

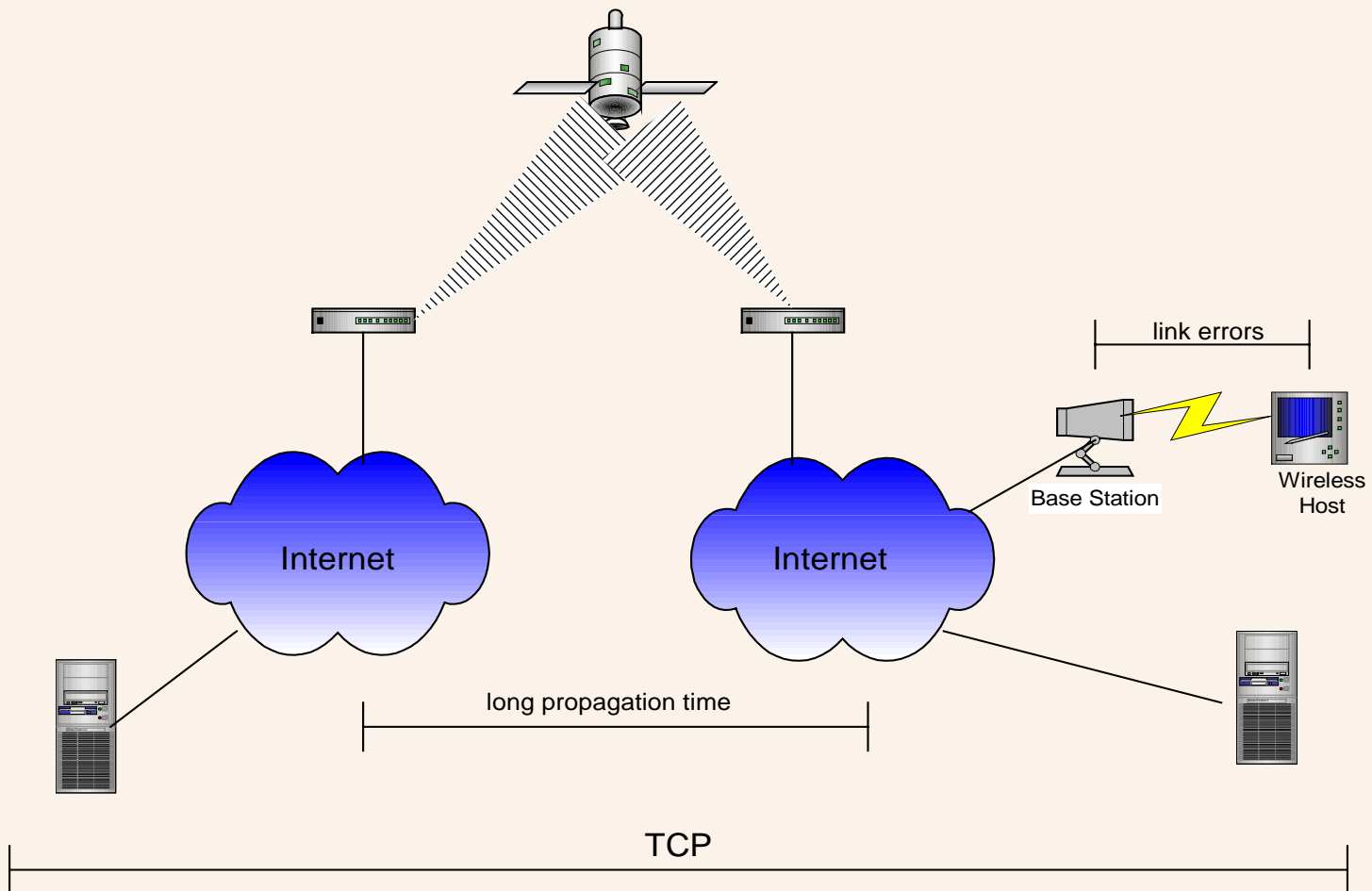
TCP Westwood with Adaptive Bandwidth Estimation to Improve Efficiency/Friendliness Tradeoffs

*Mario Gerla, Bryan Kwok Fai Ng, M. Y. Sanadidi,
Massimo Valla, I Wang*

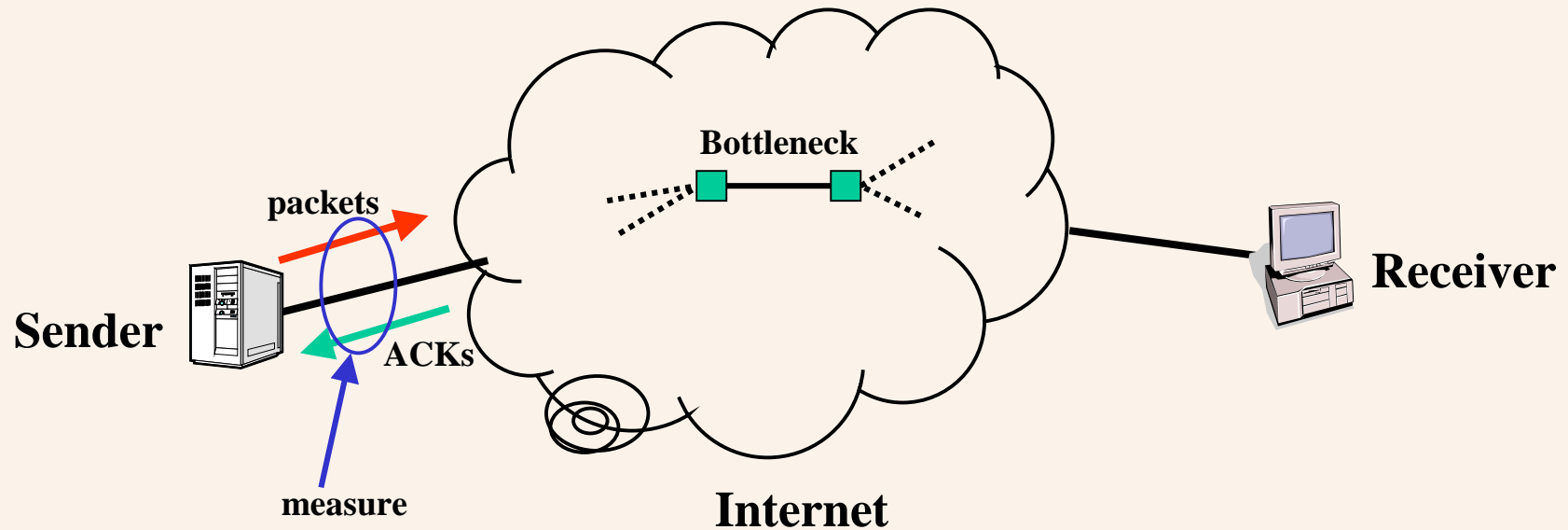
*UCLA Computer Science Department
<http://www.cs.ucla.edu/NRL/hpi/tcpw/>*

Hybrid High Speed Networks

TCPW Is Particularly Effective Over *Large Leaky Pipes*



Using Bandwidth Estimation In Congestion Control

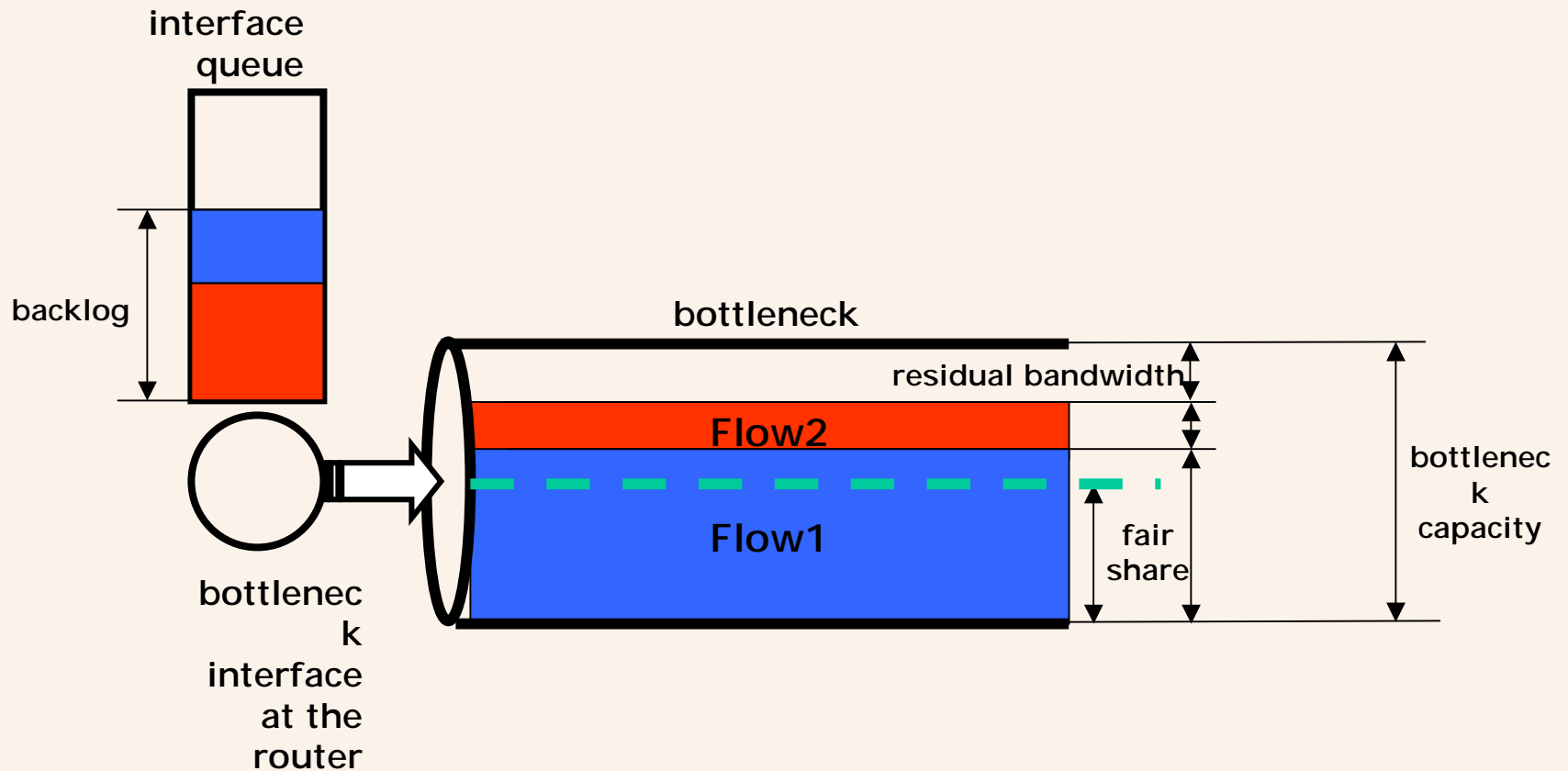


- **Goal:** determine the *fair share* on the bottleneck
- Since fair share is difficult to determine, we instead compute a *Fair Share Estimate (FSE)*

Definitions

- Connection Service Rate
 - ▶ The bottleneck router service rate regarding a connection of interest, might also be called bandwidth share
- Residual Capacity
 - ▶ Available bandwidth leftover by other flows
- Capacity
 - ▶ Physical bandwidth of the bottleneck
- Fair Share
 - ▶ The connection fair share of the link bandwidth

Bandwidth Measures



Related Bandwidth Estimation Research

- Packet Tailgating
 - ▶ Kevin Lai and Mary Baker (2000)
 - ▶ A deterministic model of packet delay
 - ▶ Use model to derive a new technique for link bandwidth measurement
 - ▶ Requires significantly fewer packets than existing techniques
- Pathchar, Pchar
 - ▶ Jacobson (1997), Downey (1999)
- B|C Probe
 - ▶ Downey (1999)
- Packet Dispersion Measurements
 - ▶ Dovrolis, Ramanathan, and Moore (2001)

TCP Vegas

- Idea: Sender watches for some signs that congestion is setting in:
 - ▶ (1) RTT grows;
 - ▶ (2) Sending rate flattens
- Sender adjust sending rate to avoid filling the buffer
 - ▶ Let $BaseRTT$ be the minimum of all measured RTTs
 - ▶ Compute $ExpectRate = CongestionWindow / BaseRTT$
 - ▶ Source calculates sending rate: $ActualRate$ (=bytes sent in RTT/RTT)
 - ▶ Source compares $ActualRate$ with $ExpectRate$

$$Diff = ExpectedRate - ActualRate$$



TCP Vegas

Advantages

- Achieving higher throughput with less retransmissions

Disadvantages

- Fairness problem has been reported
 - ▶ When a TCP Vegas connection shares a link with TCP Reno connection, the TCP Reno connection uses most of the buffer space and the TCP Vegas connection backs off, interpreting this as a sign of network congestion
 - ▶ Unfairness between TCP Vegas connections. The connection that starts up later could observe a larger RTT, causing the congestion window to be lower

Packet-Pair Flow Control

- Method to estimate bottleneck service rate to a connection

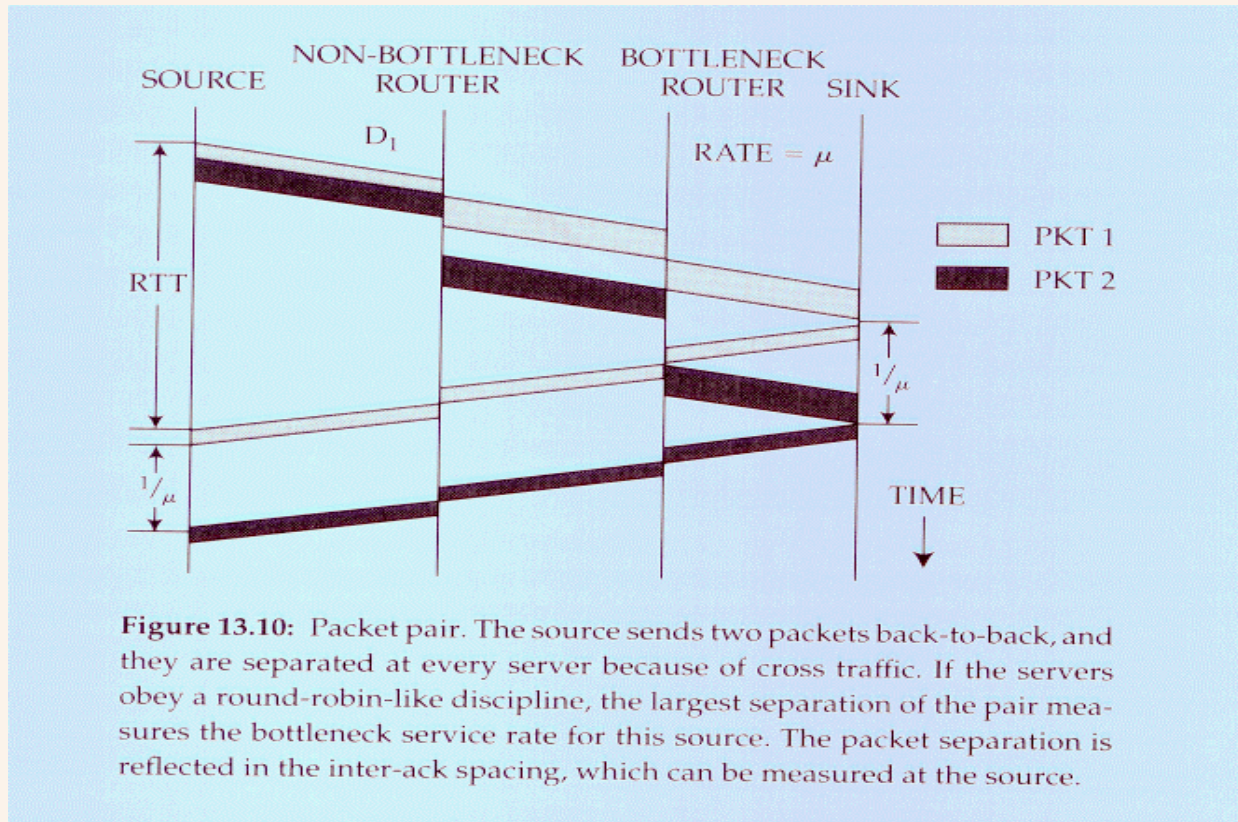


Figure 13.10: Packet pair. The source sends two packets back-to-back, and they are separated at every server because of cross traffic. If the servers obey a round-robin-like discipline, the largest separation of the pair measures the bottleneck service rate for this source. The packet separation is reflected in the inter-ack spacing, which can be measured at the source.

Packet Pair Flow Control

- Measures the service rate achieved by the connection and filter as follows:

$$\hat{\mu}(k+1) = \alpha\hat{\mu}(k) + (1-\alpha)\mu(k)$$

Where $\mu(k)$ is the bottleneck service rate detected by the k th ack pair, α is the exponential averaging coefficient

- Estimates the buffer backlog

$$X = S - R\hat{\mu}(k+1)$$

Where R is the round trip propagation delay, and S is the packets outstanding (i.e., transmitted but not ACKed)

Packet Pair Flow Control

- Adjusts the **transmitting rate** to maintain the TCP connection bottleneck queue equal to a target called **setpoint** (B)

$$\lambda(k+1) = \hat{\mu}(k+1) + (B - X) / R$$

Disadvantage

- Under round-robin, packet pair measures fair share; otherwise measure is inaccurate, and can overestimate fair share, up to link capacity
- Rate control, not window control, attempts to stay close to set *backlog*

TCP-Peach for Satellite Links

- End-to-end solution to improve TCP throughput performance in satellite networks
- Assume all the routers on the connection path apply some priority mechanism
- Use dummy (low priority) segments to probe available bandwidth, and use the info as follows:
 - ▶ At beginning of a connection, to improve over Slow Start
 - ▶ When a loss is detected, to improve over Fast Recovery
 - ▶ Detect congestion and adjust sending rate to avoid packet loss

TCP Westwood

- Enhance congestion control via **Fair Share Estimates** (*FSE*)
 - ▶ Estimates are computed at the sender by *sampling* and *exponential filtering*
 - ▶ Samples are determined from *ACK inter-arrival times* and info in *ACKs* regarding amounts of *bytes delivered*
- FSE is used by sender to appropriately set *cwnd* and *ssthresh* after packet loss (indicated by 3 DUPACKs, or Timeout)

TCPW Algorithm Outline

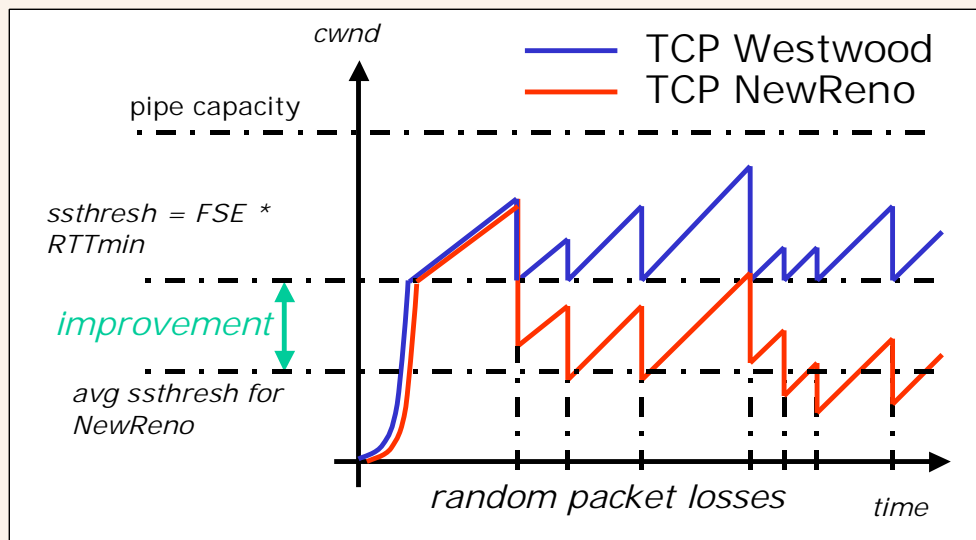
- **When three duplicate ACKs are detected:**
 - ▶ set $ssthresh = FSE * RTT_{min}$ (instead of $ssthresh = cwin / 2$ as in Reno)
 - ▶ if ($cwin > ssthresh$) set $cwin = ssthresh$
- **When a TIMEOUT expires:**
 - ▶ set $ssthresh = FSE * RTT_{min}$ (instead of $ssthresh = cwnd / 2$ as in Reno) and $cwin = 1$

Note: RTT_{min} = min round trip delay experienced by the connection

TCPW Benefits

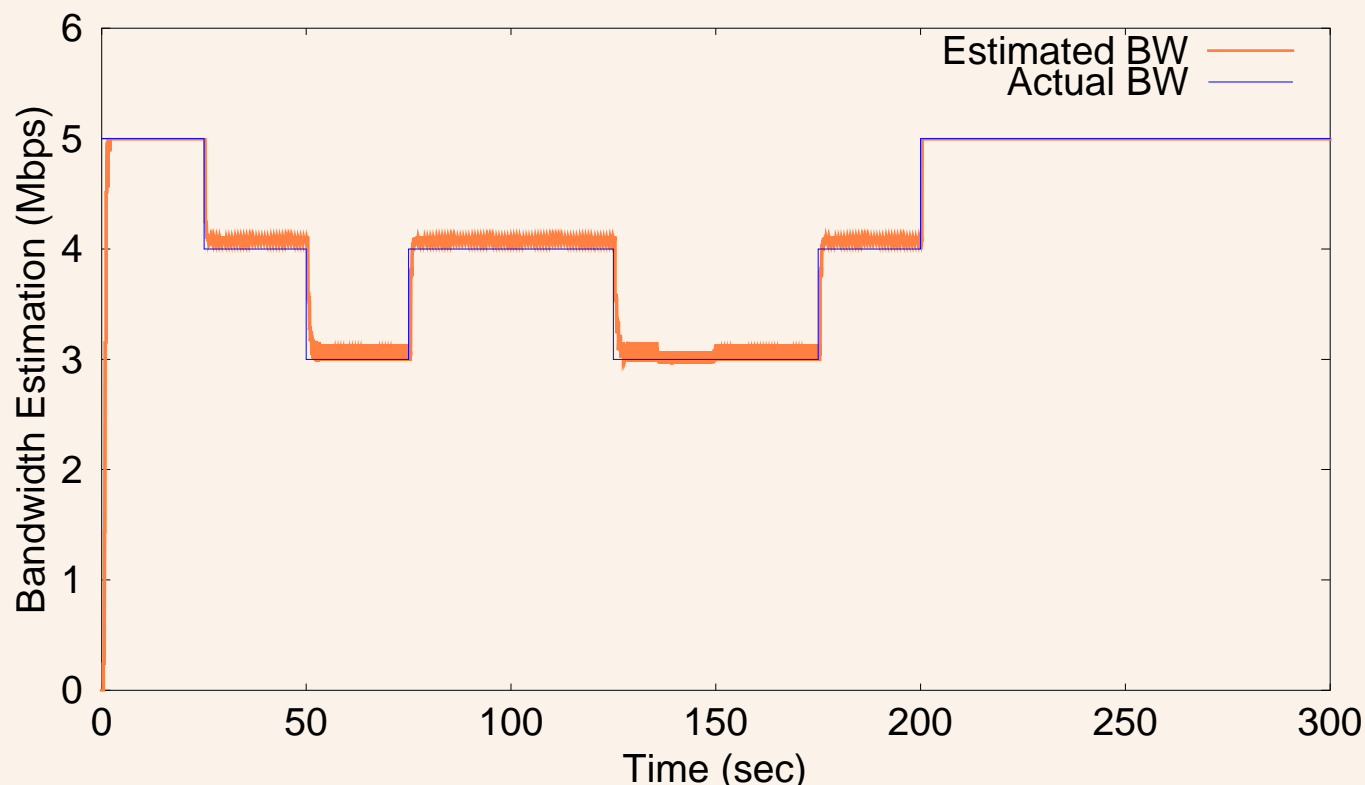
- *Efficiency*:
 - ▶ Better link utilization when losses are due to random errors (wireless) as well as buffer overflow, i.e. *Leaky Pipe*
 - ▶ Gain is significant for high bandwidth delay product, i.e. *Large Pipe*
- *Fairness*: better fairness under varying RTT
- Note: Must ensure *Friendliness* and *Stability* as well

TCPW And Random Loss



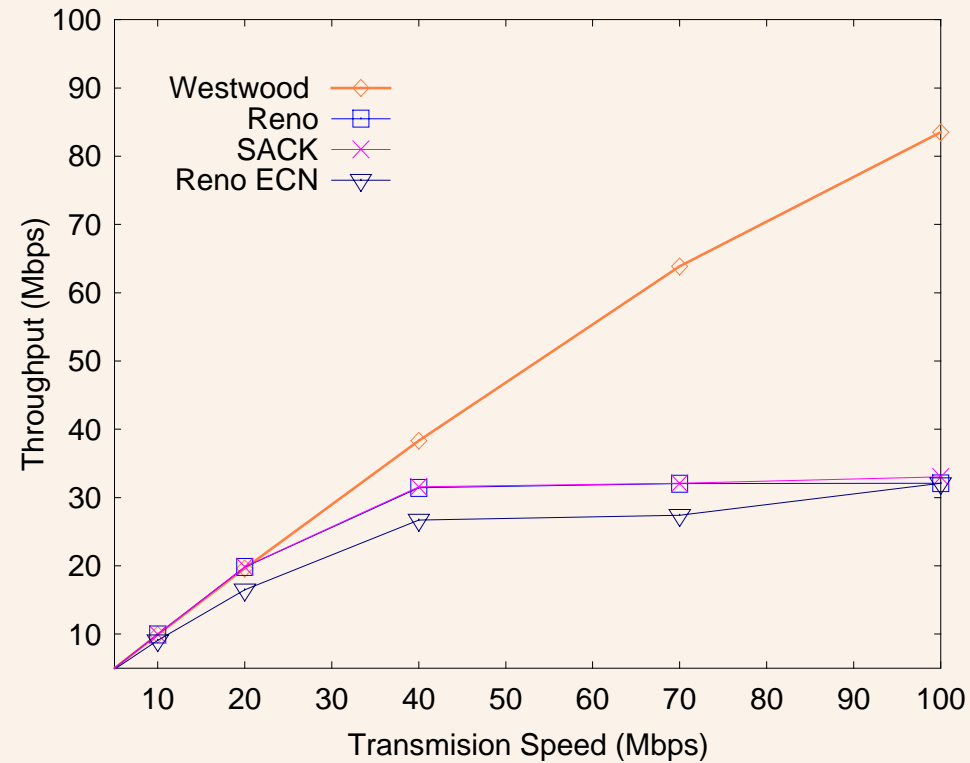
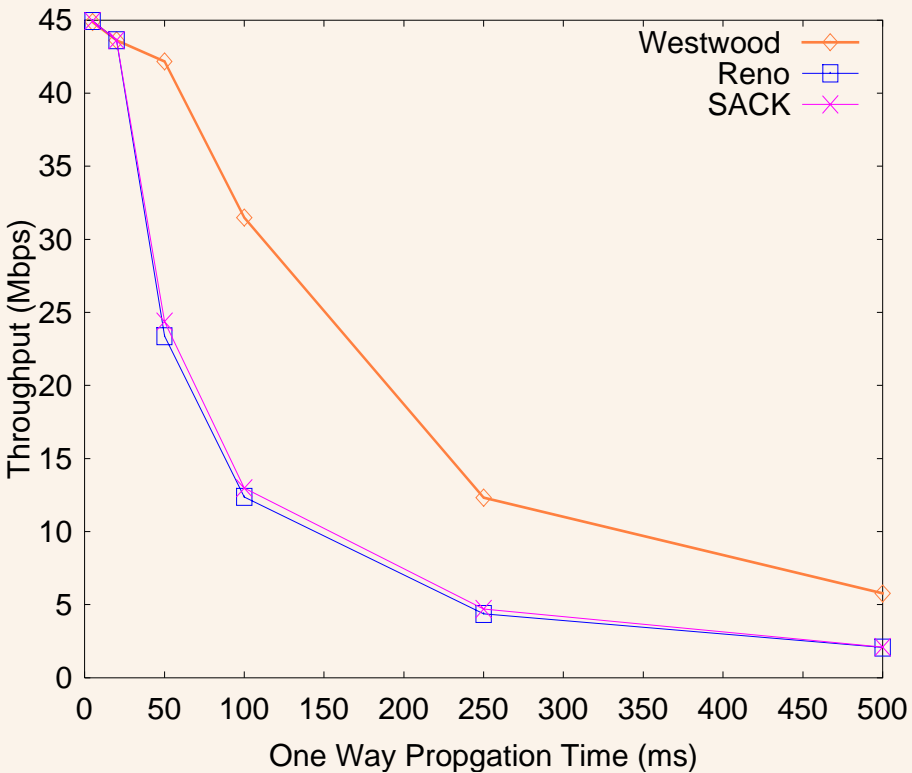
- NewReno overreacts to random loss ($cwin$ cut by half)
- A small fraction of isolated “randomly” lost packets does not impact the FSE estimate
- Thus, $cwnd = FSE * RTTmin$ remains unchanged
- As a result, TCPW efficiency is higher than NewReno

Bandwidth Estimation Accuracy

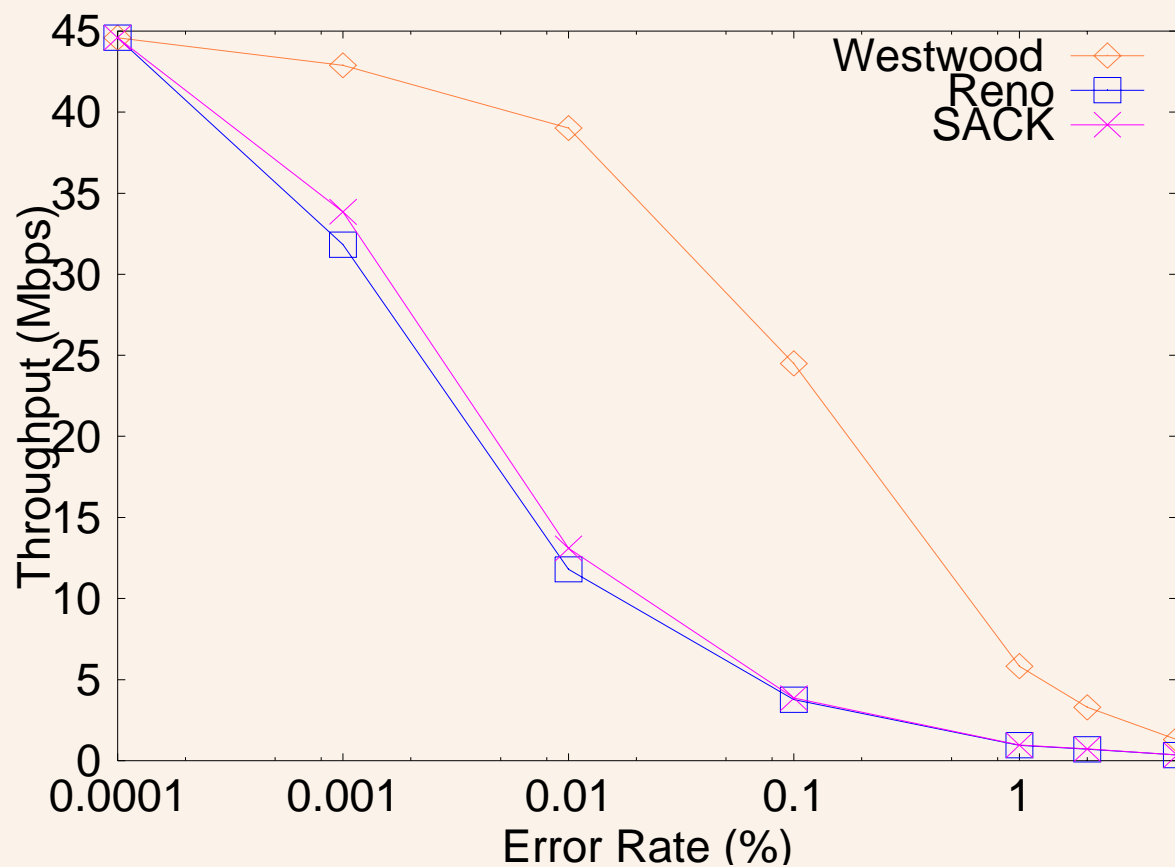


BSE Accuracy (With Concurrent On/Off UDP Traffic)

Throughput Gain



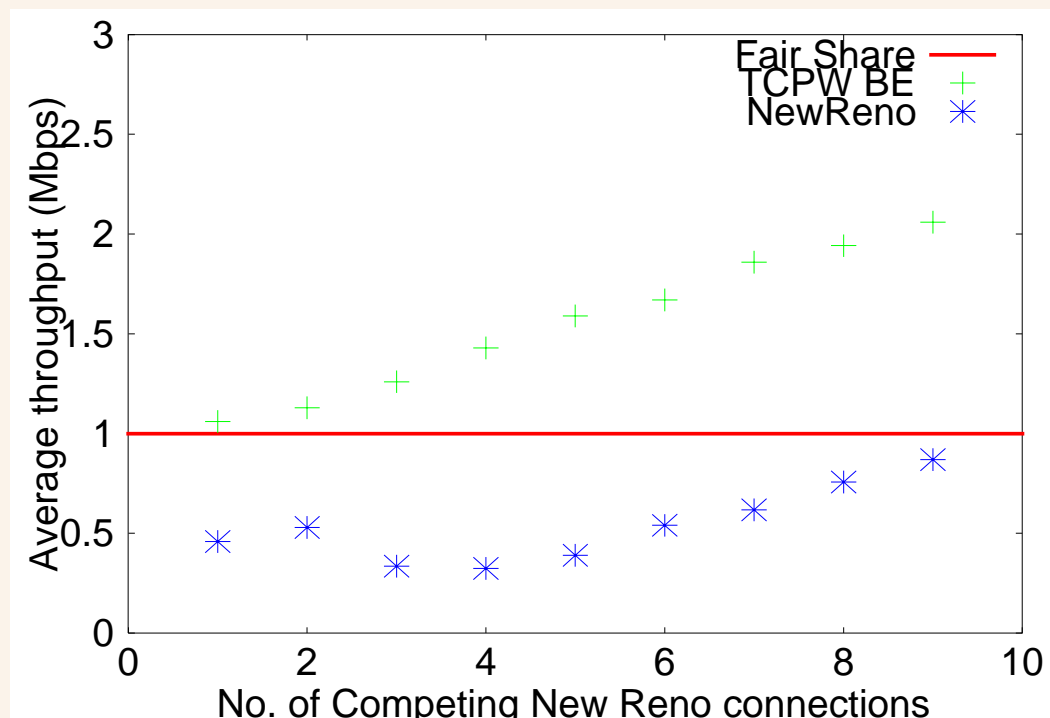
Throughput Gain (Cont'd)



TCPW Friendliness

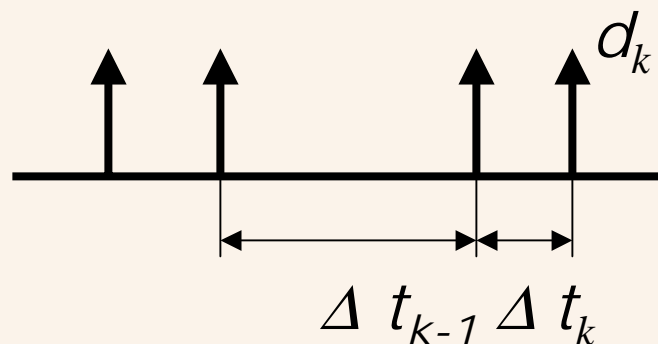
- TCPW connections should not hurt coexisting TCP NewReno; it should only get the bandwidth that was not possible to use by NewReno's "blind" window reductions

- OBSERVATION:**
The **bandwidth estimation method** is critical to guarantee **efficiency improvement** while maintaining **friendliness** towards NewReno
- TCPW BE overestimates the fair share => unfriendliness towards NewReno



10 total connections, no random errors

TCPW Bandwidth Estimation (BE)



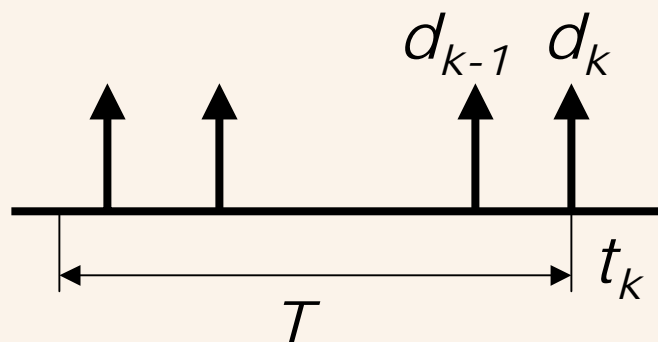
- First TCPW version (from now on: TCPW BE) used a bandwidth estimation (BE) given by:

$$b_k = d_k / (t_k - t_{k-1}) \quad \text{sample}$$

$$BE_k = \alpha_k BE_{k-1} + (1 - \alpha_k) \left(\frac{b_k + b_{k-1}}{2} \right) \quad \text{exponential filter}$$

$$\alpha_k = \frac{2\tau - \Delta t_k}{2\tau + \Delta t_k} \quad \text{filter gain}$$

TCPW Rate Estimation (RE)



T is the sample interval

- Another possible estimate can be obtained by looking at the data ACKed during the latest interval of time T (from now on: TCPW RE). This estimation is equivalent to a recent rate estimation (RE):

$$b_k = \frac{\sum_{t_j > t_k - T} d_j}{T}$$

sample ($T = \text{RTT}$)

$$RE_k = \alpha_k RE_{k-1} + (1 - \alpha_k) \left(\frac{b_k + b_{k-1}}{2} \right)$$

exponential
filter

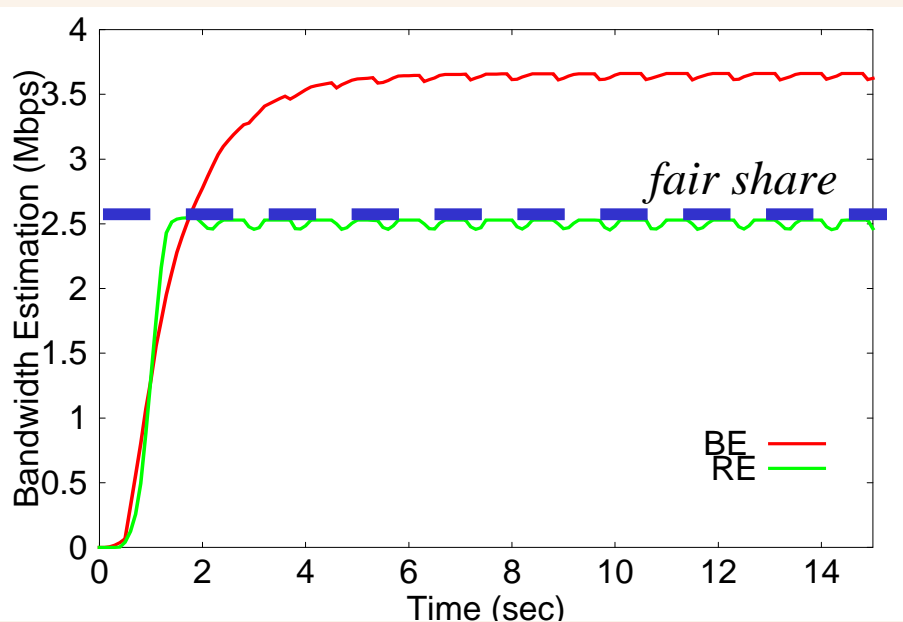
$$\alpha_k = \frac{2\tau - \Delta t_k}{2\tau + \Delta t_k}$$

filter gain

TCPW BE and RE compared

2 connections, sharing 5Mbps bottleneck

Congestion, no errors

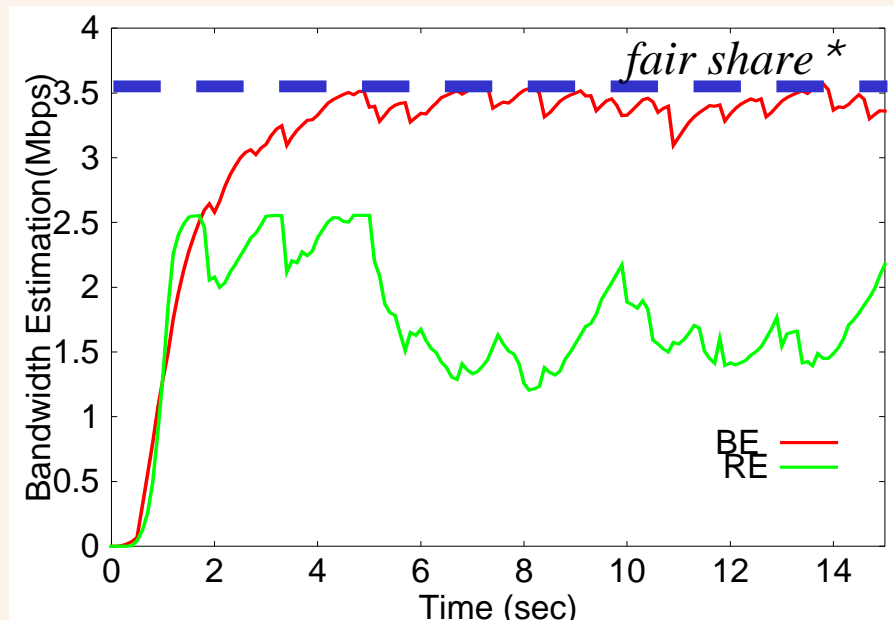


BE overestimates fair share (2.5 Mbps)



Not friendly towards NewReno

Errors (0.5%), no congestion



RE underestimates fair share (3.6 Mbps)



No throughput improvement

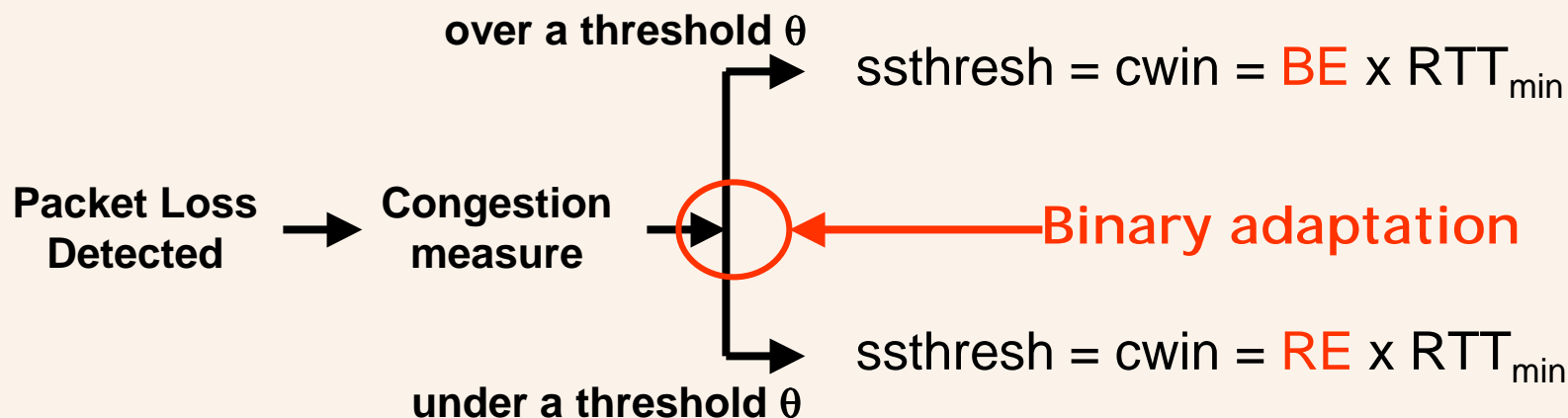
(*) TCPW fair share > 50% because NewReno is incapable of getting 50%

TCPW Adaptation

- Neither RE nor BE estimates are accurate estimates of FSE in all cases
 - ▶ BE is more effective when random error is the predominant cause of loss
 - ▶ RE method is more appropriate when packet losses are due to congestion (router buffer overflow)
- **IDEA:** We need *adaptation*: choose between an aggressive estimate (like BE) and a conservative estimate (like RE) depending on a measure of congestion intensity
- Use a congestion level measure as an indicator of predominant cause of packet loss

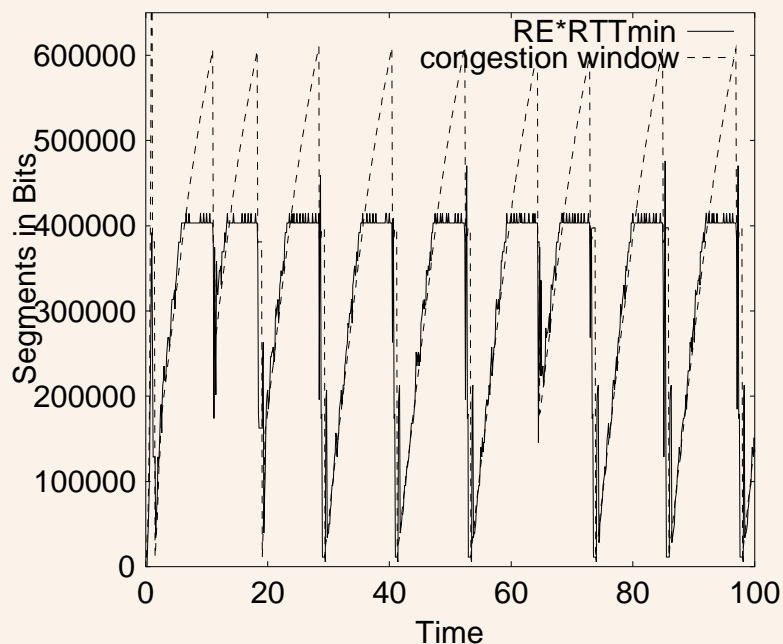
Combining Rate and Bandwidth Estimations

- TCP Westwood with *Combined Rate and Bandwidth* estimation (**CRB**) uses a congestion measure to choose between RE or BE upon packet loss to set the ssthresh

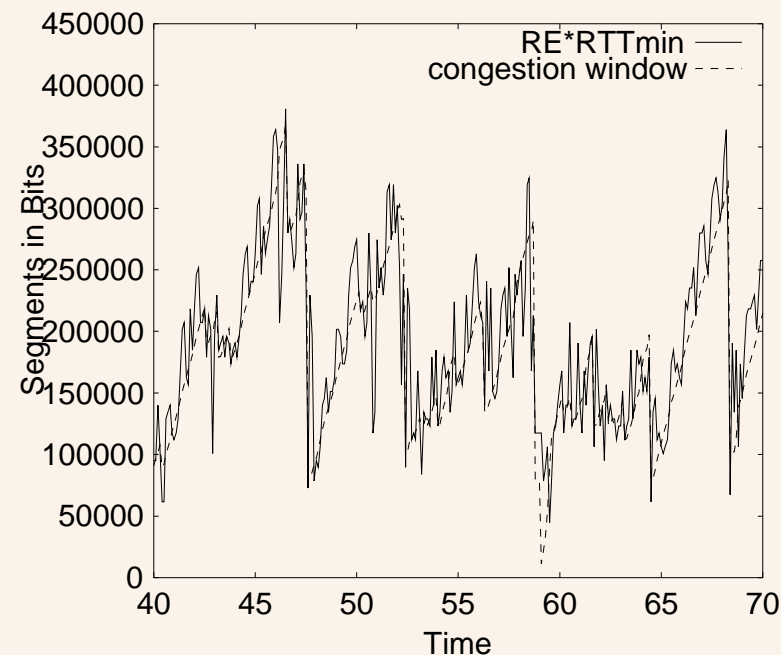


Congestion Measure

- Congestion measure is obtained comparing $cwnd / RTT_{min}$ vs. RE; i.e instantaneous current sending rate vs. achieved rate

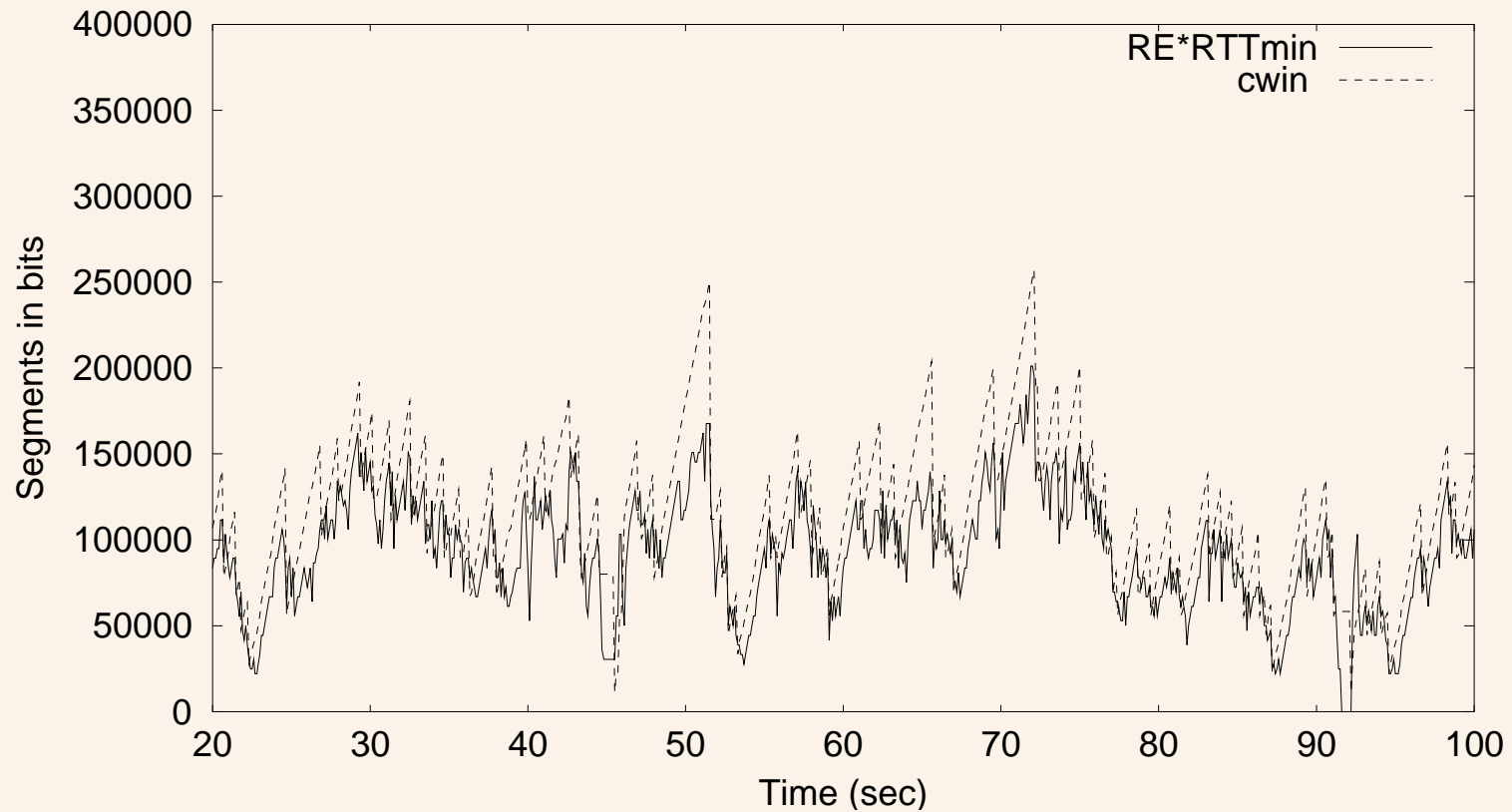


Congestion case



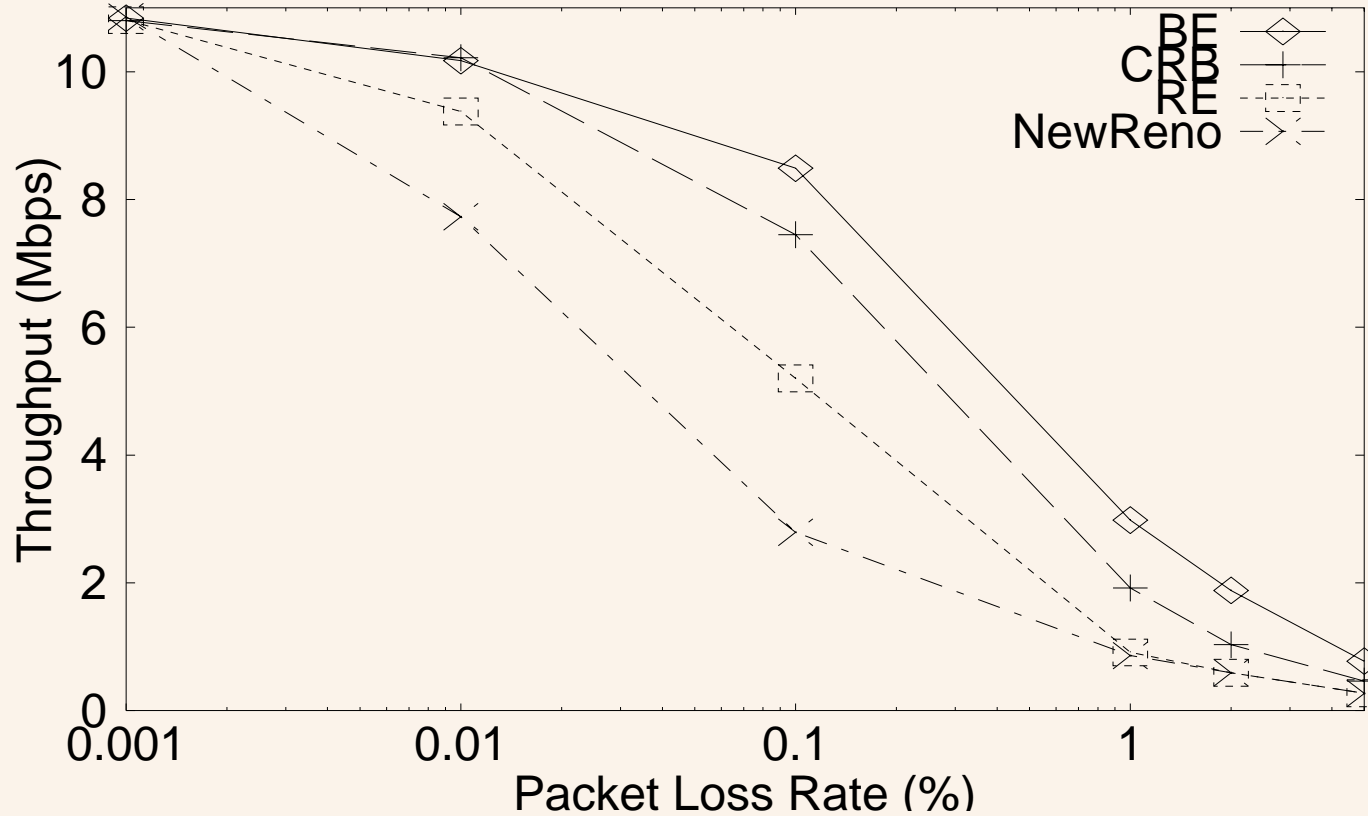
Link Error Case

Congestion Measure

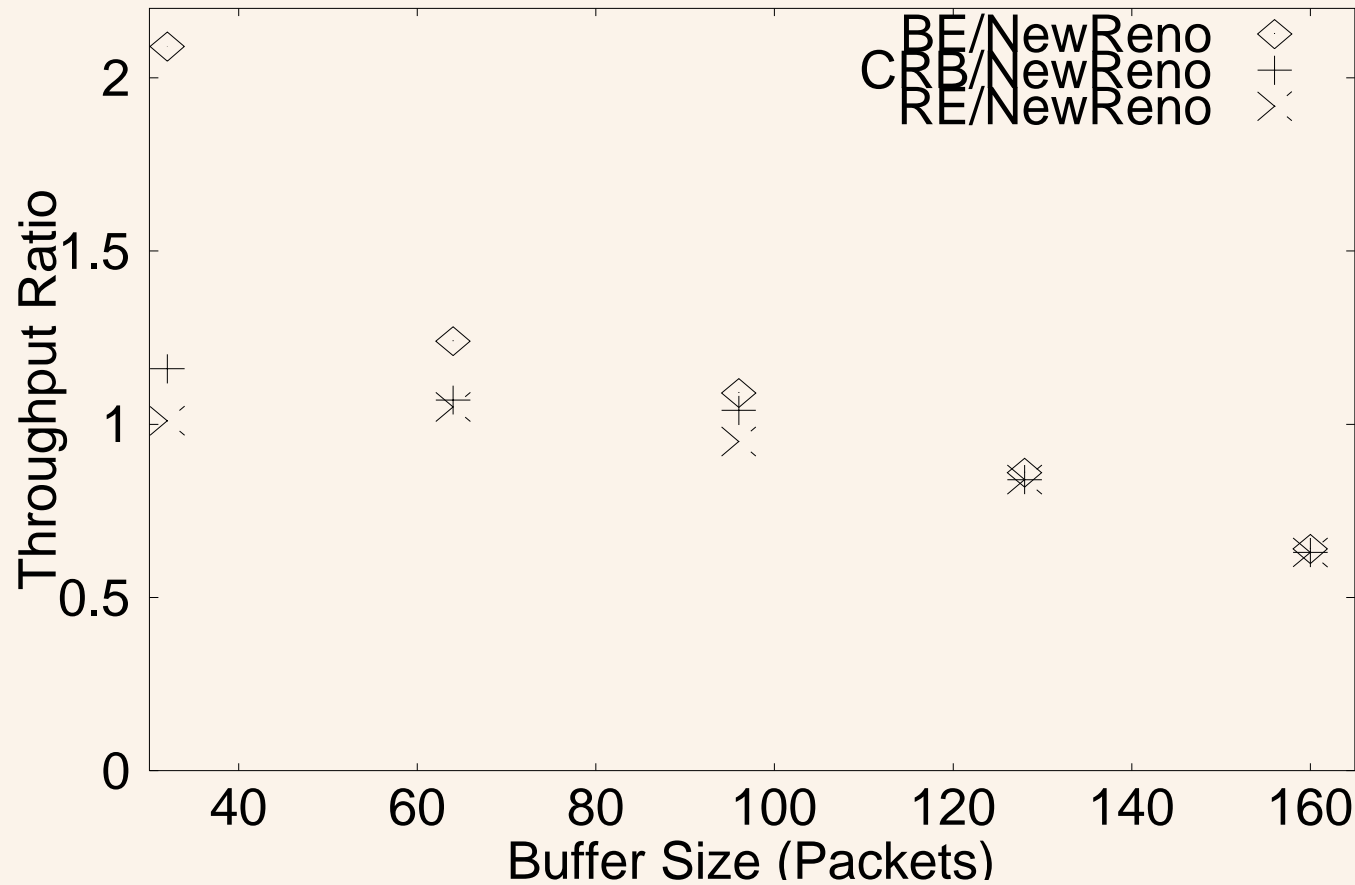


Both Congestion and Error Cause Loss

Impact of Random Error Rate



Impact of Buffer Size on Friendliness



Efficiency/Friendliness Tradeoff Graph

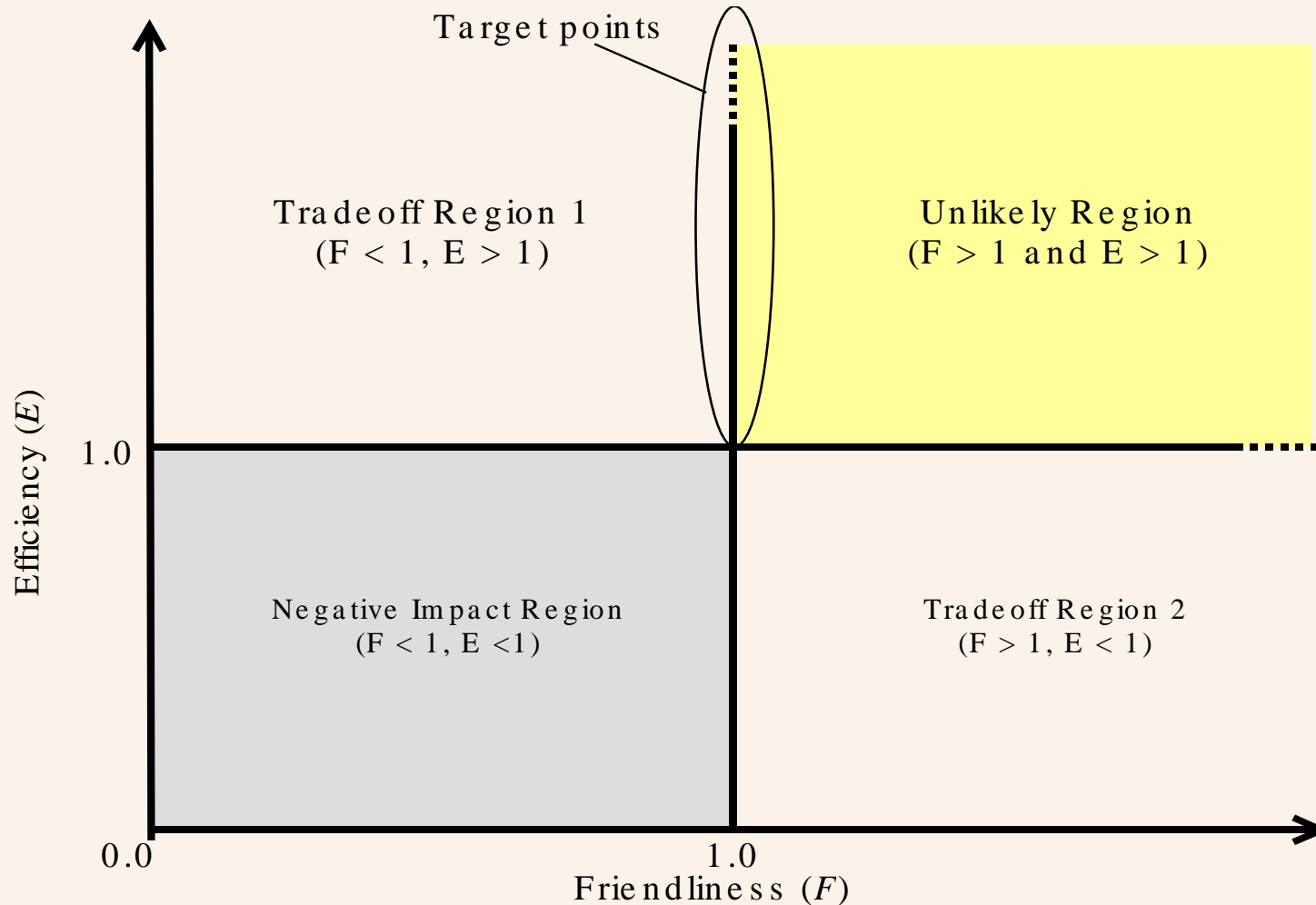
- To better visualize the efficiency/friendliness tradeoff we use the Efficiency/Friendliness Tradeoff Graph
- Each point in the graph is obtained by:
 - a) Running a simulation with N NewReno flows, we obtain avg thr t_{R1} and total link utilization U_1
 - b) Running a simulation with N/2 NewReno and N/2 TCPW flows, we obtain avg thr. t_{R2} of NewReno flows and total link utilization U_2
 - c) We define:

$$\text{Efficiency Improvement} \quad E = U_2 / U_1$$

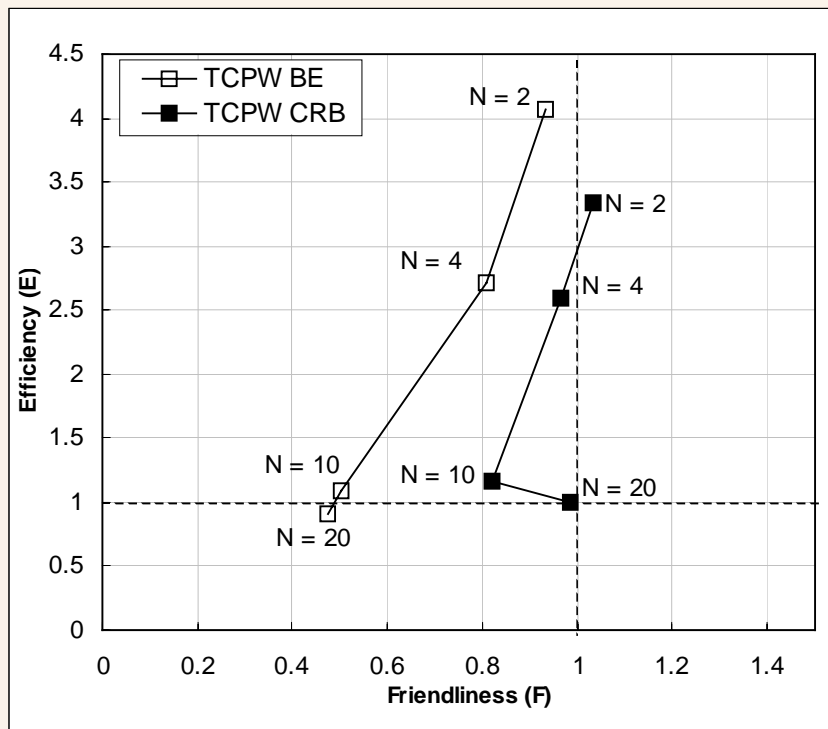
$$\text{Friendliness} \quad F = t_{R2} / t_{R1}$$

And thus obtain a point (F,E) on the graph

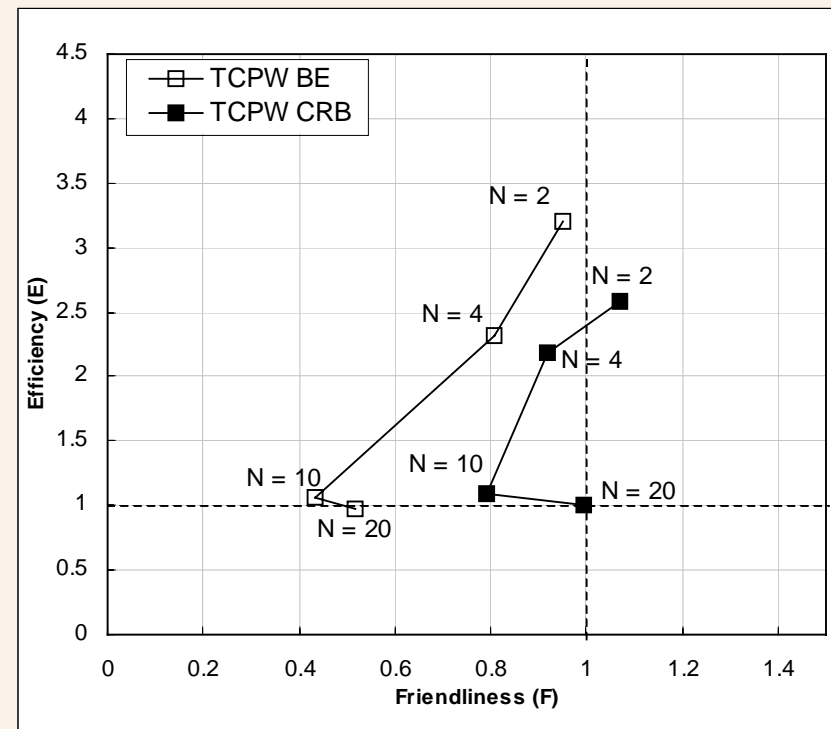
TCPW E/F Graph: Method Used



TCPW E/F Graph: Results



(a) 45 Mbps, 74ms, 0.1% error



(b) 11 Mbps, 74ms, 2% error

- TCPW CRB adapts to congestion or errors case and is both more efficient and friendly than original (non adaptive) TCPW BE

TCP Westwood vs other Protocols

- Note: the concept of using the BWE estimate to control the TCP flow is not new
- **TCP Vegas** monitors BWE and RTT to infer the **bottleneck backlog**; then, from backlog derive feedback to congestion window
- **Keshav's Packet Pair** scheme also monitors bandwidth to estimate the **bottleneck backlog** and compare to common target; adjust source **rate**

Summary

- Introduced the concept of Bandwidth Estimation and related work
- Reviewed end-to-end estimation based congestion control methods
- Presented TCP Westwood, the evolution of “fair share” estimate to improve the performance; showed simulation results to evaluate the method
- Compared TCPW with other methods