Ecology of Congestion Control Protocols

A. L. Narasimha Reddy
Dept. of Electrical and Computer Engineering
Texas A&M University

Outline

- TCP's Congestion Control
- TCP's difficulties
- Different CC protocols
- PERT in homogeneous environments
- PERT incremental deployment
- Network control of multiple protocols

TCP's Congestion Control

- Initially developed around 1988
  - Van Jacobson
- Employs Additive Increase Multiplicative Decrease (AIMD)
  - Be conservative when probing and on packet loss
  - Multiplicative decrease needed to flush queues
  - Shown to converge to a fair share by Chiu & Jain

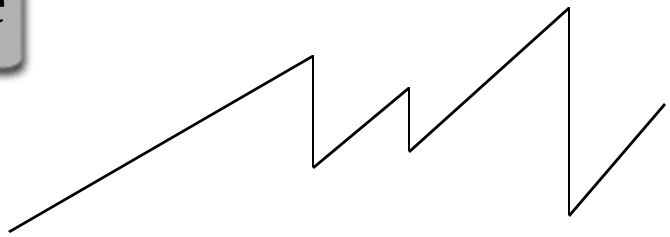
TCP's weaknesses

- Self-Induced packet losses
  - Reactive congestion response
- Doesn't distinguish congestion losses from channel losses
  - Problems in Wireless networks
- Additive Increase can't fill high-capacity, high-delay pipes
  - Intercontinental links connecting supercomputers

Other Congestion control approaches

- Proactive Congestion Avoidance
  - Employ metrics other than packet losses
  - Try to avoid packet losses
- Wireless CC protocols
  - Mitigate or separate impact of channel losses
- High-speed CC protocols
  - Employ more responsive/faster mechanisms to claim available link bandwidth

Proactive Congestion Avoidance



- TCP behavior :
  - Additive increase until packet loss is observed
  - Reduce cwnd by half ***after*** a packet loss
- Problem :
  - Bottleneck link buffers fill up
- Result :
  - Self induced packet losses
  - Wasted resources for retransmission of lost packets

Proactive Congestion Avoidance

- Well understood problem
- Explicit congestion notification by router
  - RED, REM, AVQ, BLUE, VCP etc. with ECN
- Explicit rate control by router
  - XCP, RCP, EMKC, JetMax etc.
- End-host based solutions
  - CARD, TRI-S, DUAL, VEGAS, CIM etc.

Proactive Congestion Avoidance

- Router based solutions
  - Easier to determine the onset of congestion
  - Difficult to deploy
- End-host based Solutions
  - Easier to deploy
  - Difficult to determine the onset of congestion
- We proposed an end-host based solution that emulates router-based solution --PERT

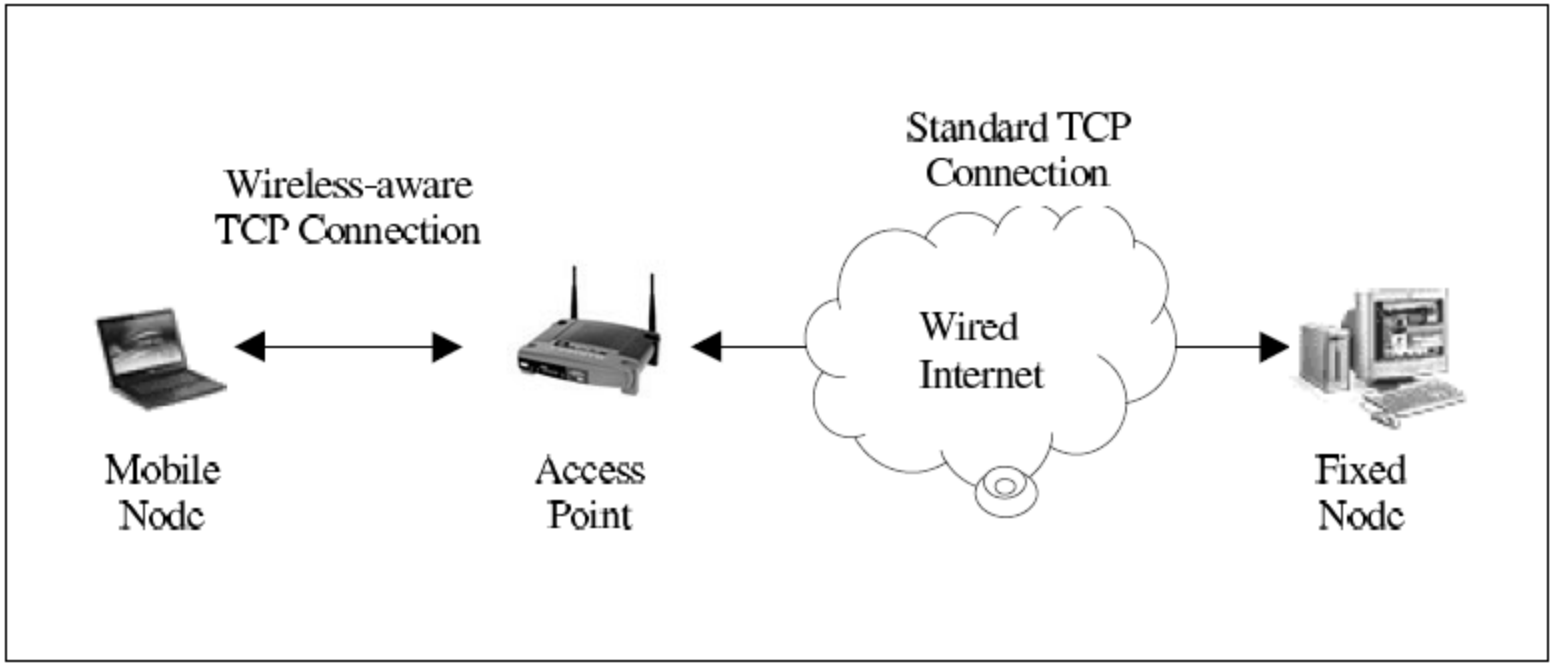
TCP in Wireless

- Behavior:
  - TCP treats losses as an indication of congestion
  - Reacts by reducing window by half
- Problem:
  - Can’t separate congestion losses and channel losses
- Result
  - At high channel loss rates, realized throughput can be low

TCP modifications for Wireless

- Split connection approaches:
  - Use multiple connections stitched at the wireless access point
  - Different congestion control applied on wired vs. wireless portions of the path
- Example Solutions
  - I-TCP, Mobile-TCP, MTCP, SRP, METP

TCP in Wireless



TCP modifications for Wireless

- Lower layer approaches:
  - Use techniques to mask channel errors or improve the link layer
  - Use retransmissions and other techniques
- Example Solutions
  - SNOOP (Network layer caching and retransmissions)
  - AIRMAIL (FEC at wireless link layer)
  - TULIP

TCP modifications for Wireless

- Explicit Notification Approaches:
  - Notify the sender of nature of losses
  - Congestion related or channel related?
- Example Solutions
  - ELN (Explicit Loss Notification)
  - ECN (Explicit Congestion Notification)

TCP modifications for Wireless

- End-to-end modifications retaining TCP semantics:
  - Modify sender/receiver with no support from intermediate nodes
- Example Solutions
  - TCP-DCR (Delayed Congestion Response)
    - Allow link layer retransmissions to work
  - WCTP –Inter-packet separation to determine rate
  - Westwood –Bandwidth Estimation using ack arrival rate

TCP problems in high-speed links

- Behavior:
  - TCP increases the window by 1 after each RTT
- Problem:
  - With high-delay links, window increases slowly
- Result:
  - Can’t saturate high-speed, high-delay links
  - Require very low, unrealistic loss rates

TCP problems in high-speed links

- Modify TCP AIMD Congestion Response
  - Design a different response function that claims available bandwidth faster
  - Can fill larger BW links at reasonable loss rates
- Example Approaches:
  - HSTCP (High-speed TCP)
  - BIC and CUBIC (Binomial TCP)
  - STCP (Scalable TCP)

TCP problems in high-speed links

- Modify TCP Increase function
  - Increase the window faster than 1 per RTT
  - The increase can be tied to the size of the window or to time since last packet loss
  - Decrease function has to be modified appropriately for fairness and convergence
- Example Approaches:
  - LTCP (Layered TCP)
  - HTCP (Hamiltonian TCP)

TCP problems in high-speed links

- Take both delay and losses into account
  - Increase faster when delays are low and at a lower pace at higher delays
  - Decrease less at lower delays and more at higher delays
- Example Approaches:
  - TCP-Illinois
  - CTCP
  - PERT

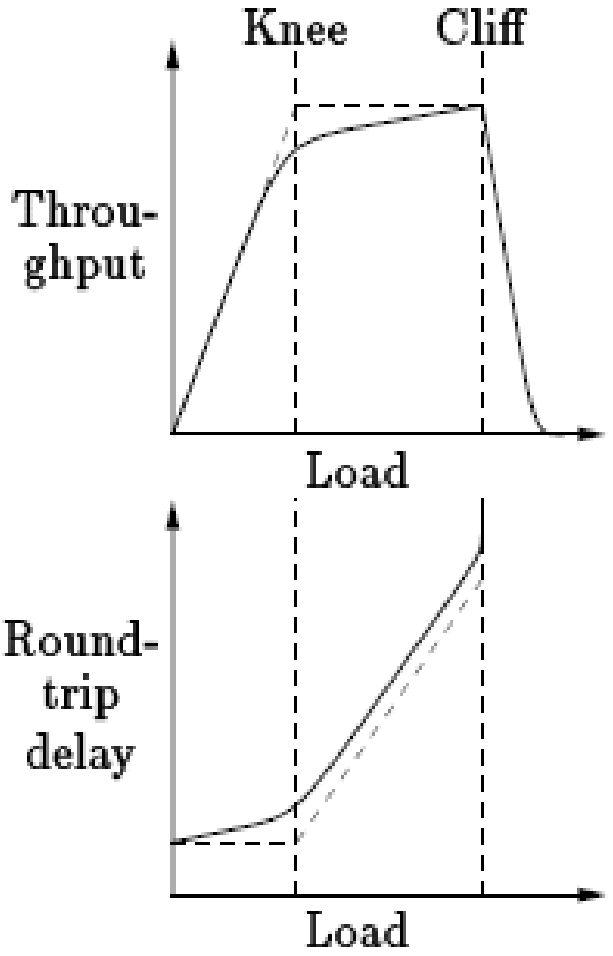
So Many problems –So many solutions

- Each TCP problem targeted
  - Many solutions proposed
  - Many variants of congestion control schemes
- One Unified solution for all the problems?
  - PERT

Proactive Congestion Avoidance

End-host based prediction

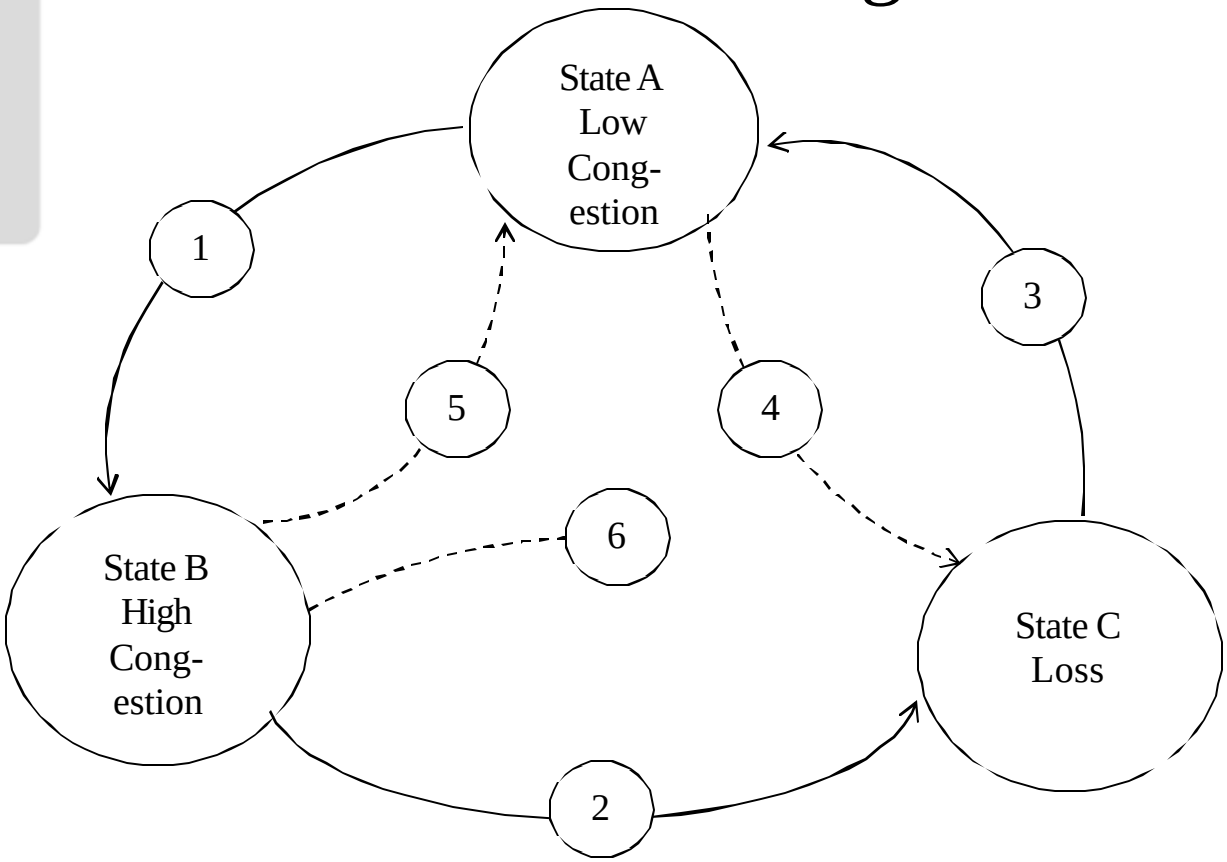
- Monitor throughput
  - Before link is full, throughput increases linearly with load
  - After link is full, throughput is constant at link capacity
- Monitor Delay
  - Before link is full, delay is low
  - After link is full, delay increases



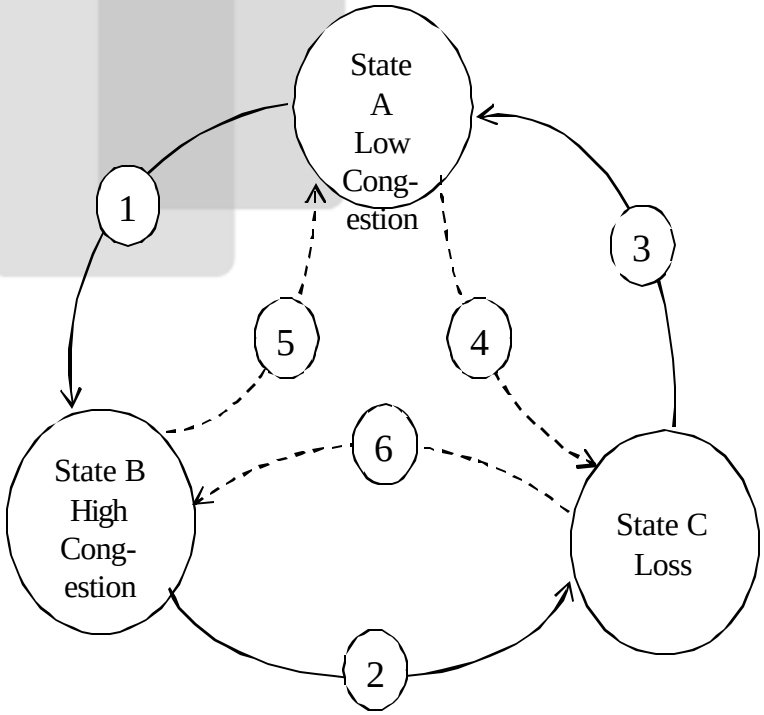
*Source: [CARD]

Proactive Congestion Avoidance

State Transition Diagram



Proactive Congestion Avoidance



$$\text{Prediction Efficiency} = \frac{(\text{“2” transitions})}{(\text{“2” transitions} + \text{“5” transitions})}$$

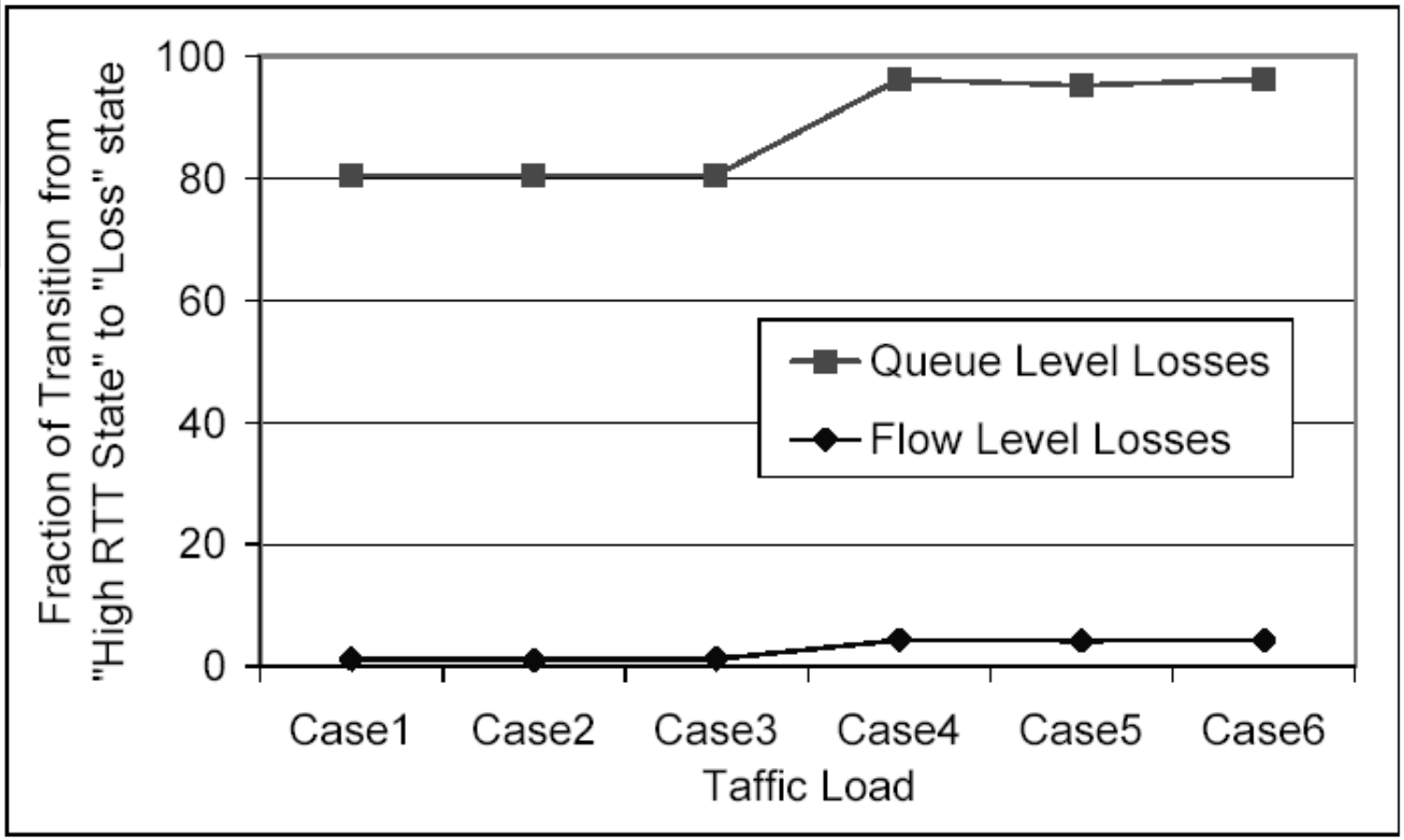
$$\text{False Positives} = \frac{(\text{“5” transitions})}{(\text{“2” transitions} + \text{“5” transitions})}$$

$$\text{False Negatives} = \frac{(\text{“4” transitions})}{(\text{“2” transitions} + \text{“4” transitions})}$$

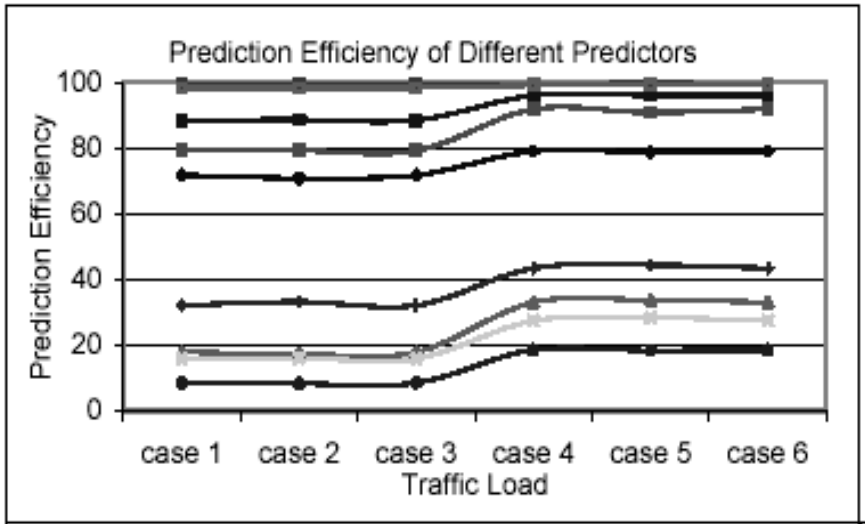
Proactive Congestion Avoidance

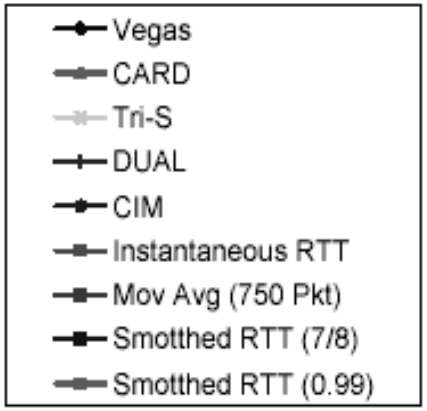
- Measurements show
  - Low correlation between RTT increase and loss
    - But loss measured at flow level
    - Absence of loss does *NOT* indicate absence of network congestion

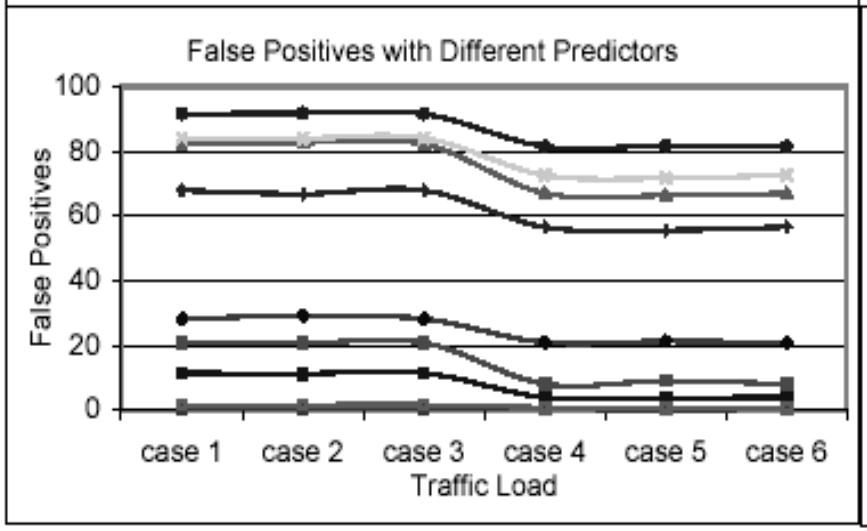
Correlation between RTT and Loss (Cont.)

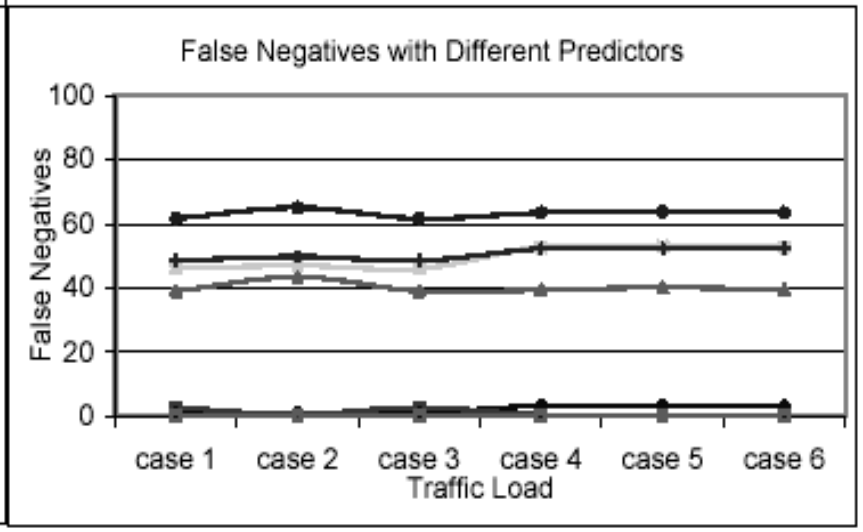


Prediction Efficiency, False Positives & Negatives









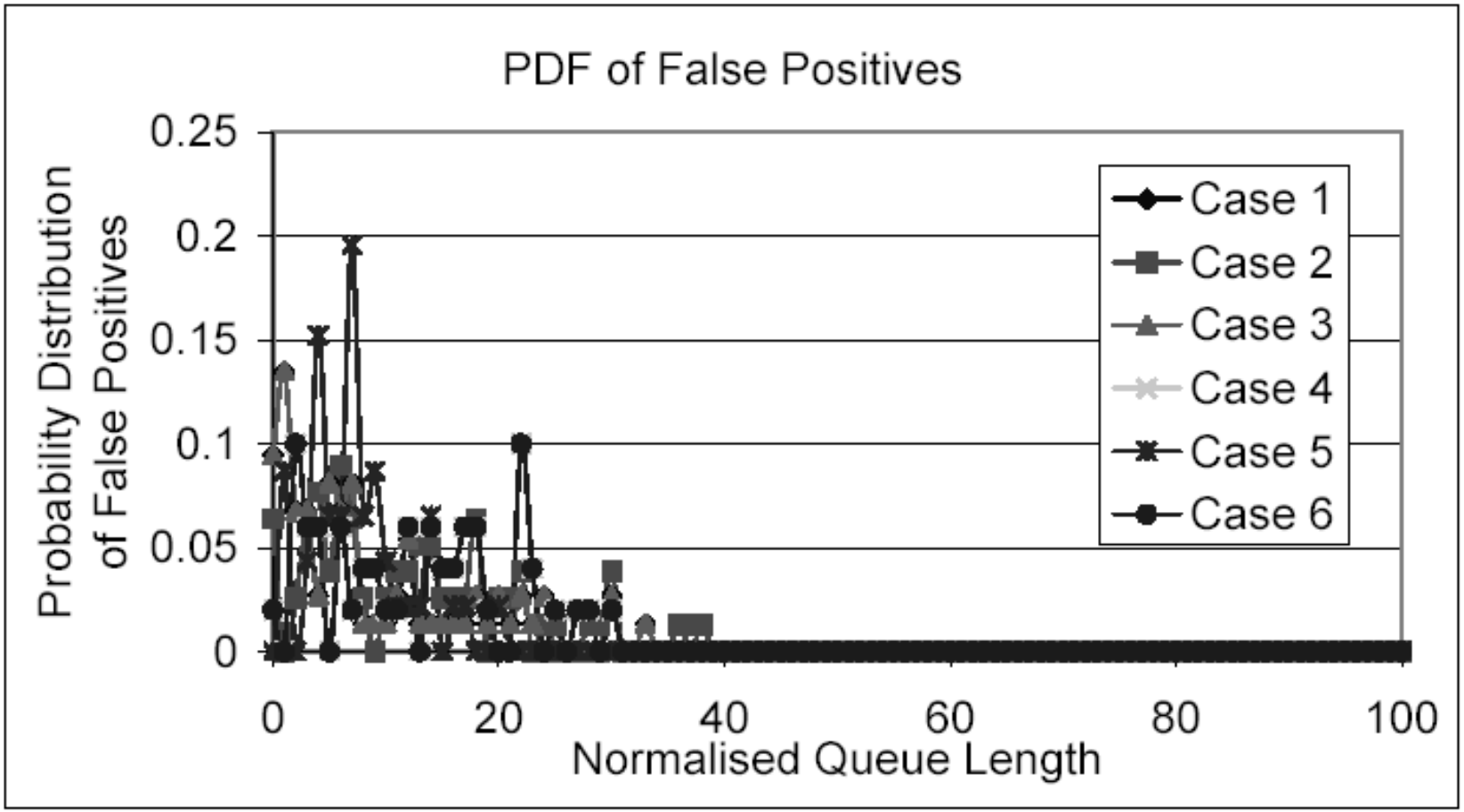
Proactive Congestion Avoidance

- Measurements show
  - End-host congestion avoidance not efficient
  - Responding to uncertain signal can cause more harm than good even at low false positive rates
    - Assume response is multiplicative decrease with factor 0.5
    - Alternate response can be designed to provide robustness to uncertainties

Proactive Congestion Avoidance

- Designing the Response
  - False positives cannot be entirely eliminated
  - Response should be chosen such that impact of false positives can be reduced
    - When to respond ?
    - How to respond ?
    - How much response ?

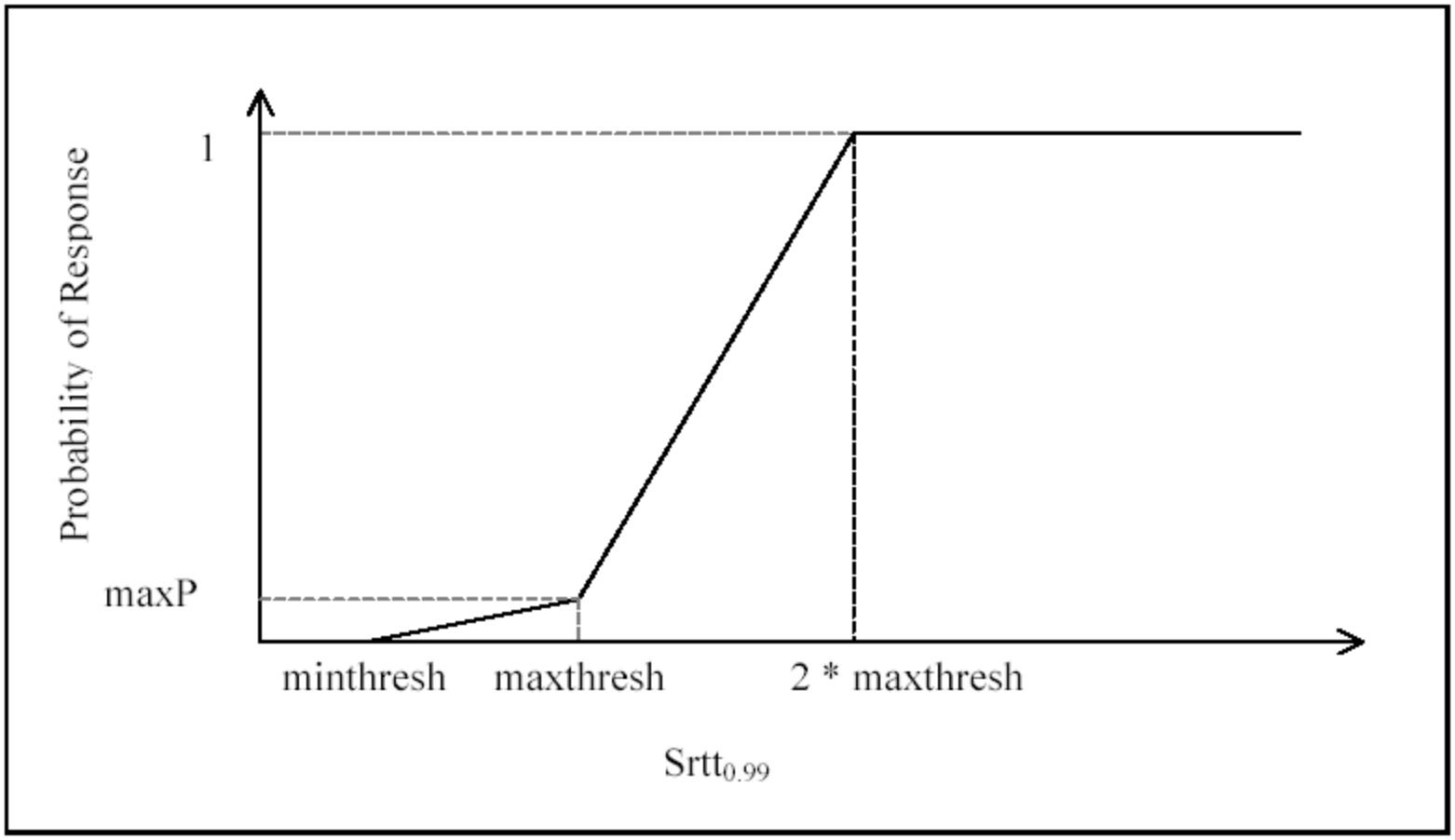
Designing the Probabilistic Response



Proactive Congestion Avoidance

- Designing the Probabilistic Response
  - False positives decrease as queue length increases
  - Make probabilistic response a function of queue length
    - At lower queue length, probability of response is low
    - Probability of response increases as queue length increases
  - Conceptually, similar to RED/ECN
    - Emulate the RED probability curve
    - Use smoothed end-to-end RTT for tracking queue length
    - Possible to emulate other AQM mechanisms also

Probabilistic Response Curve for PERT



Proactive Congestion Avoidance

- Probabilistic Early Response TCP (PERT)
  - Determine the appropriate prediction signal
    - improve the reliability of the prediction
  - Determine the appropriate response function
    - uncertainties in prediction are unavoidable
    - offset this uncertainty by making the response probabilistic
    - different AQM schemes can be emulated in the response - we choose RED/ECN

Proactive Congestion Avoidance

- Prediction signal used in PERT
  - RTT sample collected for every packet
  - Timestamp option used with high clock resolution
  - EWMA smoothing with weight 0.99 for history for eliminating noise
    - High prediction efficiency
    - Low false positives
    - Low false negatives

Proactive Congestion Avoidance

- Response function used in PERT
  - Probabilistic - emulates RED/ECN
  - Fixed values used for parameters
    - minthresh_ = 5ms
    - maxthresh_ = 10ms
    - maxP_ = 0.05
  - Adaptive values possible (similar to adaptive RED)
  - At most one response per RTT

PERT Goals

- Compete with TCP and other loss based protocols
  - Incremental deployability
- High speed networks
  - At low delays, increase window faster
- Wireless Networks
  - Decrease window based on delays
    - Impact of channel losses can be reduced

Basic Approach

$$Increase: w_{t+R} \leftarrow w_t + a; a > 0$$

$$Decrease: w_{t+dt} \leftarrow (1-b)w_t; 0 < b < 1$$

- Change the window increase factor ‘a’
- Change the multiplicative decrease factor
  - For early responses
  - For responses during losses detected by 3 dupacks
- Operate in Phases

Changes to ‘a’

Throughput(TCP): $(\frac{a}{b})^{1/(2)} * \frac{1}{Rp^{1/(2)}}$

Throughput (PERT): $(\frac{a'}{b'})^{1/(2)} * \frac{1}{R(p+p')^{1/(2)}}$

Drop Rate

Early Response Rate

For PERT to get equal share when competing with TCP:

Throughput(TCP) = Throughput (PERT)

$$(\frac{a}{b})^{1/(2)} * \frac{1}{Rp^{1/(2)}} = (\frac{a'}{b'})^{1/(2)} * \frac{1}{R(p+p')^{1/(2)}}$$

$$(\frac{a}{b})^{1/(2)} * \frac{1}{\quad} = (\frac{a'}{\quad})^{1/(2)} * \frac{1}{\quad}$$

a =1 for TCP

$$(\frac{a'}{b'}) * b = \frac{p+p'}{p}$$

$$a' = 1 + \frac{p'}{p}$$

Conservatively ß=ß’

Changes to ‘ß’

Bottleneck Buffer when each flow reduces its window by a factor ‘f’

$$\rightarrow B \geq \frac{f}{1-f} * BDP$$

We call ‘f’, ß here! Therefore, we have

$$curq \geq \frac{b}{1-b} * maxq$$

Playing with this equation, we get

$$b \leq \frac{curq}{curq + maxq}$$

$$\text{We Set} \quad ß = \frac{curq}{curq + maxq}$$

Proactive Congestion Avoidance

- Response function used in PERT (cont.)
  - Multiplicative decrease is tied to observed delay

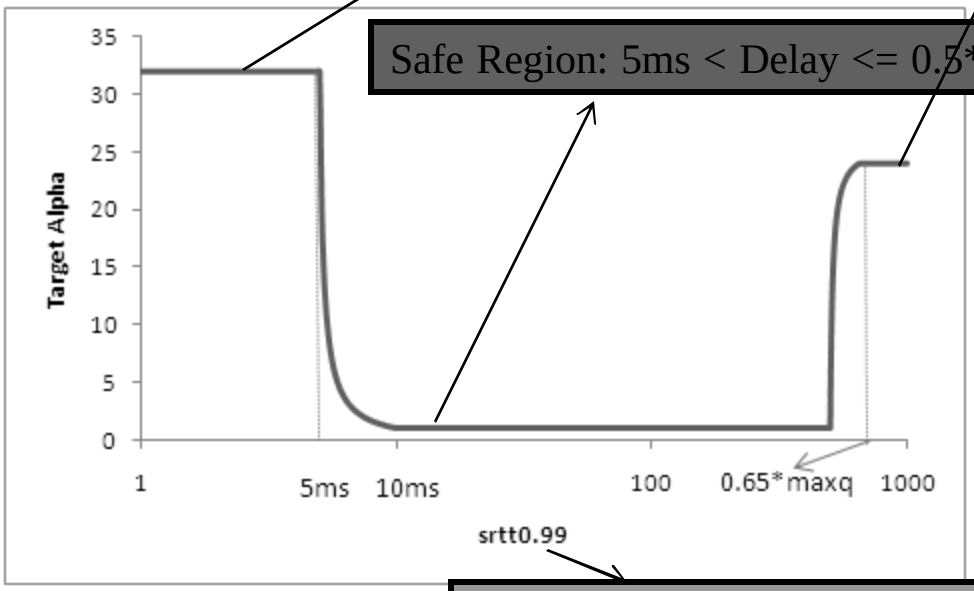
$$\text{We Set} \quad ß = \frac{curq}{curq + maxq}$$

  - At lower queuing delays, reduce window by small amount
    - Automatically tolerates channel errors, without having to distinguish them from congestion losses

Operation in phases

PERT needs to perform well in heterogeneous environments!

- Achieved By operating in Phases



Tuning ‘a’

- The actual ‘a’ is increased additively and decreased multiplicatively.
  - a = a + 0.5 (High speed region)
  - a = 0.9*a (Safe/Moderate region)
  - a = a + 0.1 (Compete region)
- a = Min(a, 32)
  - a values could be larger in the Compete region
  - May result in a large burst of packets
    - Increased drop rates
    - Increased queue lengths

Implementation Issues

- Estimation of p and p’
  - TFRC style estimation of loss rates
  - Store number of packets between consecutive packet losses
  - Give different weights to the historical data based on how recent it is

Mean number of packets between consecutive losses :

$$\frac{(n1+n2+n3+n4+0.8*n5+0.6*n6+0.4*n7+0.2*n8)}{6}$$

Therefore, packet loss rate can be written as:

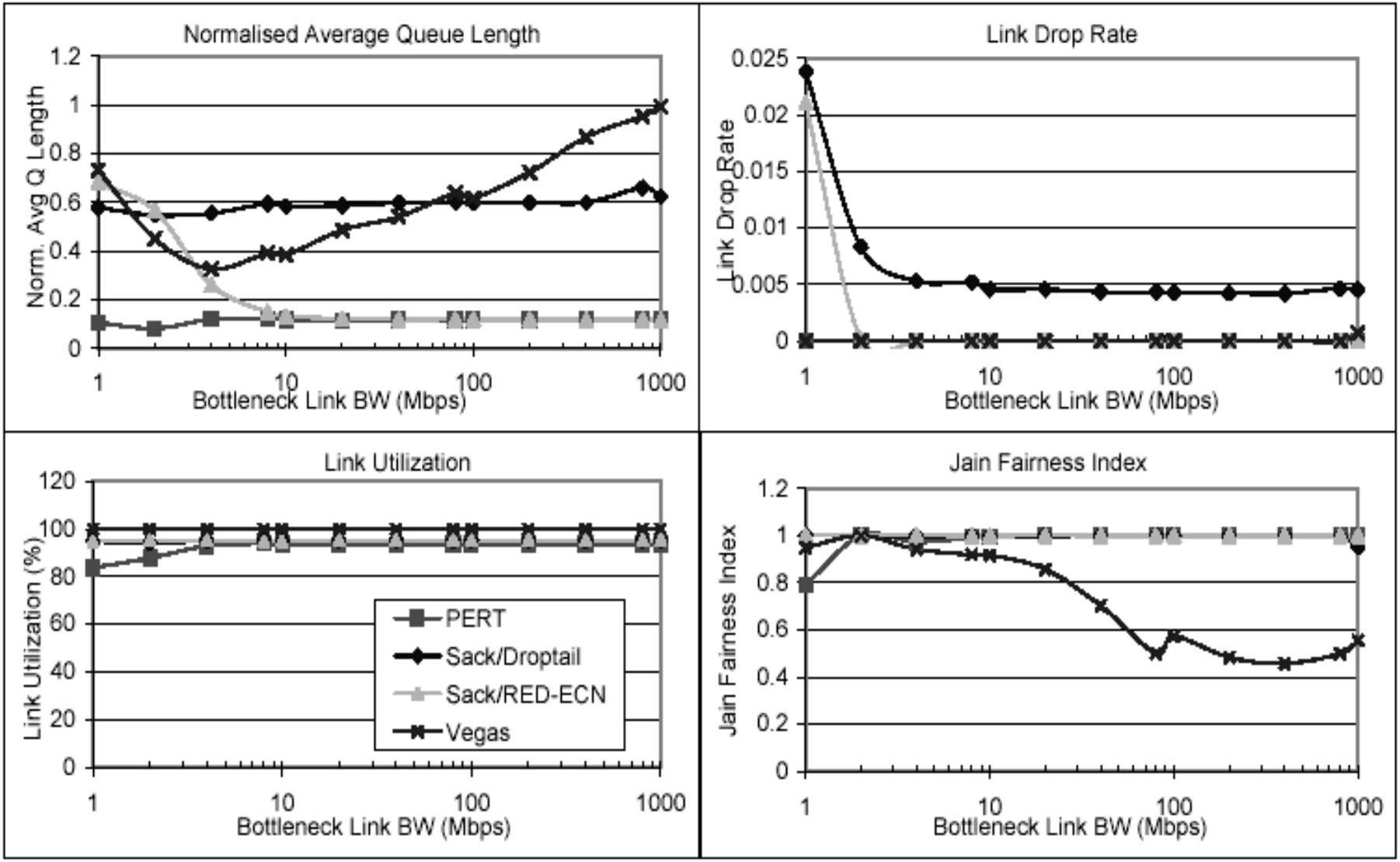
$$\frac{6}{(n1+n2+n3+n4+0.8*n5+0.6*n6+0.4*n7+0.2*n8)}$$

Evaluations

- Ns-2 based Evaluations
- Linux Kernel based Evaluations

PERT : Homogenous network

Varying the Bottleneck Link Bandwidth

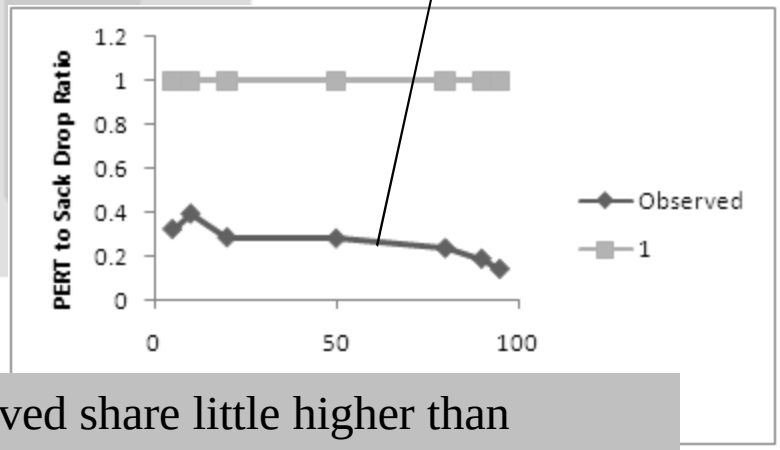


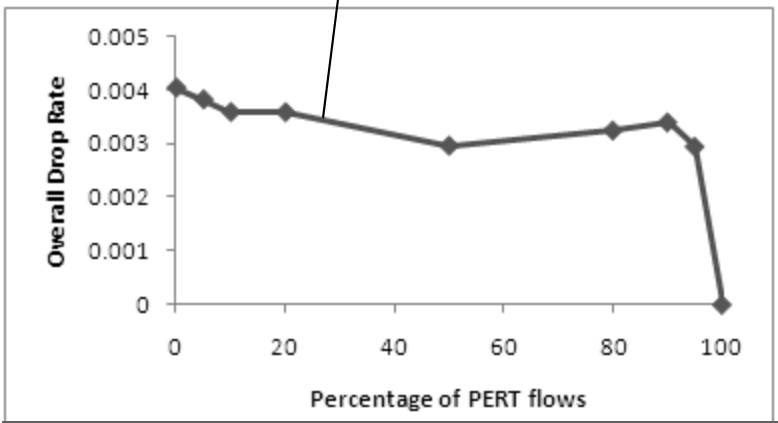
Heterogenous network

PERT observes smaller losses than SACK

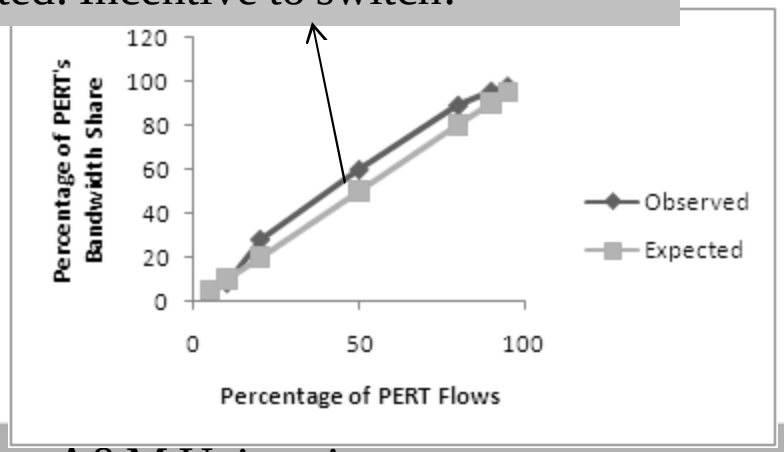
Smaller losses in the mix compared to 100% SACK

Varying Mix (PERT-SACK):

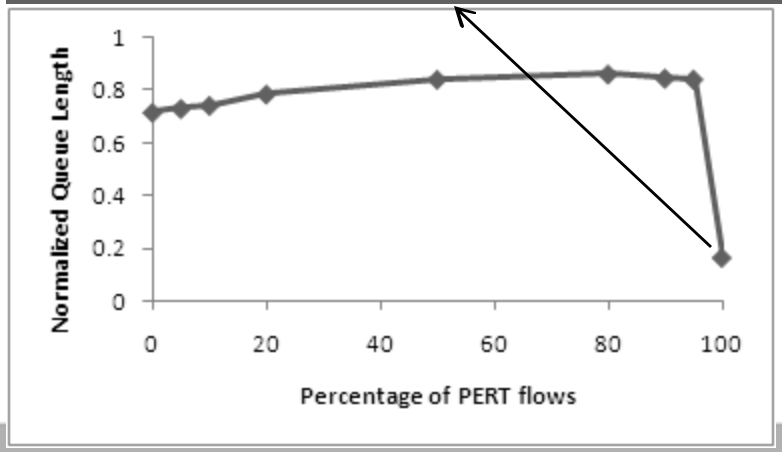




Observed share little higher than expected: Incentive to switch!



Negligible queue length at 100% PERT

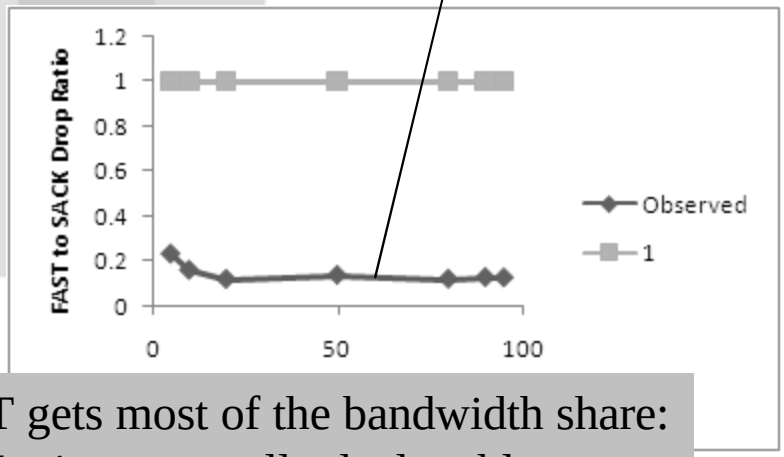


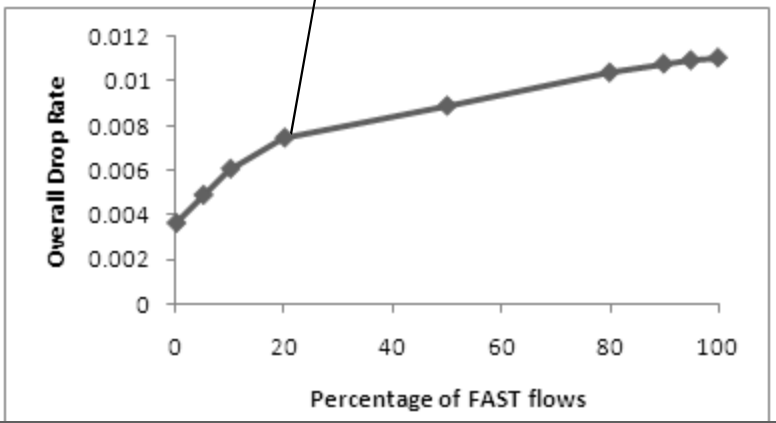
Ns-2 Based Evaluation

FAST observes smaller losses in the mix

Loss rate increases as the mix goes towards 100% FAST

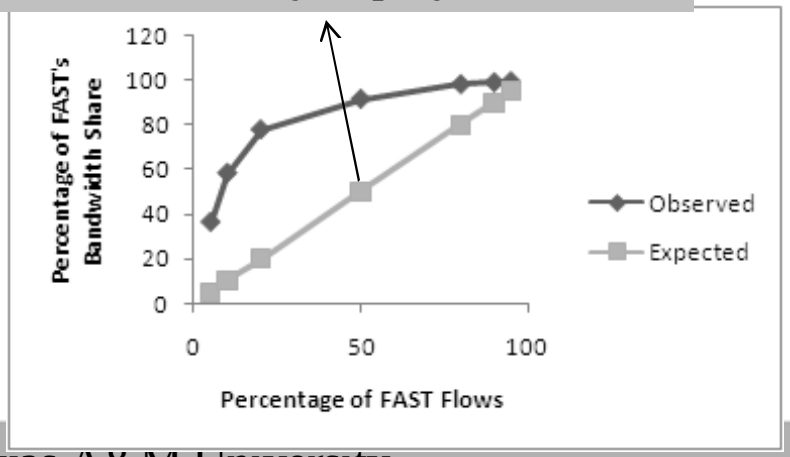
Varying Mix (SACK-FAST):

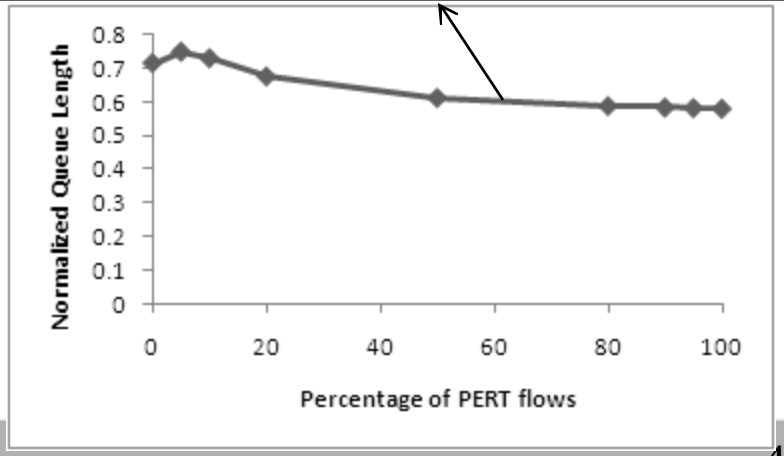




FAST gets most of the bandwidth share: Not incrementally deployable

Queue does not decrease much

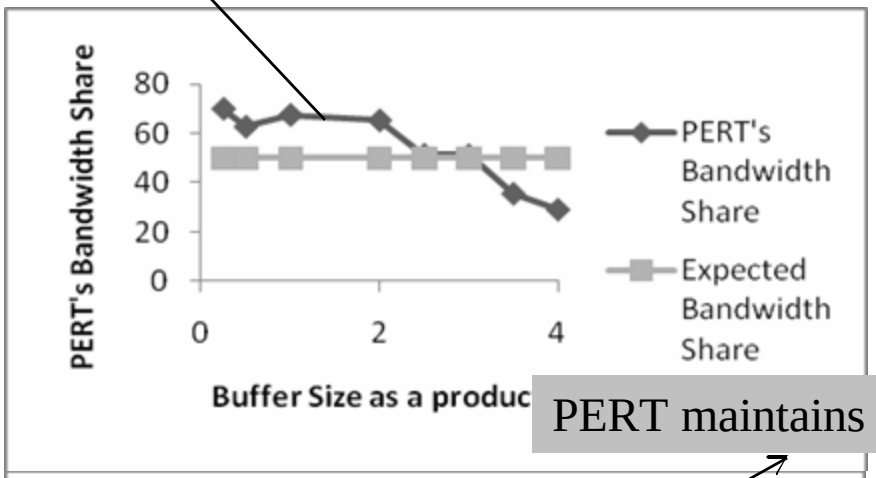




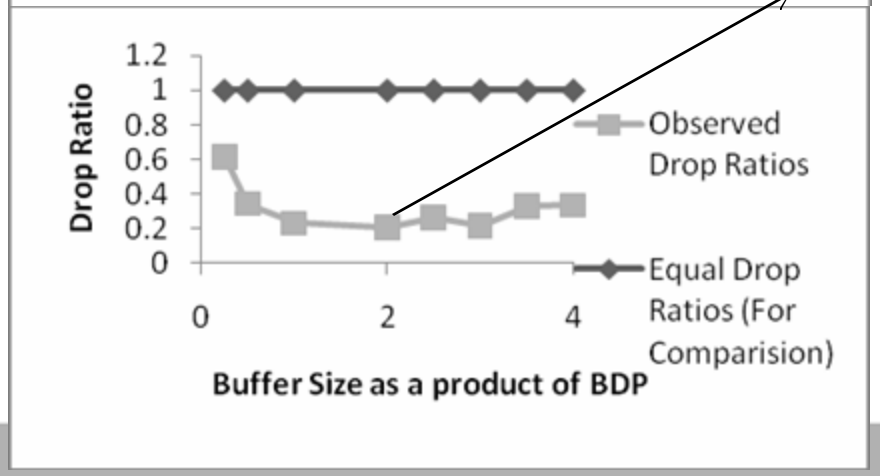
Ns-2 Based Evaluation

Better performance at lower buffers

Variation of buffer size in a 50-50 mix (PERT-SACK)



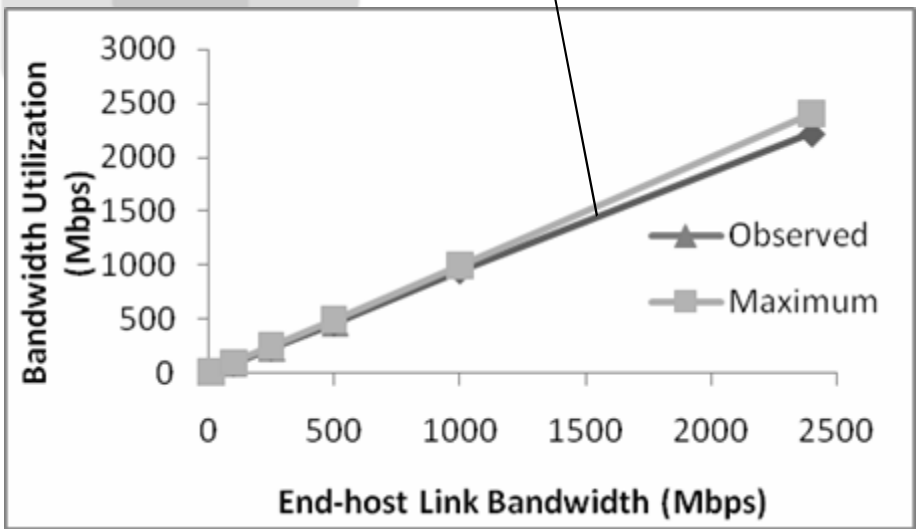
PERT maintains smaller losses throughout



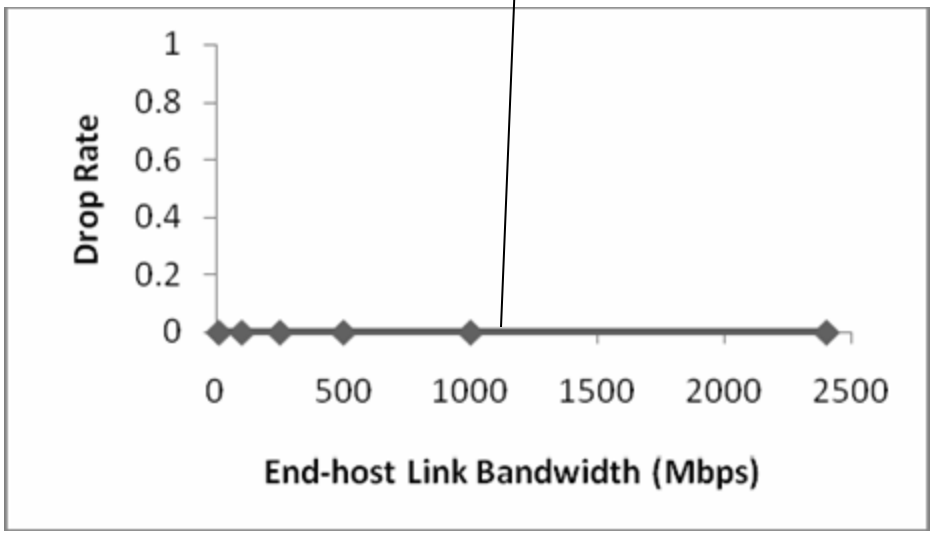
Homogeneous Environments

Performance in High-speed networks (Single flow):

PERT utilizes near-full bandwidth



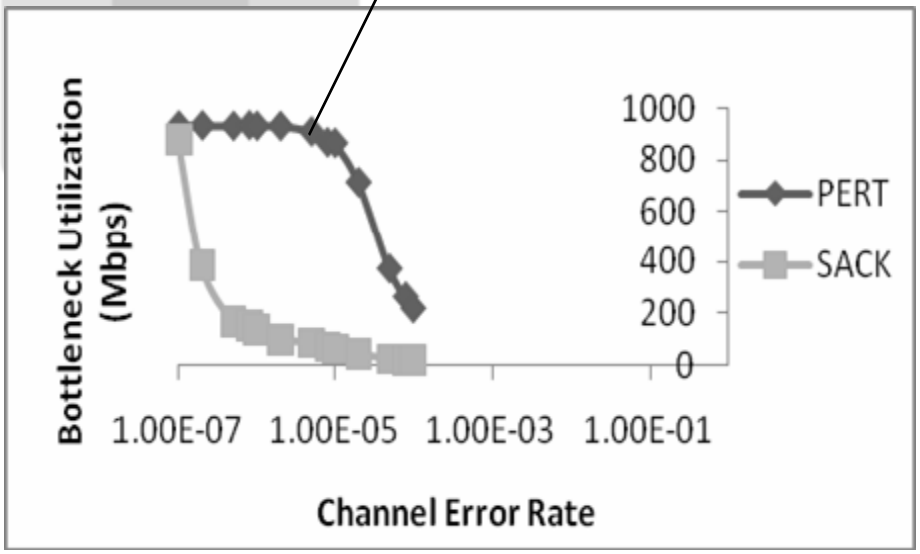
Zero losses throughout



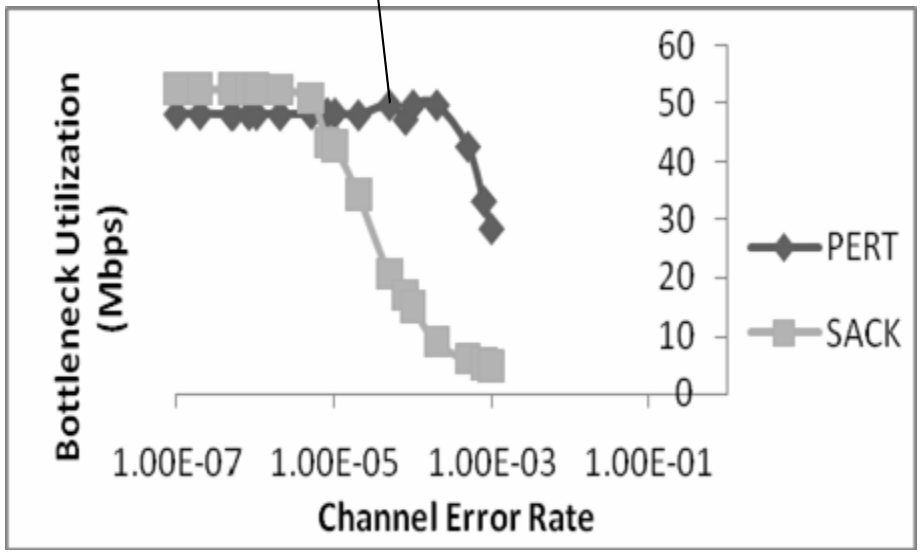
Homogeneous Environments

Robustness to Channel Errors:

High speed link: utilizes link better

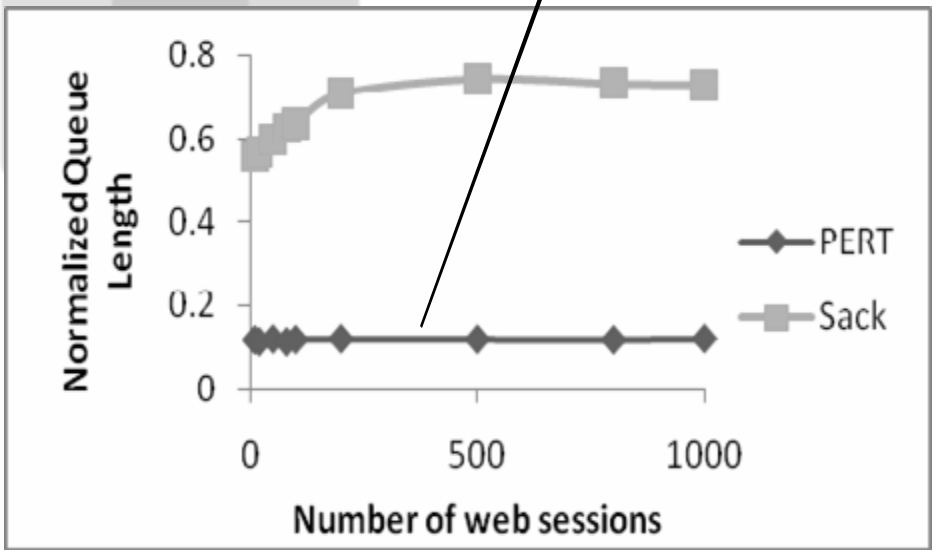


55 Mbps wireless bottleneck: utilizes better than SACK

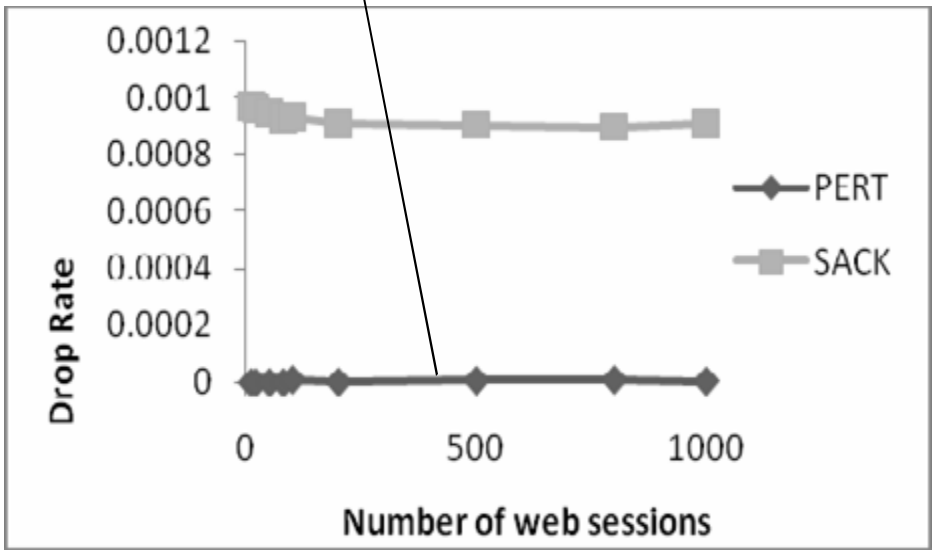


Impact of web traffic

PERT observes negligible queue lengths

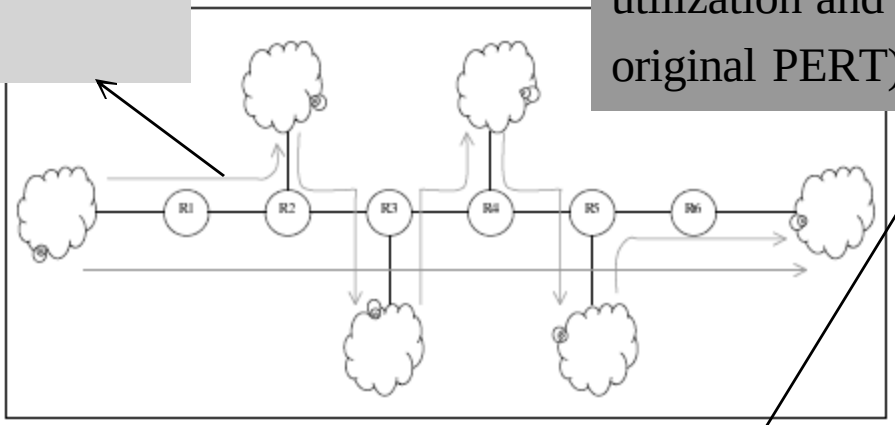


Near zero losses throughout



Multiple bottlenecks

Multiple Bottleneck scenario: 20 clients in each cloud

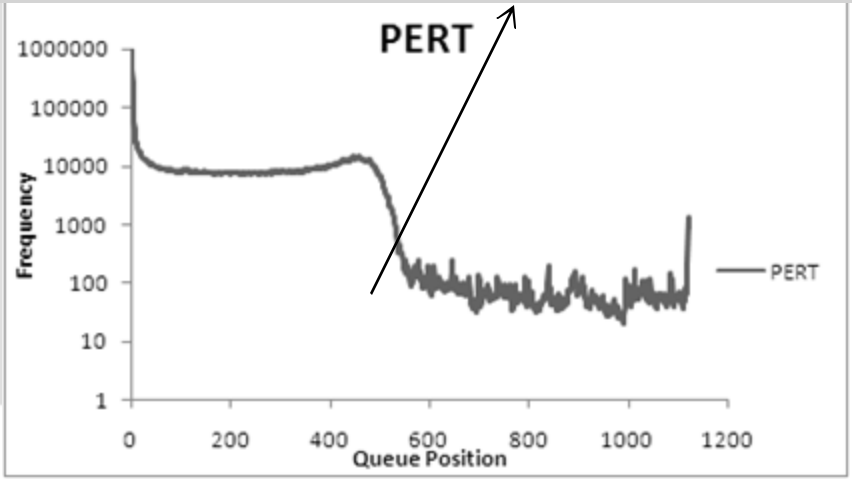


Negligible queue lengths, good utilization and fairness (similar to original PERT)

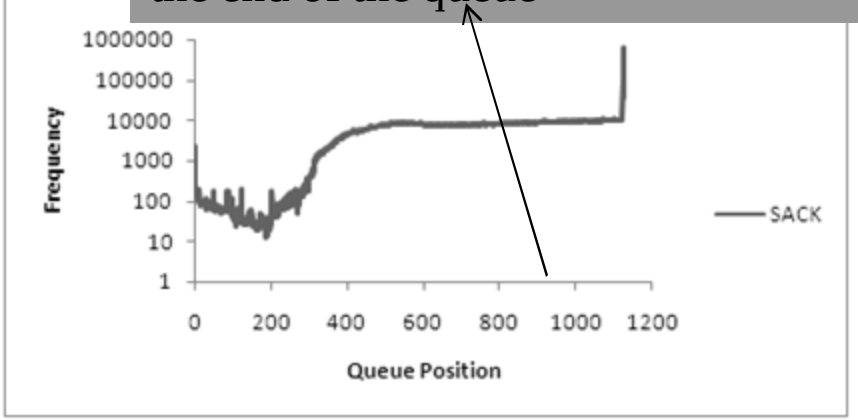
Link	drop rate	Normalized queue length	Utilization(Mbps)	Jain's Fairness Index
R1-R2	0	0.172138	95.2394	0.999875
R2-R3	0	0.168135	95.238	0.999872
R3-R4	0	0.17454	95.2374	0.969581
R4-R5	0	0.172938	95.2385	0.999889
R5-R6	0	0.164932	95.2385	0.99986

Queue lengths Comparison

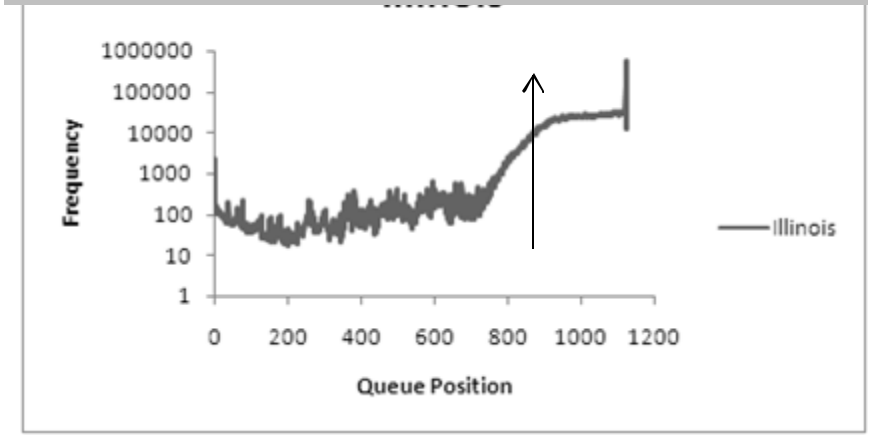
PERT enqueues most of the packet at the lower end



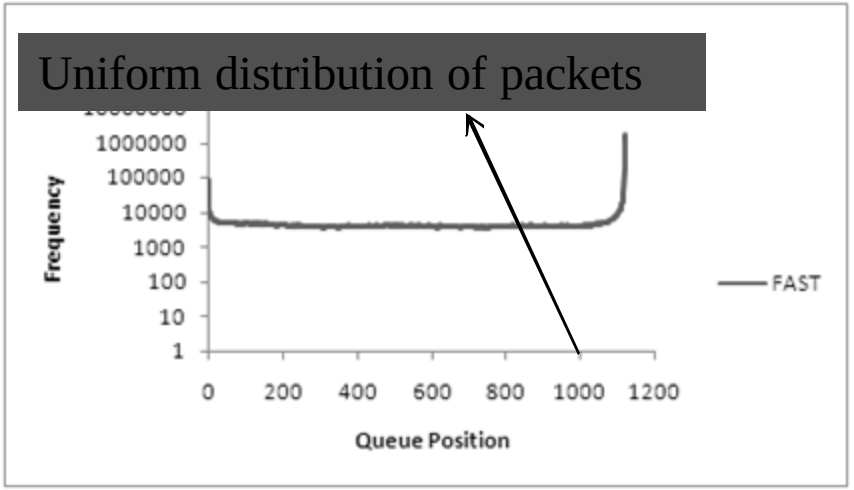
SACK enqueues most of the packets at the end of the queue



Performance worse than SACK



Uniform distribution of packets



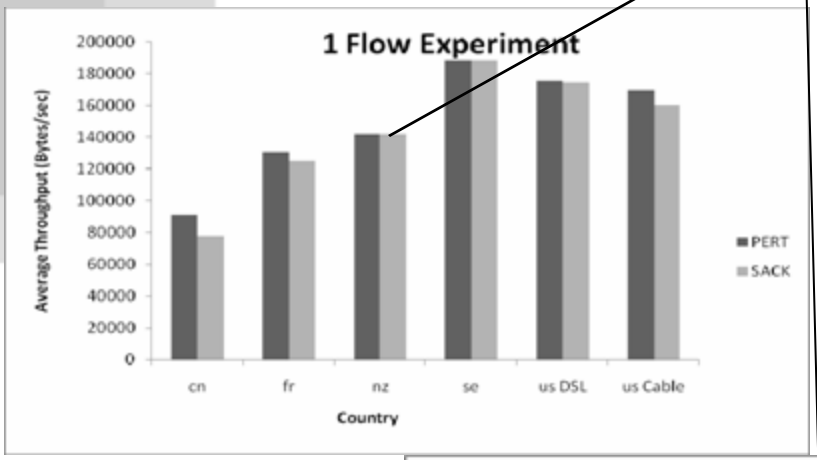
Linux Emulations

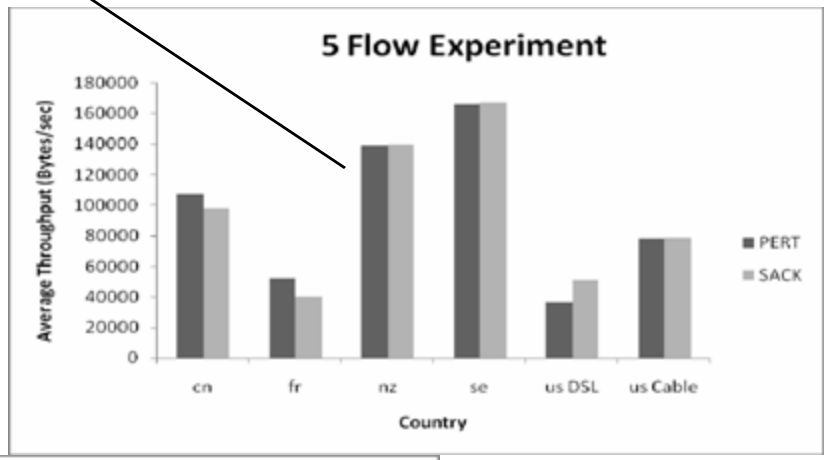
- Client at diverse locations spread across the world. Two more clients (DSL and Cable modem based) were also considered
- Two machines in our lab in the same subnet act as servers
- Equal number of PERT and SACK flows compete for bandwidth

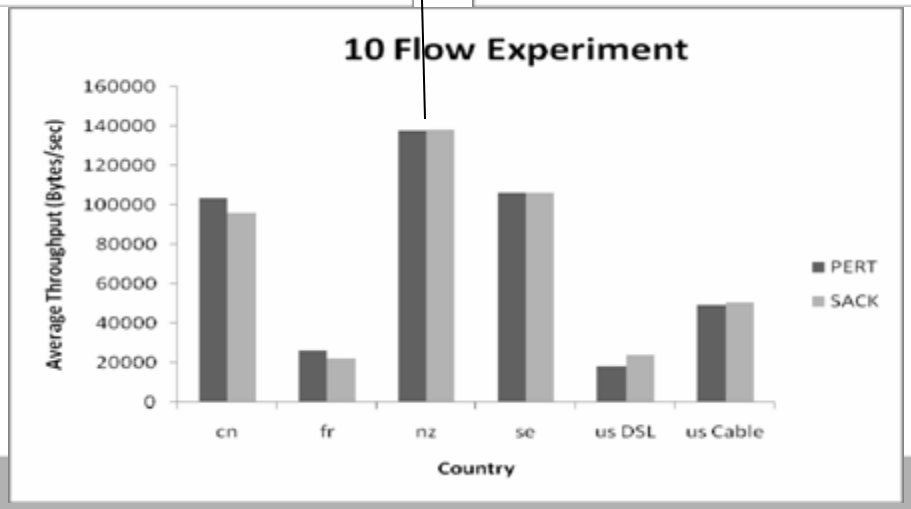
Linux Emulations

PERT observes higher throughput most of the time

- Throughput Comparison: PERT-SACK



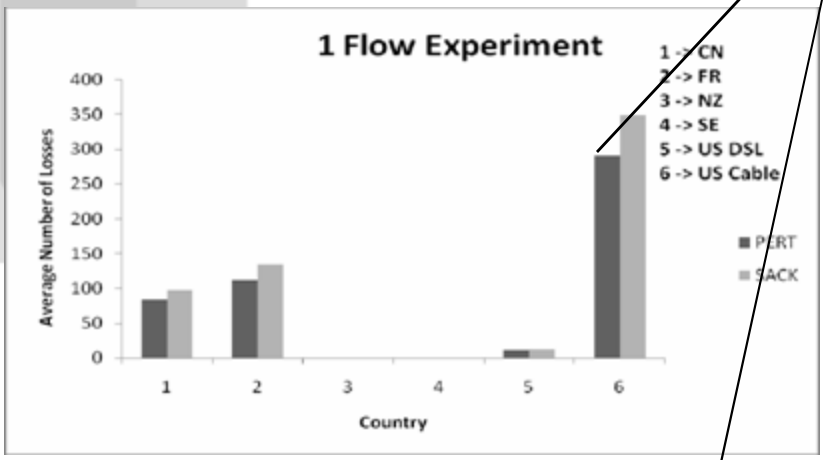


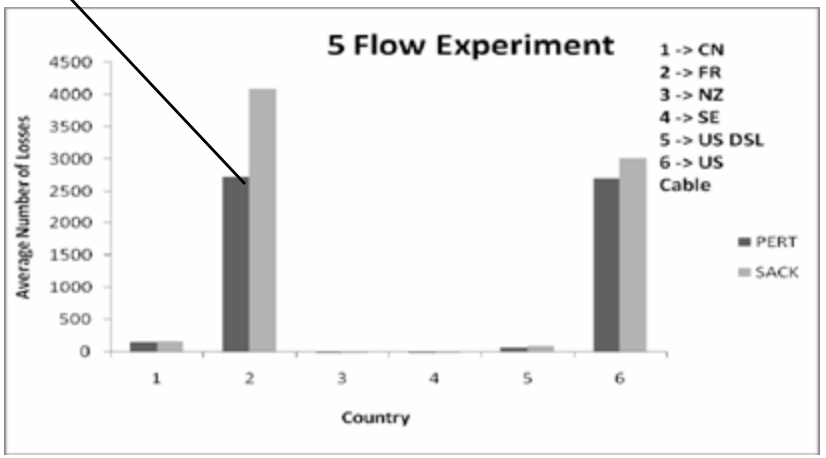


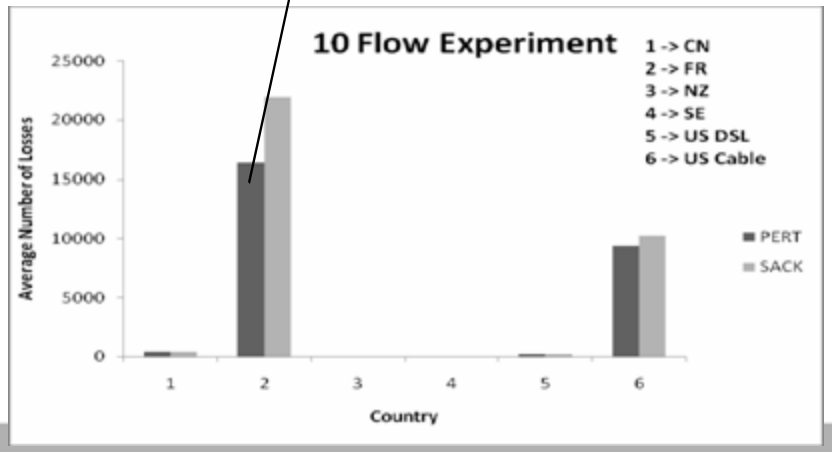
Linux Emulations

PERT sees less number of losses

- Number of losses PERT-SACK



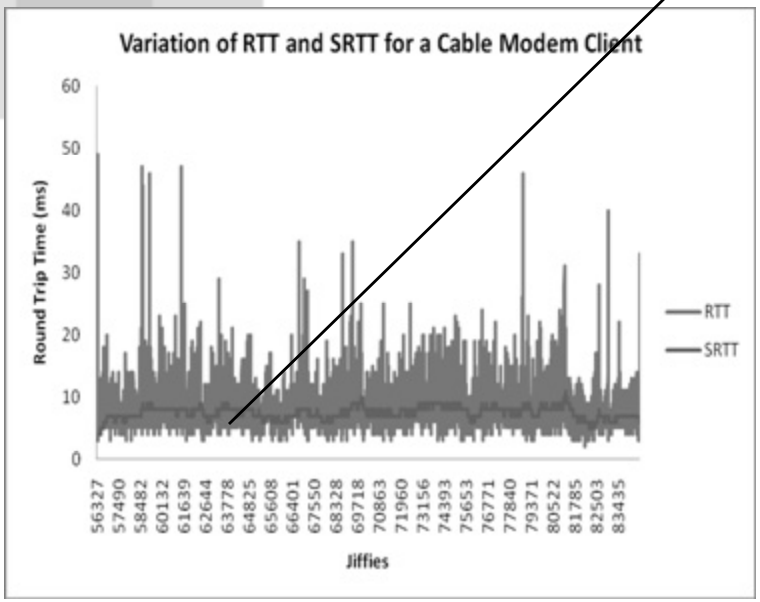


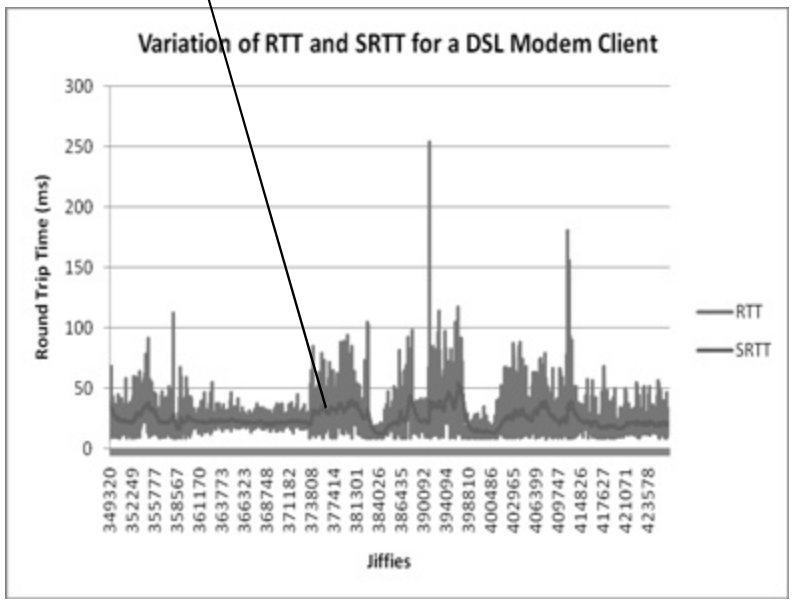


Linux Emulations

Smoothen RTT signal does not vary much despite highly varying RTTs

RTT and SRTT measurements for Cable and DSL modem clients





Summary of PERT

- PERT can compete with TCP flavors like SACK.
- Can be deployed Incrementally by providing incentives to switch.
- Single flow of PERT utilizes a high speed link well with negligible drops.
- Gracefully degrades with channel errors better than SACK.
- Works well with real-world traffic.

Single Congestion protocol in the future?

- No Single protocol likely to dominate in the future
- Linux uses BIC/CUBIC as default
- Microsoft shipping CTCP
- Many other TCP flavors available in Linux
  - Users can choose one
- Proliferation of many CC protocols
- What impact does this have?

Multiple CC protocols

Protocol	Bandwidth Share	Drop rate
SACK	34.5888	0.00275
Illinois	21.3449	0.00311
PERT	44.0663	0.00228

- SACK, TCP-Illinois, and PERT nearly fair

Multiple CC protocols

Protocol	Bandwidth Share	Drop rate
FAST	82.5992	0.00326
SACK	6.8721	0.0242
Illinois	6.5698	0.02499
PERT	3.9588	0.03598

- FAST nearly kills other competing flows

Multiple CC protocols

- Can a protocol outcompete others for bandwidth?
  - Yes, very possible
- Why don't we see the impact of these outcomes?
  - Congestion at the end hosts or last mile
  - Individual hosts throttled at access points and not in the network core.
  - With faster access links, outcomes will be different
- How should network control the interaction of different protocols?

Multiple CC protocols

- Network Control of different protocols?
- Fair queuing
  - Flow isolation
- Pricing based on origination of congestion
  - Being discussed at IETF
- Determine Safe protocols
  - How to define Safe?
  - Is it enough to respond to congestion?
  - Is it enough to allow TCP to get some BW?

Multiple CC protocols

- Different CC protocols being deployed in data centers
  - Low RTTs, small buffer switches
- Impact of increased use of video?
  - Favors protocols that employ proactive congestion avoidance

Summary

- TCP dominant CC protocol for a long time
- Many new protocols addressing TCP's weaknesses
- Several with considerable deployment
- Many CC algorithms in the future
- Diversity is being masked at the edges now
- Will need network mechanisms for diversity to coexist in the future