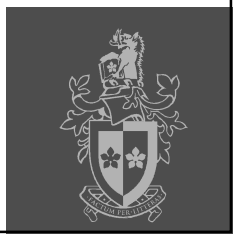
SWINBURNE

CENTRE FOR ADVANCED INTERNET ARCHITECTURES

Understanding TCP Parallelization

Qiang Fu

qfu@swin.edu.au



Outline

- TCP Performance Issues
- TCP Enhancements
- TCP Parallelization (research areas of interest)
  - Related Approaches
  - TCP Parallelization vs. Single Connection Based Approach
  - Active Queue Management
  - Satellite/Terrestrial Networks
  - Wireless/Mobile Networks
  - High Speed Networks
  - End-to-End Service Differentiation
  - Effectiveness vs. Fairness
- Modelling
- Simulation Results
- A Delay-Based Approach
- Conclusions

TCP Performance Issues

- Application performance perceived by end users largely depends on TCP performance
- Future Internet: fast links, diversity in network access technologies
- One TCP for all or specific TCPs for specific networks

TCP Performance Issues

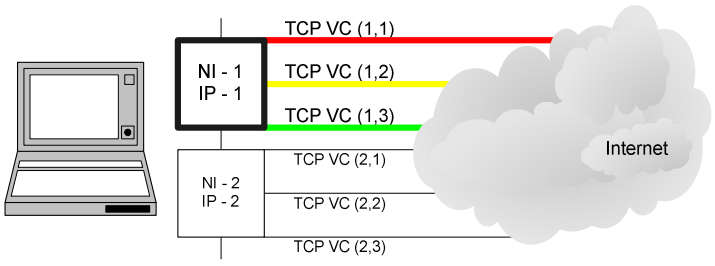
- Unnecessary timeouts and congestion control
  - Impossible to design **retransmit timeout** algorithms that never result in an **unnecessary timeout**
  - Impossible to design **fast retransmit** algorithms that always correctly determine whether or not a packet loss has occurred (**unnecessary congestion control**)
  - Difficulty in distinguishing between congestion and corruption
  - Informative ACKs/notifications: lost themselves, overhead, network asymmetry
- Conservative AIMD algorithm (Additive Increase Multiplicative Decrease)
  - More aggressive AI (more than 1 segment per RTT)
  - Milder MD (less than W/2)
  - Effectiveness up, fairness down
- Network asymmetry, service differentiation, active queue management (AQM), etc.

TCP Enhancements

- Existing TCP Congestion Control Mechanisms
  - Tahoe, Reno, New Reno, Sack, Vegas
- Enhancements for wireless/mobile computing
  - End-to-end solutions: Freeze-TCP, TCP-Probing, TCP Santa Cruz, TCP-Real, Explicit Loss Notification (ELN), Explicit Congestion Notification (ECN), etc.
  - Proxy-Based Solutions: Indirect-TCP (I-TCP), MTCP, Explicit Bad State Notification (EBSN), WTCP, Snoop, etc.
- Active Queue Management (AQM)
  - RED, REM, BLUE, ECN, XCP
- File/data striping
  - MFTP, XFTP, GridFTP, PSockets, etc.
- Enhancements for high bandwidth-delay product (BDP) networks
  - High-Speed, Scalable, FAST
- The TCP that we need
  - Robust across a wide range of environments rather than fine-tuned for a particular environment
  - Take all the problems into account with lowest cost, complexity and inflexibility
  - Ease the impact of the problems rather than solve them

TCP Parallelization (TP)

- Splitting a single TCP connection into a number of parallel virtual connections (VCs)



- VCs could be established through multiple network interfaces
- The VCs could be standard or modified TCP connections
- A single TCP control block (TCB) or individual TCBs for VCs
- The windows of VCs could be centrally or individually controlled

TP: Related Approaches

- File/data striping over multiple TCP connections
  - MFTP, XFTP
    - Overcome the 64KB limit on window size
    - Satellite networks
  - GridFTP, PSockets
    - Increase throughput
    - Data intensive applications
- Single connection based approach
  - MulTCP
    - Emulate the behaviour of a set of multiple standard TCP connections
    - End-to-End service differentiation
  - High-Speed, Scalable, FAST
    - (sort of) Emulate the behaviour of multiple TCP (Reno, Vegas) connections
    - High bandwidth-delay product (BDP) networks
- CM (Connection Manager), Ensemble TCP, Fractional Method
  - Use a set of parallel TCP connections to emulate a single TCP connection
- SCTP (Stream Control Transmission Protocol)
  - Multi-streaming designed to solve the head-of-line blocking problem
  - A single TCB based window control to emulate a standard TCP

TP: TP vs. Single Connection Based Approach

- Advantages
  - Localize disturbances such as timeouts, packet losses, out-of-order packets
  - Disperse packet losses
  - Reduce head-of-line blocking by independent sequencing for VCs
  - Reduce the impact of fast recovery
  - Introduce “self-curing”
  - Easier to maintain several smaller/less aggressive windows than a single large / aggressive window
  - Less bursty traffic
- Disadvantages
  - Complexity and inflexibility, fairness concern

TP: Active Queue Management

- Interaction with AQM (RED, ECN)
- AQM reduces unnecessary packet losses and smoothes traffic by marking or dropping packets
  - Misjudgement on dropping and marking
- TCP parallelization reduces the burstiness of packet arrivals
- AQM may allow TCP:
  - More aggressive to increase window (more than 1 segment per RTT)
  - Milder to reduce window (less than W/2)

TP: Satellite/Terrestrial Networks

- Demand on the integration of satellite and terrestrial networks
  - “Out-of-area” coverage
  - Load switching
- Long propagation delay and high error rate
  - Significant performance improvements
  - TCP parallelization with Reno/New Reno/SACK can outperform single connection based SACK
- Network asymmetry
  - Downstream bandwidth could be hundreds times upstream bandwidth
  - Delayed/lost ACKs over upstream channel could either create bursts of packets or disrupt window growth
  - Reducing the no. of ACKs at the risk of destroying the self-clocking of TCP and unnecessary retransmit timeouts
  - Reducing ACK size: not good for informative ACKs

TP: Wireless/Mobile Networks

- Wireless transmission errors, route failures, medium access contention, contention between TCP data and ACKs
- Forward Error Correction (FEC)
  - Small FEC gives the most efficient throughput gain
  - Large FEC gives the maximum throughput
  - Small FEC with TCP parallelization
- Bit interleaving
  - Random losses
- Bad link state
  - Transient (random interference)
  - Persistent (mobility, link failure)
- Unreliable estimated upper bound on RTT in the wireless environments, persistent reordering of packets (especially cellular systems)

TP: High Speed Networks

- The traditional AIMD algorithm is too conservative for high BDP networks
- Tremendous cost for unnecessary congestion control and timeouts
- Difficult and costly to maintain a single large/aggressive window
- Multiple small/mild windows can reduce costs and improve performance

TP: End-to-End Service Differentiation

- Inter-domain issues for Weighted Fair Queuing (WFQ), Class Based Queuing (CBQ)
- MulTCP: TCP level service differentiation
  - Poor performance
  - Rely on SACK and AQM
- TCP parallelization: better performance, even better with AQM

TP: Effectiveness vs. Fairness

- Effectiveness up, fairness down
- Balance between effectiveness and fairness
- Decoupling effectiveness and fairness
- AQM support to eliminate persistent congestion and give better RTT estimation
- Queue stabilizing
- Optimising effectiveness and fairness factors

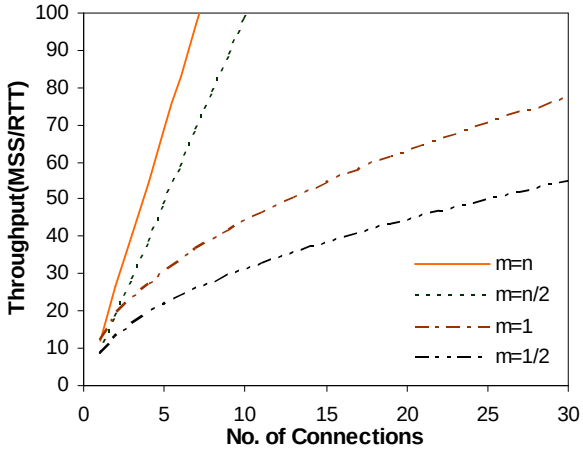
Modelling

- An Analytical Model for TCP parallelization (Fast recovery not considered)

$$BW_n = \sqrt{\frac{m(2cn-1)}{2p}} \times \frac{MSS}{RTT}$$

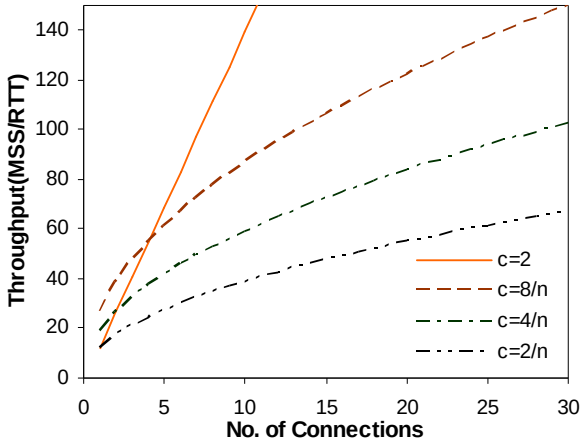
BW_n:	estimated throughput of a set of *n* parallel connections
MSS:	Maximum Segment Size (fixed packet size)
RTT:	Round-Trip Time
n:	number of parallel connections
m:	window increase factor (the aggregate window increases *m* packets per *RTT*)
c:	window decrease factor (the individual window cuts by 1/*c*, in response to a packet loss)
p:	packet loss rate

Modelling



Throughput vs. various *m* schemes (*c*=2, *p*=0.01)

Modelling



Throughput vs. various *c* schemes (*m*=*n*, *p*=0.01)

Modelling

- Another Analytical Model for TCP parallelization (Fast recovery considered)

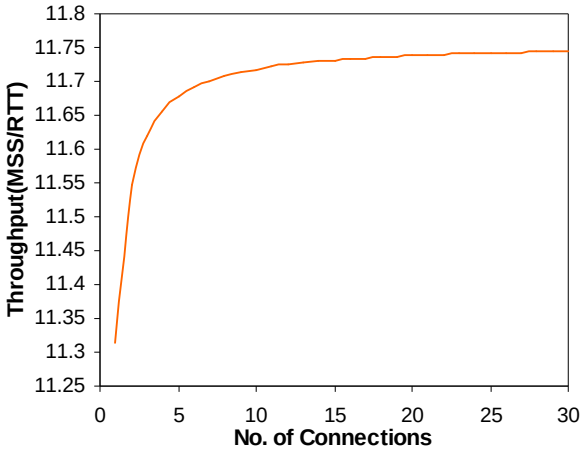
$$\left[2\frac{(cn-1)W_n}{cn}+\frac{m(n_e-1)}{n_e}+\frac{W_n}{cn}\right]\times\left(\frac{W_n}{cmn}-\frac{n_e-1}{n_e}\right)\times\frac{1}{2}+$$

$$\left[2\frac{(cn-1)W_n}{cn}+\frac{m(n_e-1)}{n_e}\right]\times 1\times\frac{1}{2}=\frac{1}{p}$$

$$BW_n=\frac{(1/p)*MSS}{\left\{1+\left(\frac{W_n}{cmn}-\frac{n_e-1}{n_e}\right)\right\}*RTT}$$

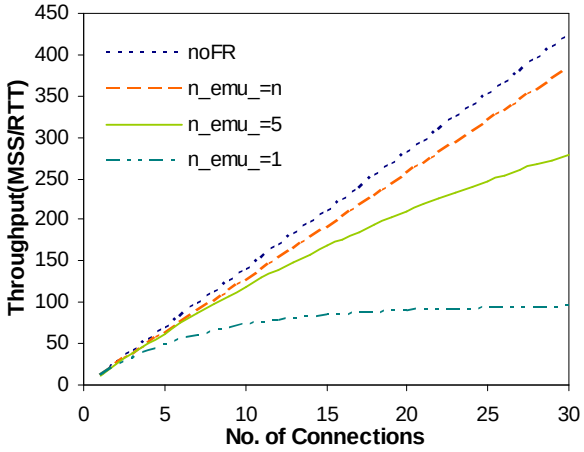
***n*$_e$**: actual number of connections used to emulate the behaviour of a set of *n* parallel connections

Modelling



Emulating a single standard TCP connection, p=0.01

Modelling

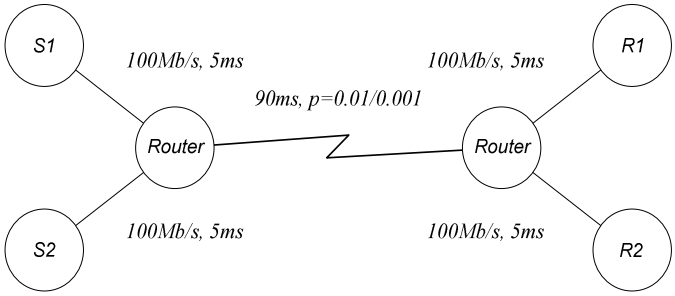


Emulating a set of parallel TCP connections, p=0.01

$n\_emu\_=n_e$

Simulation Results (SR)

- Simulation Topology



Simulation Results (SR)

- Simulation Parameters

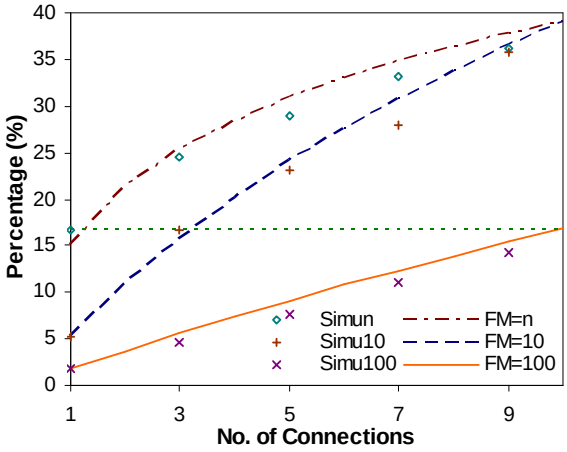
TCP Scheme	*Reno/Newreno/Sack*
Bottleneck Capacity	28.6/8Mbps
Link Delay	100ms (5+90+5)
Packet Size (MSS)	1,000bytes
Buffer Size	Unlimited/Limited
Error Distribution (*p*)	Uniform/Two-state Markov
Background Traffic	TCP/UDP Traffic
Simulation Time	1,000s

Simulation Results (SR)

- Fractional Method
  - Fractional Approach
    - The aggregate window of a set of *n* connections increases 1 packet per RTT (e.g., *m*=1)
    - In response to a packet loss, the involved individual window is halved (e.g., *c*=2)
    - *FM*=*n*/*m*=*n* (*FM*: fractional multiplier)
  - Combined Approach
    - A standard TCP connection + a set of *n* parallel connections
    - The aggregate window of the set of *n* connections increases 1/*FM_C* packets per RTT (e.g., *m*=1/*FM_C* | *FM_C*>=1, *FM_C*: combined fractional multiplier)
    - In response to a packet loss, the involved individual window is halved (e.g., *c*=2)

SR: Fractional Method

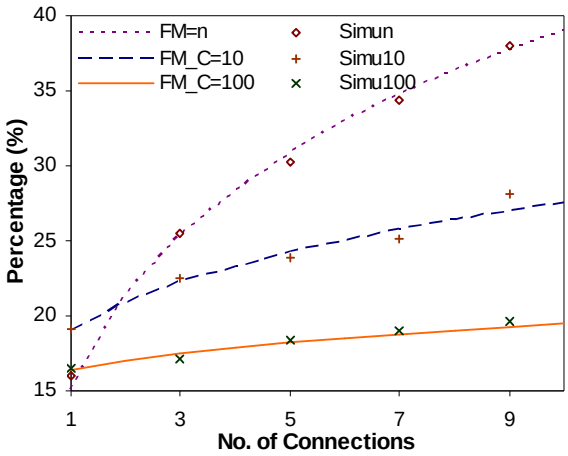
- Uniform Errors



Throughput share (*p*=0.001)

SR: Fractional Method

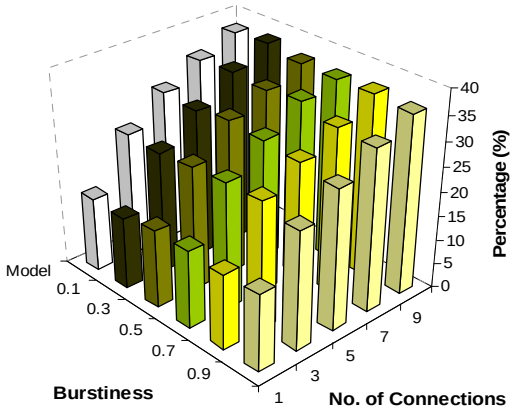
- Uniform Errors



Throughput share (p=0.01)

SR: Fractional Method

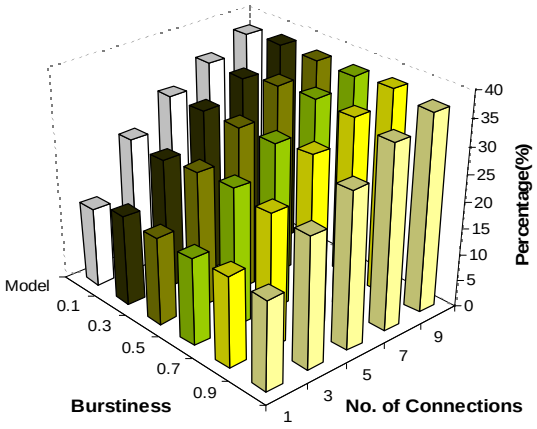
- Bursty Errors



Throughput share vs. Burstiness (0.001)

SR: Fractional Method

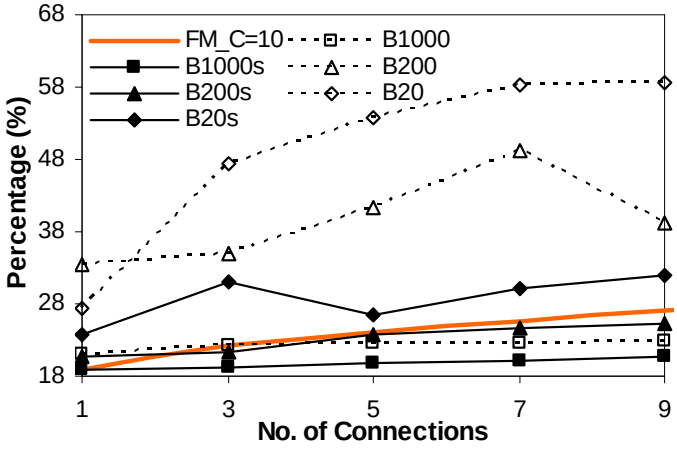
- Bursty Errors



Throughput share vs. Burstiness (0.01)

SR: Fractional Method

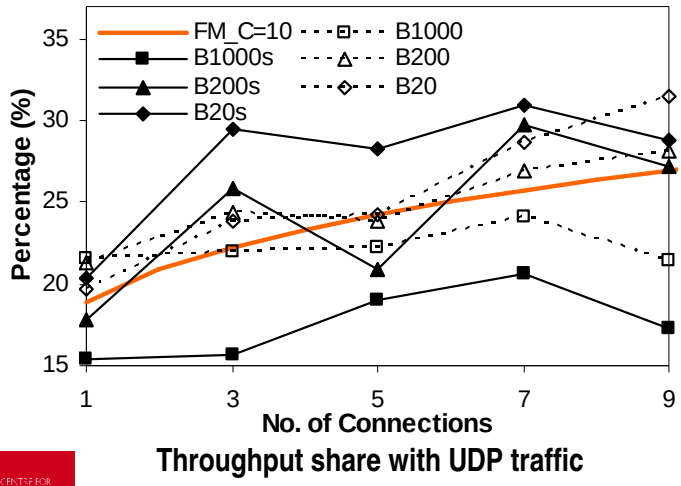
- Buffer Overflow Errors



Throughput share with TCP traffic

SR: Fractional Method

- Buffer Overflow Errors



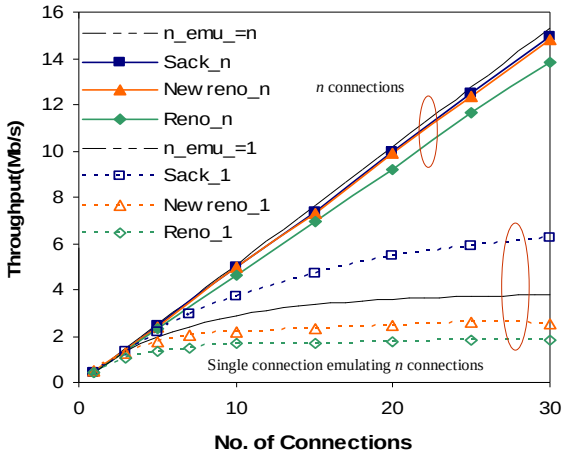
Throughput share with UDP traffic

Simulation Results

- SC Emulating TP (Single Connection Emulating TCP Parallelization)
  - MulTCP, High-Speed, Scalable, FAST
  - MulTCP
    - Use a single connection to emulate the behaviour of a set of *n* standard TCP connections.
    - The single window increases *n* packets per RTT (e.g., *m*=*n*)
    - In response to a packet loss, the window is reduced by 1/2*n* (e.g., *c*=2*n*)

SR: SC Emulating TP

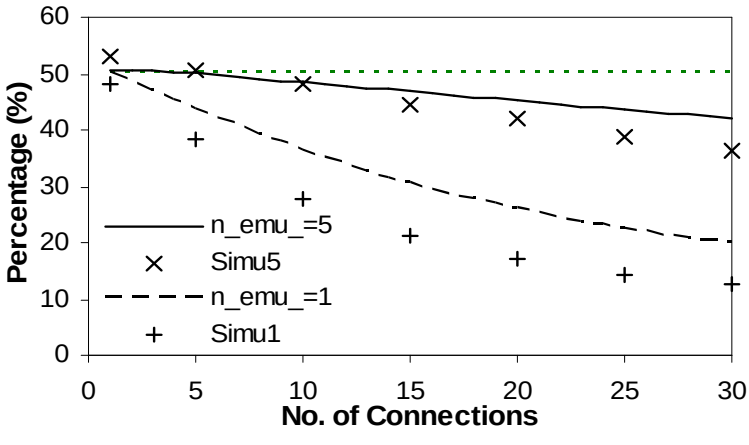
- Uniform Errors



Throughput performance (p=0.01)

SR: SC Emulating TP

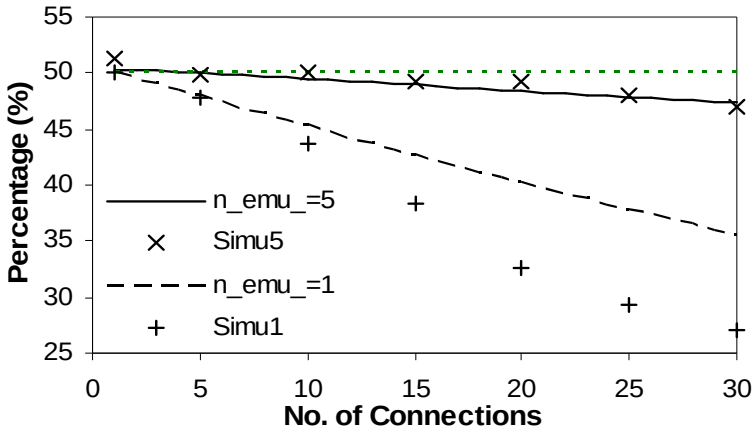
- Uniform Errors



Throughput share (p=0.01)

SR: SC Emulating TP

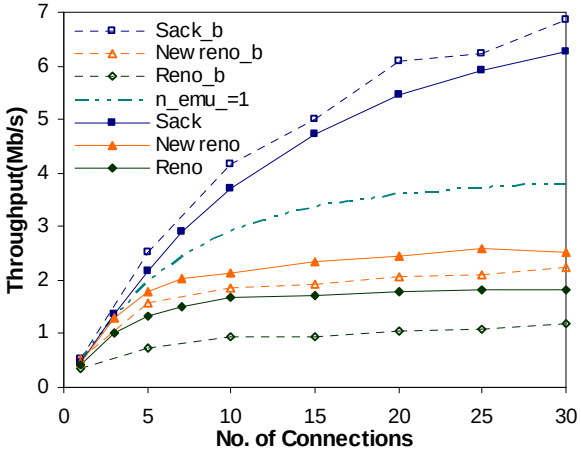
- Uniform Errors



Throughput share (p=0.001)

SR: SC Emulating TP

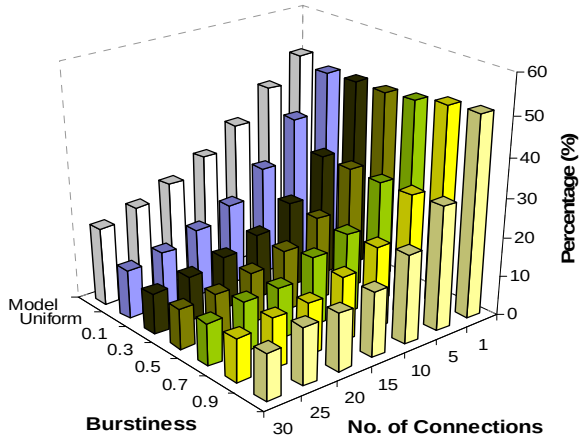
- Bursty Errors



Throughput performance with a single connection (Burstiness=0.3)

SR: SC Emulating TP

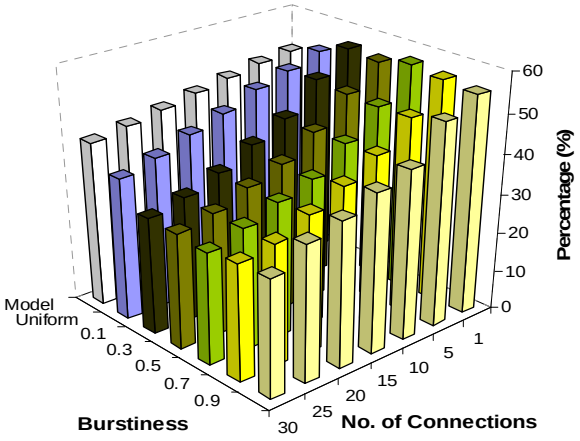
- Bursty Errors



Throughput share vs. Burstiness

SR: SC Emulating TP

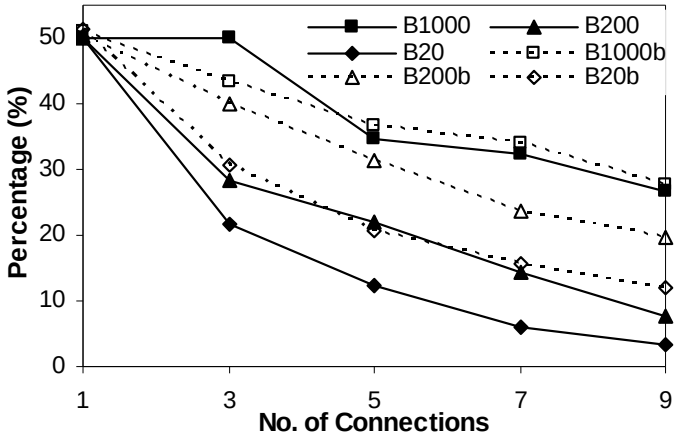
- Bursty Errors



Throughput share vs. Burstiness (5 connections)

SR: SC Emulating TP

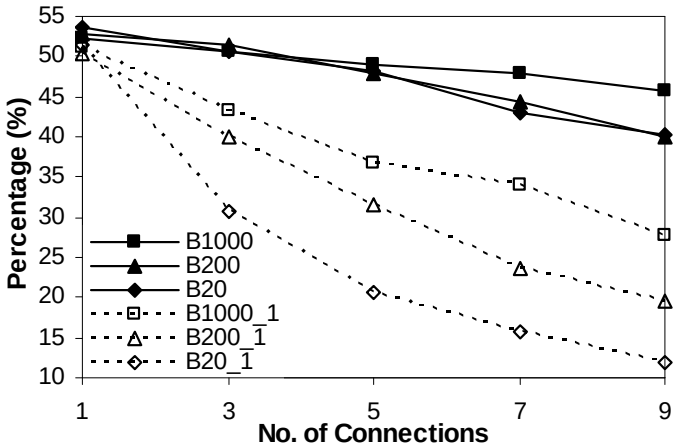
- Buffer Overflow Errors



Throughput share: UDP traffic vs. no UDP traffic

SR: SC Emulating TP

- Buffer Overflow Errors



Throughput share with UDP traffic (single connection vs. 5 connections)

A Delay-Based Approach

- Fairness and effectiveness bonded for loss based approaches
- Decouple fairness and effectiveness
  - RTT measurement to detect congestion
  - Window growth controlled by fairness factor in presence of congestion
  - Window growth controlled by effectiveness factor in absence of congestion
- Challenges
  - RTT may not reflect queue length
  - Stabilise queue size

Conclusions

- Significant performance improvement by TCP parallelization over single connection based approach
- Good option for improving TCP performance over a wide range of networks
- Not a complete solution: a fundamental step/option towards TCP performance improvement in heterogeneous networks
- Complexity, fairness concern, etc. can be addressed to a reasonable level