proportional to fraction in-queue rates

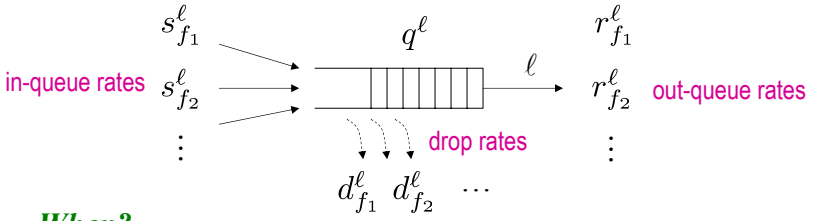
Drops events



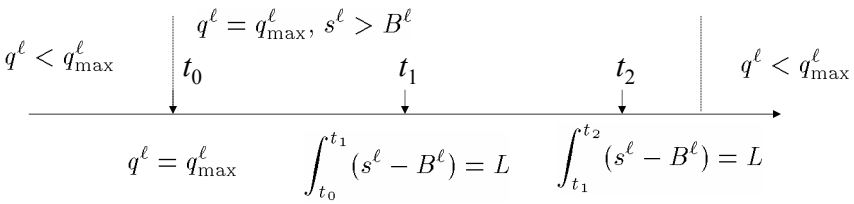
When?



Drops events



When?



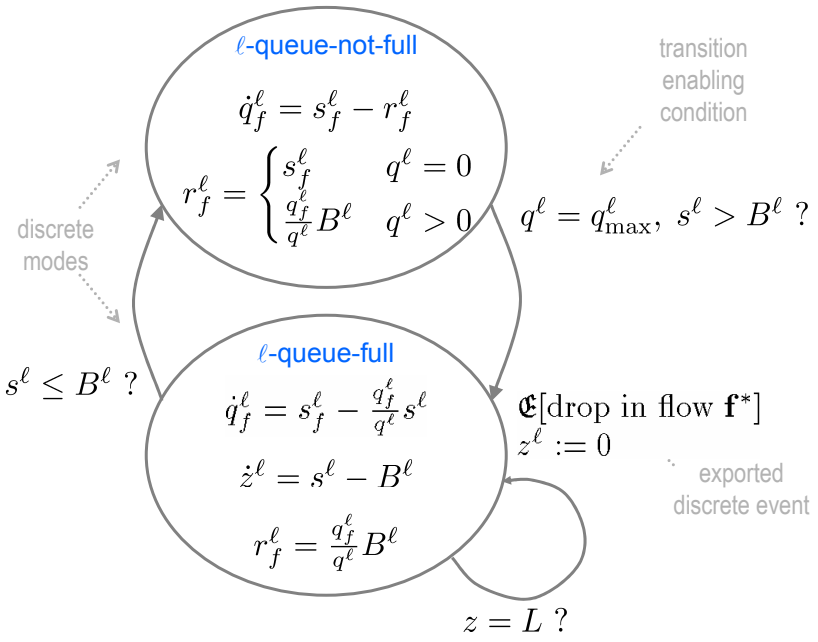
Which flows?

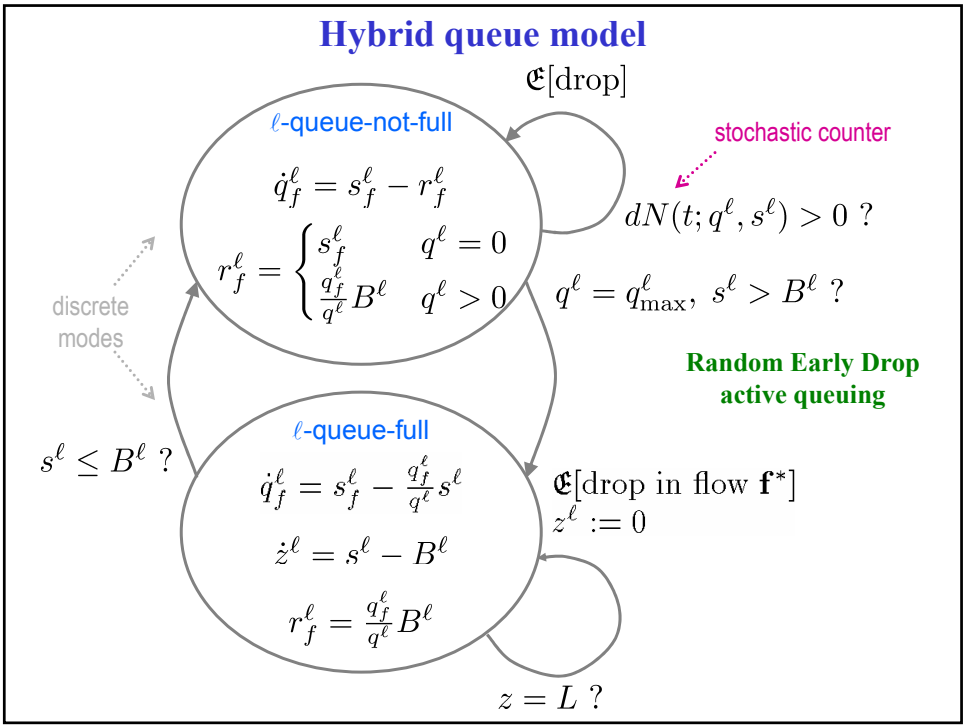
$$\mathrm{P}(\mathbf{f}^*(t_k) = f) = \frac{s^\ell_f}{s^\ell}, \quad f \in \mathcal{F}$$

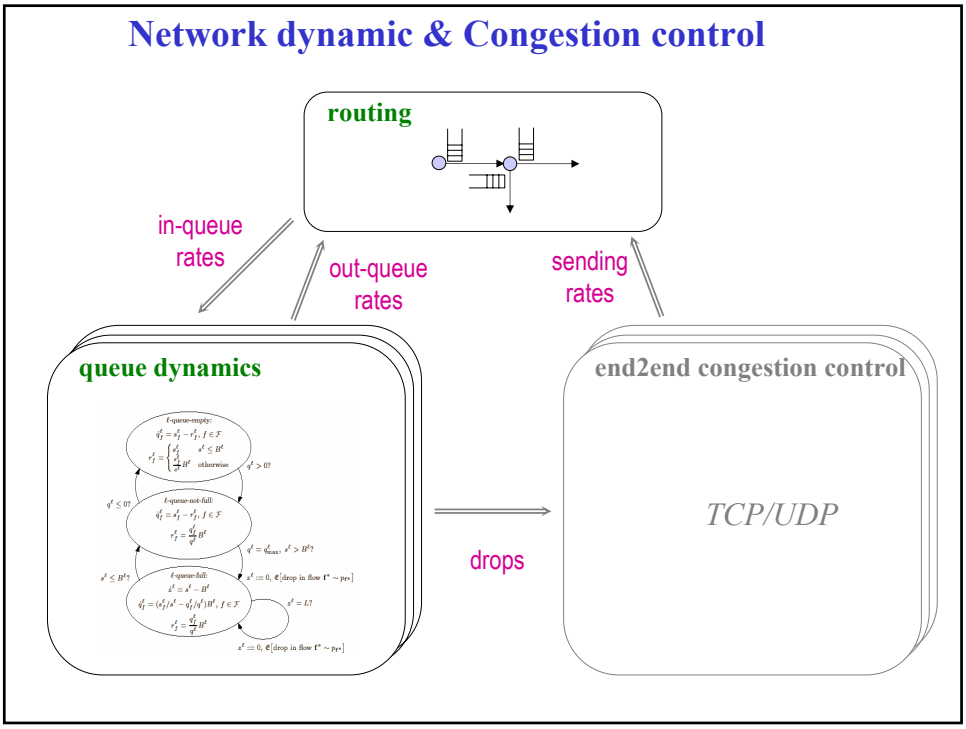
(drop tail dropping)

flow that suffers drop at time t_k

Hybrid queue model

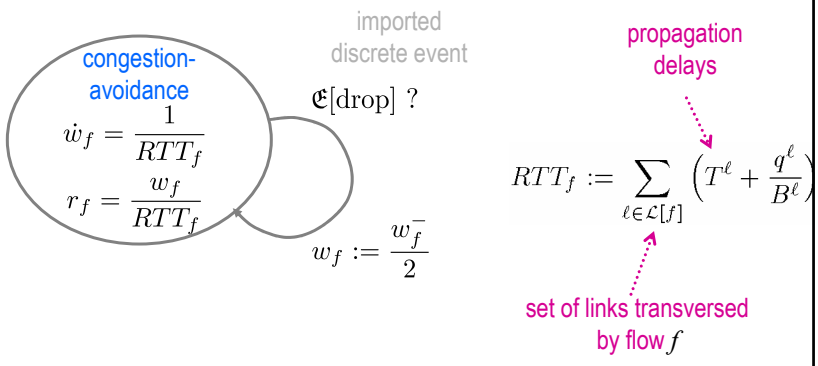



Hybrid queue model
$\mathfrak{E}$[drop]
ℓ-queue-not-full
$\dot{q}_f^\ell = s_f^\ell - r_f^\ell$
$r_f^\ell = \begin{cases} s_f^\ell & q^\ell = 0 \\ \frac{q_f^\ell}{q^\ell} B^\ell & q^\ell > 0 \end{cases}$
stochastic counter
$dN(t; q^\ell, s^\ell) > 0$?
$q^\ell = q_{\max}^\ell,\ s^\ell > B^\ell$?
discrete modes
Random Early Drop
active queuing
ℓ-queue-full
$\dot{q}_f^\ell = s_f^\ell - \frac{q_f^\ell}{q^\ell} s^\ell$
$\dot{z}^\ell = s^\ell - B^\ell$
$r_f^\ell = \frac{q_f^\ell}{q^\ell} B^\ell$
$s^\ell \le B^\ell$?
$\mathfrak{E}$[drop in flow $\mathbf{f}^*$]
$z^\ell := 0$
$z = L$?


Network dynamic & Congestion control
routing
in-queue rates
out-queue rates
sending rates
queue dynamics
end2end congestion control
TCP/UDP
drops

Additive Increase/Multiplicative Decrease

1. While there are no drops, increase w_f by 1 on each RTT (additive increase)
2. When a drop occurs, divide w_f by 2 (multiplicative decrease)
   (congestion controller constantly probe the network for more bandwidth)

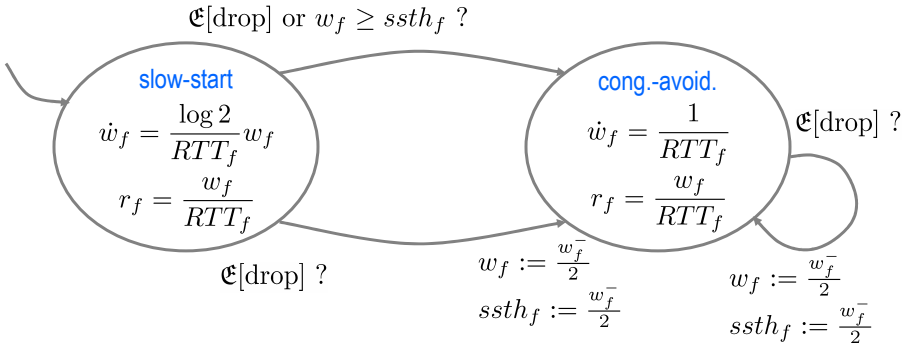


TCP-Reno is based on AIMD but uses other discrete modes to improve performance

Slow start

In the beginning, pure AIMD takes a long time to reach an adequate window size

3. Until a drop occurs (or a threshold $ssth_f$ is reached), double w_f on each RTT
4. When a drop occurs, divide w_f and the threshold $ssth_f$ by 2

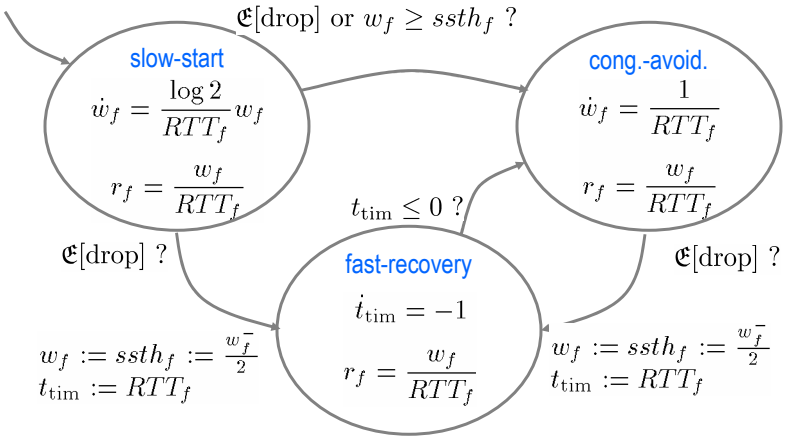


especially important for short-lived flows…

Fast recovery

After a drop is detected, new data should be sent while the dropped one is retransmitted

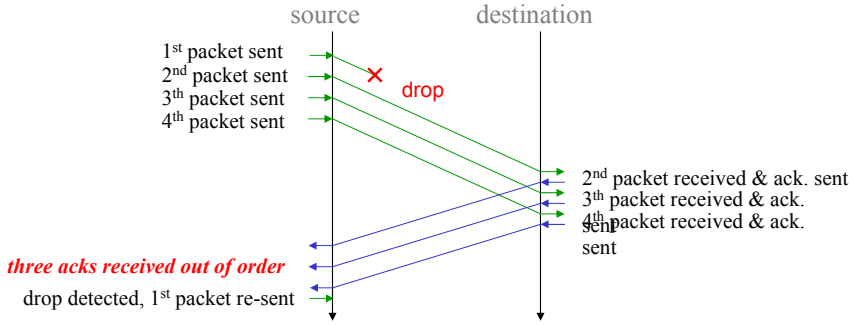
5. During retransmission, data is sent at a rate consistent with a window size of $w_f/2$



(consistent with TCP-SACK for multiple consecutive drops)

Timeouts

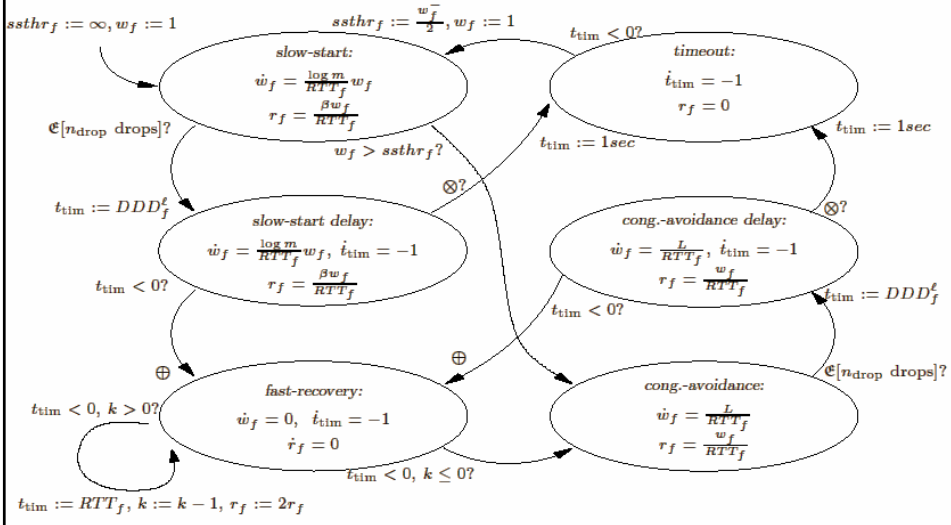
Typically, drops are detected because one acknowledgment in the sequence is missing.



When the window size becomes smaller than 4, this mechanism fails and drops must be detected through acknowledgement timeout.

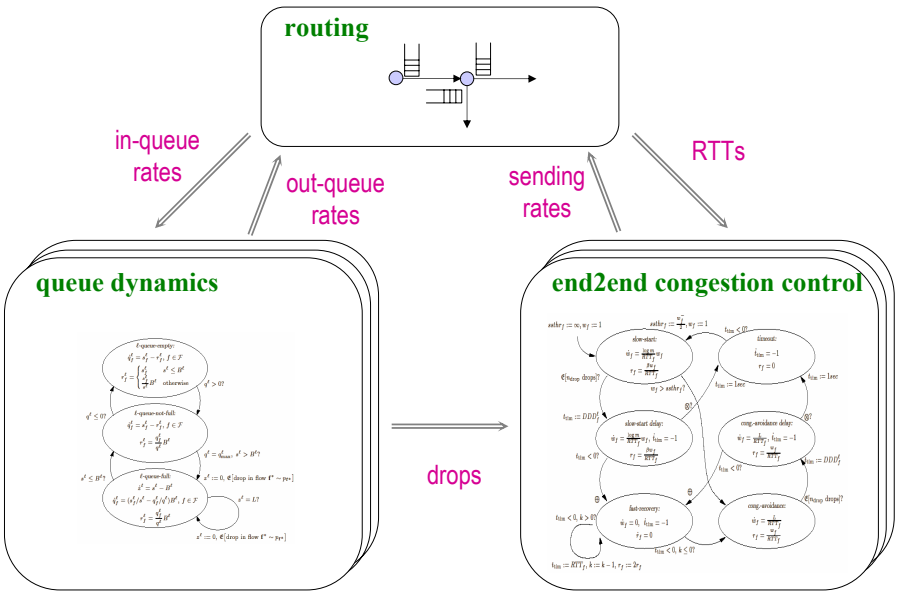
6. When a drop is detected through timeout:
   a. the slow-start threshold $ssth_f$ is set equal to half the window size,
   b. the window size is reduced to one,
   c. the controller transitions to slow-start

Fast recovery, timeouts, drop-detection delay…



TCP SACK version

Network dynamic & Congestion control

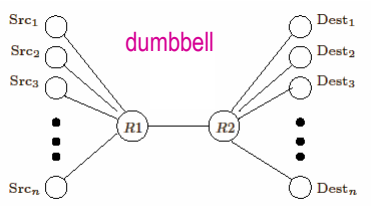


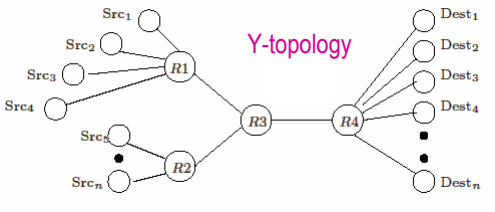
see SIGMETRICS paper for on/off TCP & UDP model

Validation methodology

Compared simulation results from
- ns-2 packet-level simulator
- hybrid models implemented in Modelica

Plots in the following slides refer to two test topologies

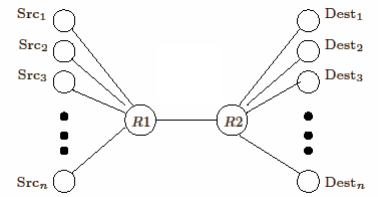




- 10ms propagation delay
- drop-tail queuing
- 5-500Mbps bottleneck throughput
- 0-10% UDP on/off background traffic

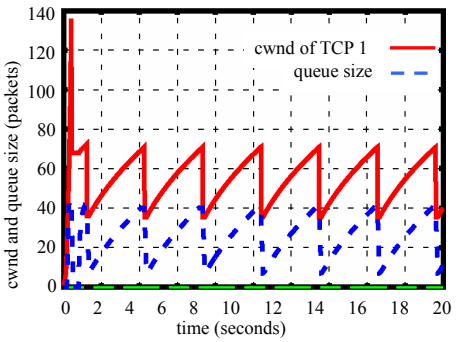
- 45,90,135,180ms propagation delays
- drop-tail queuing
- 5-500Mbps bottleneck throughput
- 0-10% UDP on/off background traffic

Simulation traces

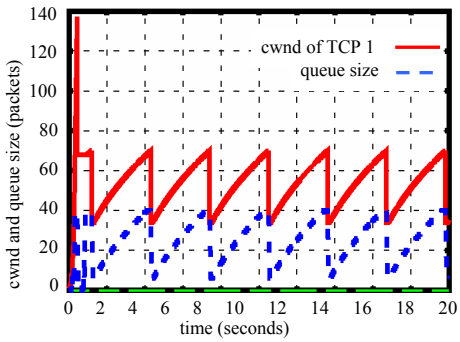


- single TCP flow
- 5Mbps bottleneck throughput
- no background traffic

hybrid model

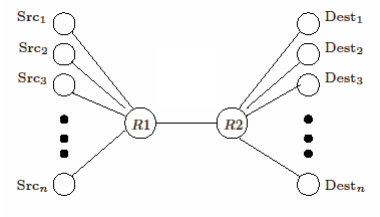


ns-2



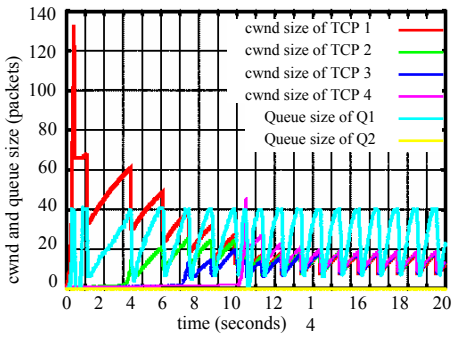
slow-start, fast recovery, and congestion avoidance accurately captured

Simulation traces

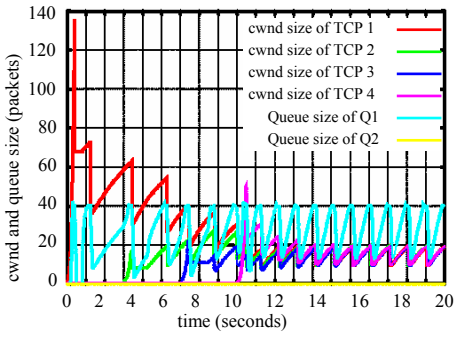


- four competing TCP flow (starting at different times)
- 5Mbps bottleneck throughput
- no background traffic

hybrid model

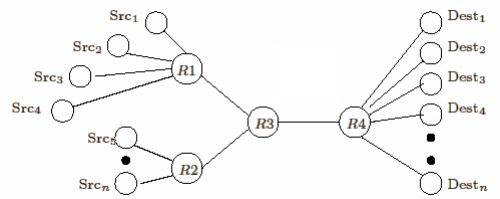


ns-2



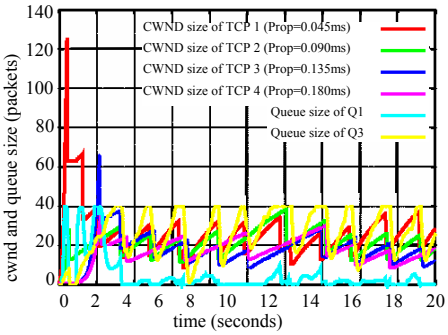
the hybrid model accurately captures flow synchronization

Simulation traces

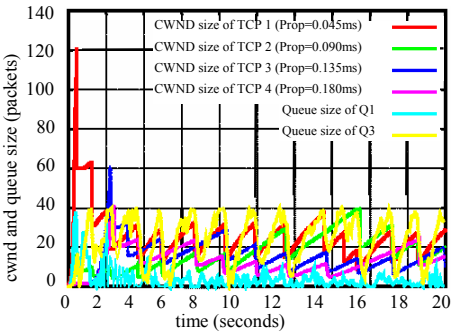


- four competing TCP flow (different propagation delays)
- 5Mbps bottleneck throughput
- 10% UDP background traffic (exp. distributed on-off times)

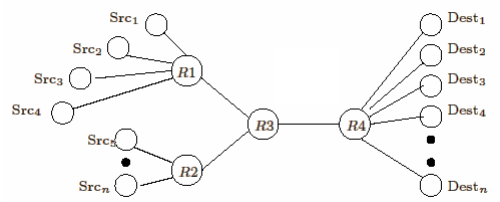
hybrid model



ns-2



Average throughput and RTTs



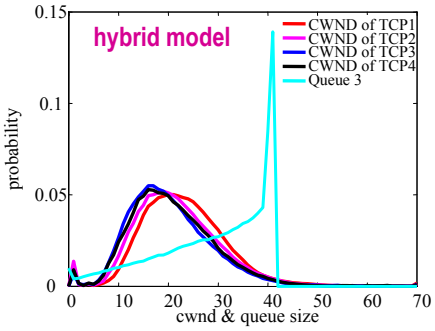
- 45,90,135,180ms propagation delays
- drop-tail queuing
- 5Mbps bottleneck throughput
- 10% UDP on/off background traffic

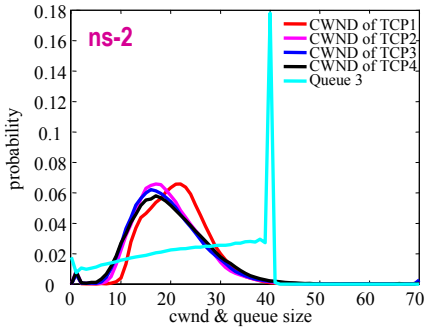
- four competing TCP flow (different propagation delays)
- 5Mbps bottleneck throughput
- 10% UDP background traffic (exp. distributed on-off times)

	Thru. 1	Thru. 2	Thru. 3	Thru. 4	RTT1	RTT2	RTT3	RTT4
ns-2	1.873	1.184	.836	.673	.0969	.141	.184	.227
hybrid model	1.824	1.091	.823	.669	.0879	.132	.180	.223
relative error	2.6%	7.9%	1.5%	.7%	9.3%	5.9%	3.6%	2.1%

the hybrid model accurately captures TCP unfairness for different propagation delays

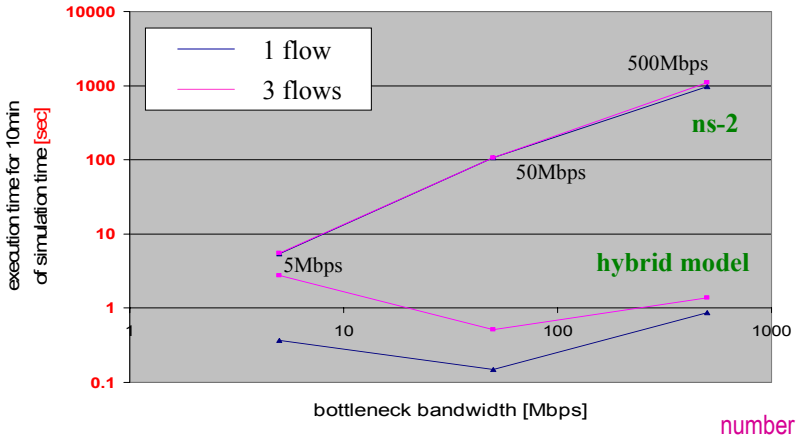
Empirical distributions





the hybrid model captures the whole distribution of congestion windows and queue size

Execution time



number of flows

- ns-2 complexity approximately scales with $o(nT)$ (# packets)
- hybrid simulator complexity approximately scales with $o\left(\frac{1}{RTT^2}\frac{n}{T}\right)$

per-flow throughput

hybrid models are particularly suitable for large, high-bandwidth simulations (satellite, fiber optics, backbone)