

# Stochastic TCP: A Statistical Approach to Congestion Avoidance

**Thomas J. Hacker**  
**Assistant Professor**  
**tjhacker@purdue.edu**

**Preston Smith**  
**Senior Systems Administrator**

Computer & Information Technology, Discovery Park Cyber Center,  
Rosen Center for Advanced Computing  
Purdue University  
West Lafayette, IN  
tjhacker@purdue.edu

# Outline

- Motivation
- Need for a Simulation Testbed
- Using the Testbed: Stochastic TCP
- Developing a Network Simulation Model
- Assessment
- Evaluation
- Conclusions

# Motivation

- Poor network performance is a serious problem
  - Achieving good performance on LFNs is difficult
  - Poor performance has a serious impact on science
  - Research relying on Cyberinfrastructure is affected
- Examples
  - Large Synoptic Survey Telescope project
    - Need to move petabytes of data every night
    - From mountaintop in La Serena, Chile to NCSA
    - Network protocols must be fast and reliable
  - Compact Muon Solenoid Project
    - Transfer 15 PB/year
    - Thousands of physicists



# Improving TCP performance

- Many approaches designed to improve TCP performance
- Alternative congestion avoidance control functions
  - Manage the size of the TCP congestion window
  - E.g. Scalable, BiC, CUBIC, FAST, Highspeed, MulTCP, TCP-Africa, ...
- Rate pacing schemes that modulate rate of packet transmission
  - E.g. TCP Pacing
- Congestion avoidance control functions that manage a UDP flow
  - E.g. Tsunami
- A combination of these approaches
  - E.g. UDT
- Parallel TCP streams
  - Widely used
  - Simple and effective
  - Extensively used in utilities such as GridFTP and bbcp

# Improving TCP performance

- These approaches improve performance, but at a cost
  - Aggressive TCP produces undesirable effects
    - Unfair bandwidth stealing from less aggressive flows
    - Potential congestion collapse from variant “arms race” by users
  - Issues with Parallel TCP
    - Can be unfair if excessive number of streams used
    - Overhead of multiplexing and demultiplexing data among packets
    - Extra memory required on receiver to buffer out-of-order packets
    - Bursty application throughput
    - Application stalls waiting for late packets.
- Increasing throughput is simple
  - Retaining necessary fairness characteristics is difficult

# Need for Simulation Testbed

- Thoroughly testing these protocols and applications that use them prior to deployment is critical
- Impractical to just deploy to see what happens
  - Testing may disrupt critical production work
  - Difficulties in partitioning and reserving large parts of the infrastructure for testing
  - Background load not easily reproducible
  - Can't reconfigure large portions of the network and systems for testing

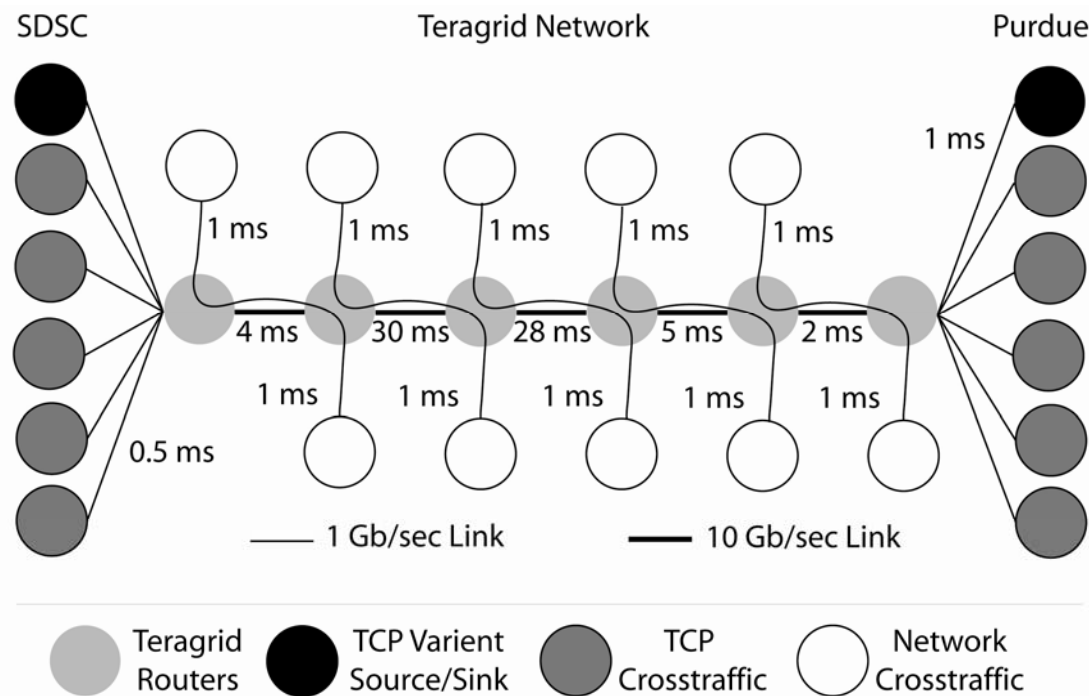
# Developing the Testbed

- Many network simulation packages are available
  - Ns-2, NISTnet, dummysnet, OPNET
- Creating a topology is straightforward in any of these simulators
- Creating a *realistic* network environment is much more difficult

# Realistic Network Simulation

- There are several critical characteristics that affect network performance that must be taken into account in developing a simulation
  - Link speed
  - Round trip time
  - Packet loss characteristics
  - Network queue lengths
  - Background/competing network traffic

# Link Speed, RTT, and Topology



- Link speeds and latencies on Teragrid network path from SDSC to Purdue

# Packet Loss Characteristics

- Packet loss has a *significant* effect on TCP performance
- Effects are especially pronounced on *Long-Fat Networks* like the Teragrid
- Using a realistic loss model is essential

# Developing a Loss Model

- Simulation loss model
  - TCP is very sensitive to loss – sensitivity increases as  $(cwnd)^2$
  - Loss model **must** be accurate
  - Inaccurate loss model can lead to inaccurate simulation results
- Developing a realistic loss model
  - Transferred 114M packets (1 TB) from SDSC to Purdue
  - Collected packets using *tcpdump*
  - Modified *tcptrace* to report time between packet losses
  - Network Operations Centers verified no router overflows in TG network during transmissions
  - Used MathWave to find best fitting statistical distribution

# Developing a Loss Model

- Weibull distribution was closest fit to observed elapsed time between packet losses
  - Shape parameter 2.9, scale parameter 0.3
- Extended ns-2 to add configurable Weibull loss model
- Time between losses follow Weibull distribution

$$G(t) = 1 - e^{-(\lambda t)^\alpha}$$

- Where  $1/\lambda$  is the scale parameter, and  $\alpha$  is the shape parameter
  - Note that this distribution is closely related to an exponential distribution: when  $\alpha=1$ ,  $G(t)$  is an exponential distribution.

# Developing a Loss Model

- Validating the loss model
- Simulated transfers using Weibull loss model
  - 83 simulated single SACK TCP transfers for 1000 sec
- Found that throughput results similar to real transmission experiments

# Network Queue Lengths

- The size of the network queues in routers on the network path influences performance
- In practice, size of the router queue is often set large enough to avoid packet drops due to overload
- A realistic simulation model needs to take into account the effects of long router queues and short router queues

# Network Queue Lengths

- Congestive vs. Noncongestive Loss
  - Real packet losses arise from congestion, as well as from non-congestive sources
  - Two simulation methods to take both into account
  - Longqueue – 1000 packet router queues
    - Primary source of loss will be non-congestive
    - Weibull loss model in effect to account for non-congestive loss
  - Shortqueue – 50 packet router queues
    - Losses from congestion and non-congestion sources
    - Weibull loss model in effect to account for non-congestive loss

# Background/Competing Traffic

- Background and competing traffic also affects performance
- Network flows for an application must compete with other users and applications for network bandwidth
- Created two simulation scenarios to assess effects of crosstraffic
  - No Crosstraffic
  - TCP Crosstraffic
    - Simulations with no TCP cross traffic, and TCP cross-traffic from 5 nodes at SDSC to Purdue
- Crosstraffic used simulated FTP traffic
  - Recorded crosstraffic throughput
  - Duration of crosstraffic was the entire length of simulations

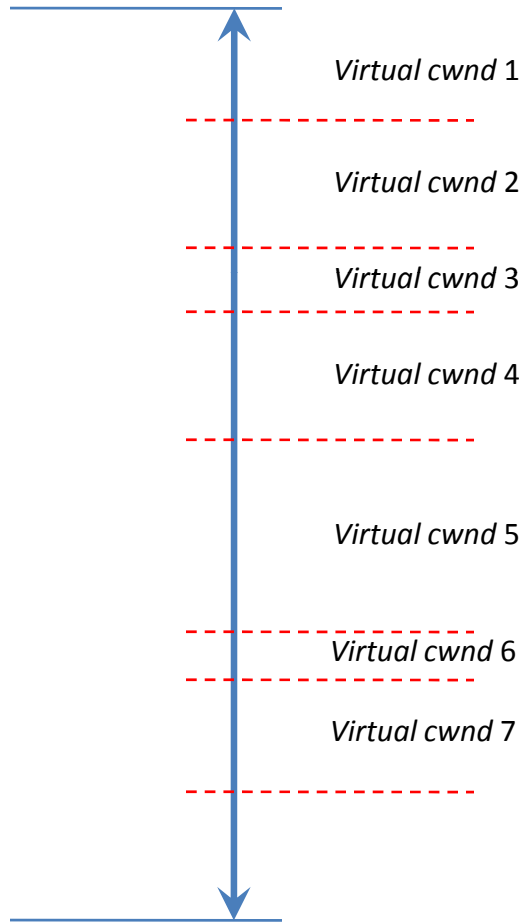
# Using the Testbed: Stochastic TCP

- We sought to develop an improved TCP that...
- Uses only a single TCP stream
  - Parallel TCP requires too much overhead
- Improves performance over SACK and Reno
  - Matches Parallel TCP performance
  - Matches or exceeds high speed variant performance
- Is fairer to competing TCP streams
  - Better fairness than Parallel TCP
  - Equivalent or better fairness than high speed variants

# Stochastic TCP

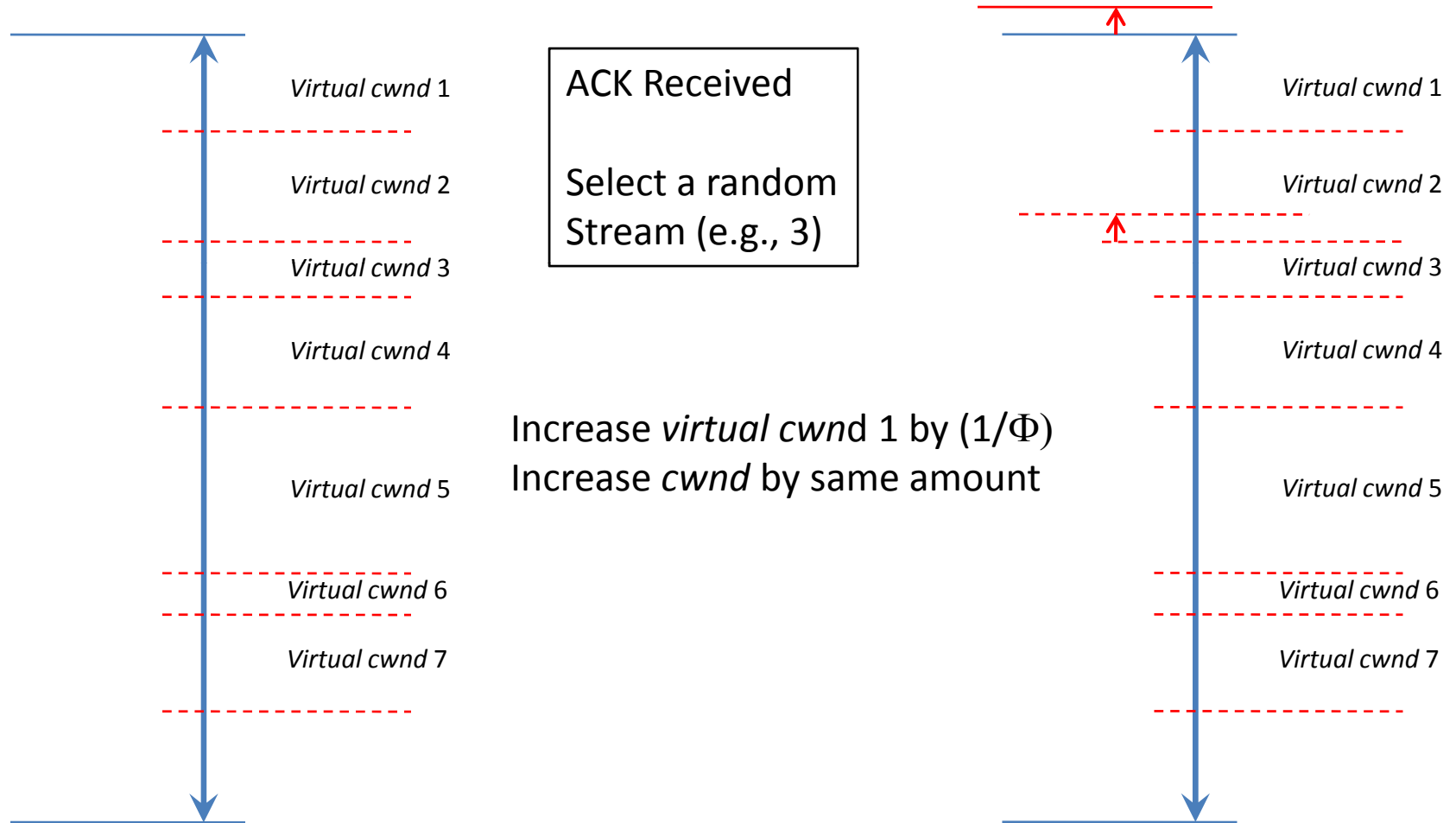
- Stochastic TCP
  - Emulates Parallel TCP behavior using a *single TCP stream*
  - Manages a single congestion window as a set of *virtual TCP streams*
- TCP congestion window is partitioned
  - Set of independent *virtual congestion windows*
  - Virtual congestion windows are stochastically managed as if there were individual TCP streams
- Why “Stochastic”?
  - Virtual congestion windows are selected for gains and loss using a stochastic process
  - Adjustments made to virtual *cwnd* using stochastic loss/gain bookkeeping
  - TCP congestion window updated to include changes

# Stochastic TCP



Stream TCP Congestion Window *cwnd*

# Stochastic TCP



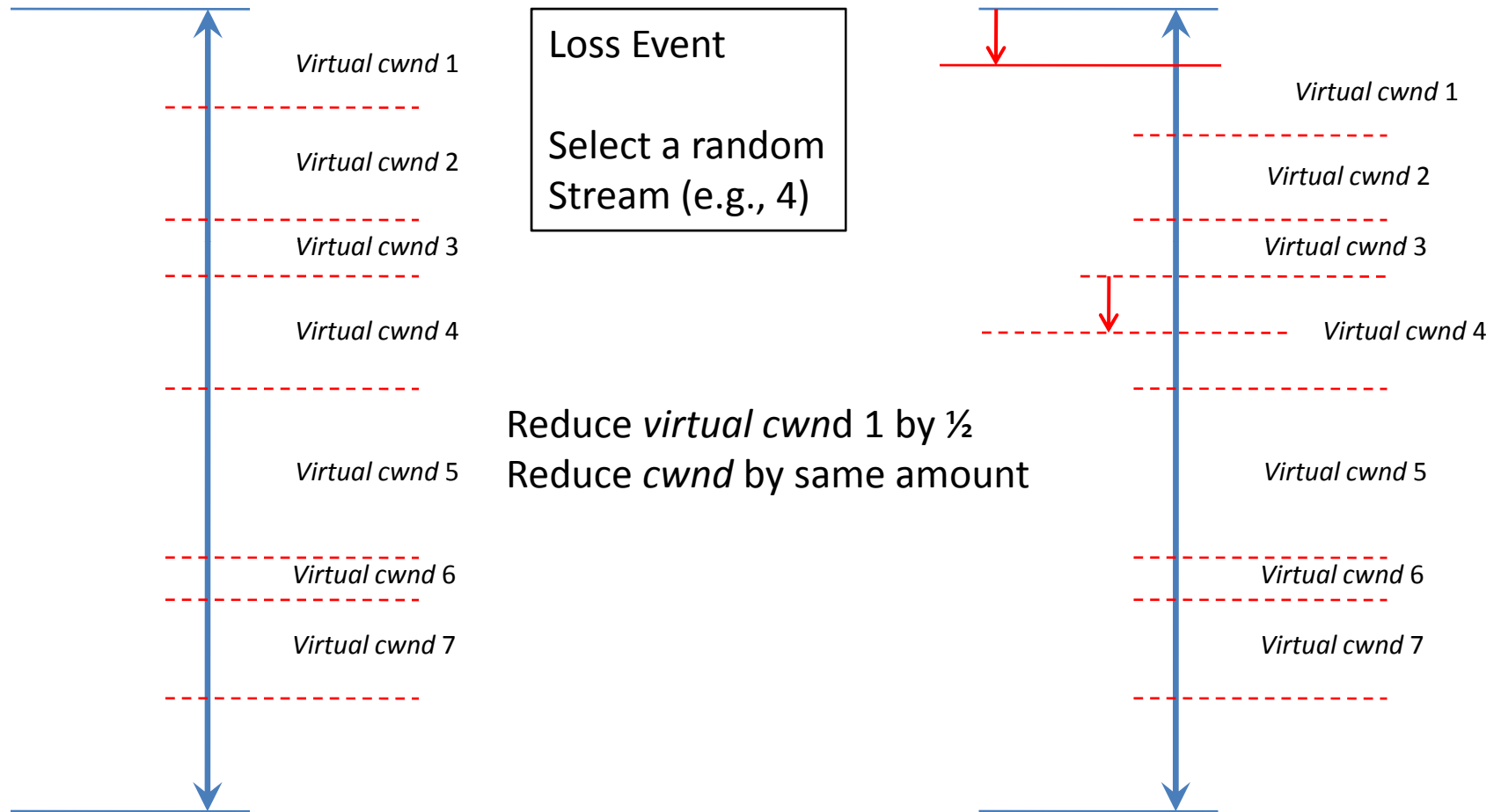
ACK Received

Select a random  
Stream (e.g., 3)

Increase *virtual cwnd 1* by  $(1/\Phi)$   
Increase *cwnd* by same amount

Stream TCP Congestion Window *cwnd*

# Stochastic TCP



Stream TCP Congestion Window *cwnd*

# Emulating Parallel TCP Performance

- Parallel TCP characteristics
  - An  $N$  stream Parallel TCP session increases throughput by
    - Increasing  $cwnd$   $N$  times faster than single stream
    - Decreasing  $cwnd$  by  $1/(2*N)$  instead of  $\frac{1}{2}$  on loss
- Prior work on Combined Parallel TCP
  - Showed how to increase throughput, but retain fairness
  - A set of parallel TCP streams that couple two types of TCP streams
    - One standard SACK TCP stream
    - A set of *fractional* streams with reduced aggressiveness
  - Exploits RTT unfairness
  - Fractional streams experience a *virtual* long RTT
- Stochastic congestion avoidance algorithm
  - Congestion window behaves as a set of parallel TCP streams
  - Modulate aggressiveness of virtual streams using fractional approach

# Stochastic TCP goals

- Single TCP stream interface for applications
- Match Parallel TCP performance
  - Without the block management (mux/demux)
  - Eliminate out-of-order packet caching at client
  - Reduce application stalls
- Using Combined Parallel TCP approach
  - Improve throughput over a SACK/Reno TCP stream
  - Provide better fairness than Parallel TCP

# Assessment Methodology

- Measuring the effects of Stochastic TCP
  - Interested in assessing
    - Effectiveness
      - Does throughput increase compared with Reno/SACK?
    - Efficiency
      - Is the network utilization increased (w/o inducing congestion)?
    - Fairness
      - Is bandwidth unfairly stolen from competing flows?
- Investigate interactions with other TCP variants
  - SACK, Reno, Stochastic TCP
  - A set of other high speed TCP variants
    - CUBIC, BIC, Scalable, Compound, FAST, Highspeed, H-TCP

# Assessment Methodology

- Experimental procedure
  - Each experiment was run 10 times, varying random number seeds for ns-2
  - Simulated transmission time of 1000 seconds
  - Varied the number of virtual and real TCP streams ( $N$ )
  - Long and short router queues
  - TCP crosstraffic and no crosstraffic
  - Stochastic TCP alone
  - Stochastic TCP and Parallel TCP
  - Stochastic TCP and a TCP variant
    - SACK, Reno, CUBIC, BIC, Scalable, Compound, FAST, Highspeed, H-TCP
  - $\Phi$  values of 1,  $N$ ,  $N^2$ ,  $N^3$
  - Very large simulation space
- Used BoilerGrid Condor installation at Purdue to run simulations
  - Around 30,000 total simulation trials
  - Generated a tremendous amount of simulation data

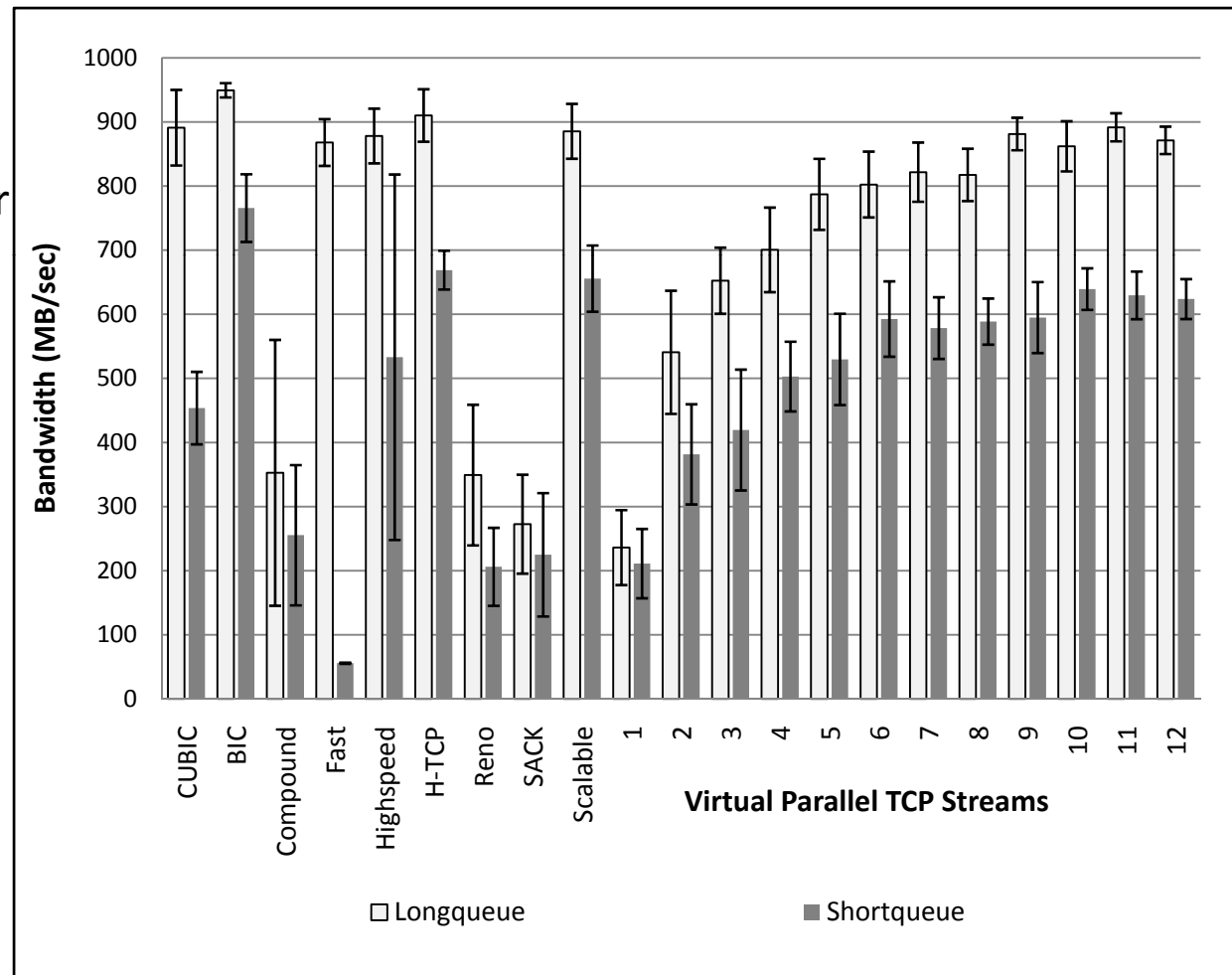
# Assessment Methodology

- Hypotheses we tested
  - 1) Stochastic TCP increases throughput compared with a single TCP SACK Stream
  - 2) Stochastic TCP has performance characteristics similar to Parallel TCP
  - 3) Stochastic TCP delivers similar or better effectiveness, efficiency, and fairness compared with other TCP variants

# Evaluation - Effectiveness

- Throughput of variant competing with no crosstraffic

Stochastic TCP identical to Parallel TCP as the number of streams increased

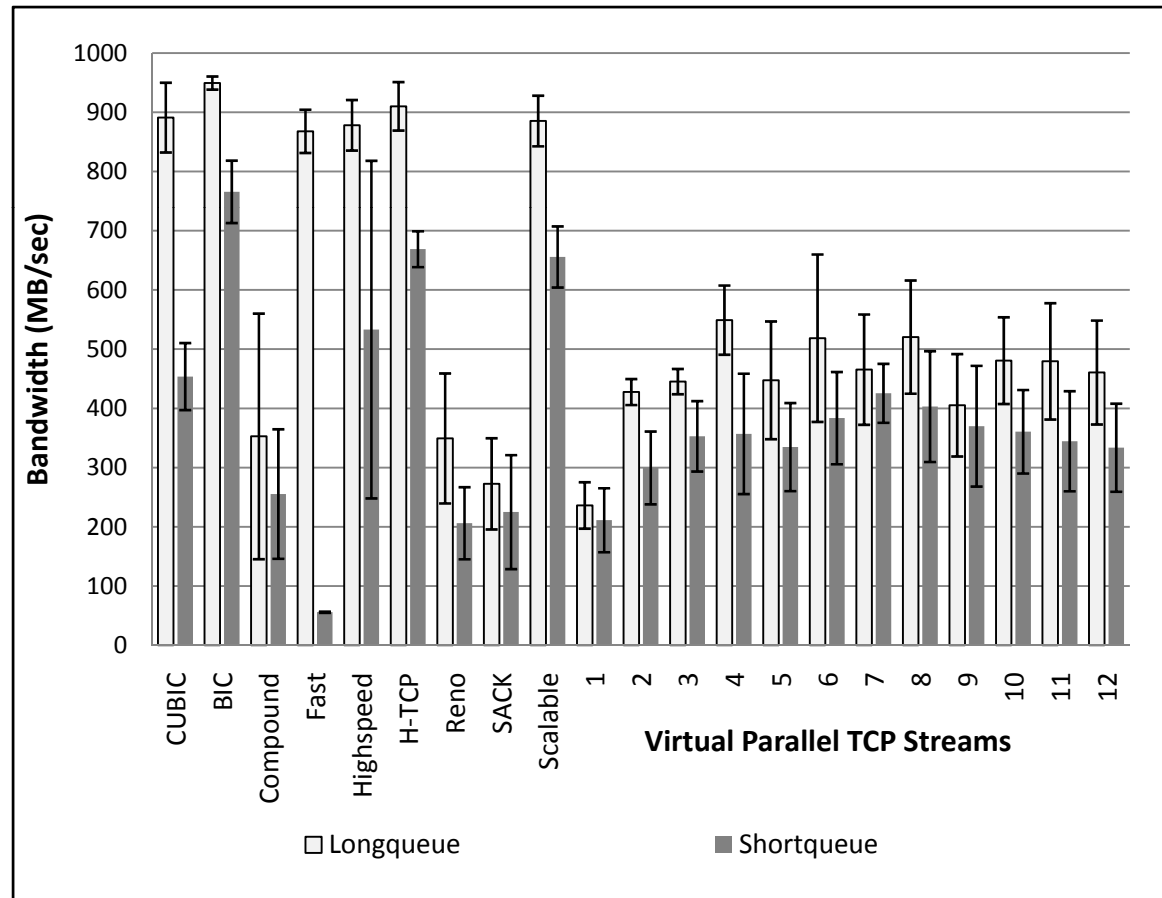


$$\Phi = 1$$

# Evaluation - Effectiveness

- Throughput of variant competing with 5 TCP crosstraffic streams

Stochastic more effective than single SACK/Reno



$$\Phi = n$$

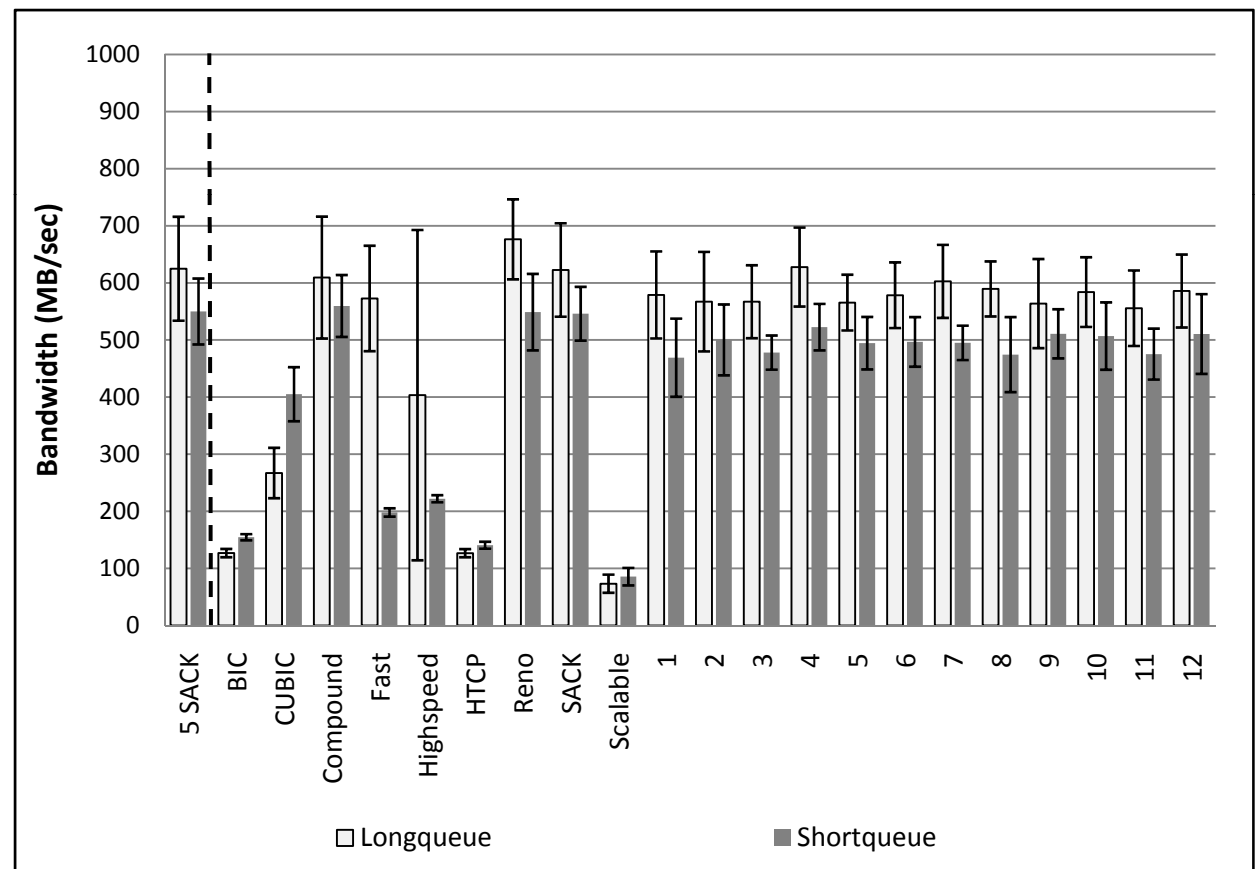
# Evaluation - Fairness

- Total throughput of 5 TCP crosstraffic streams competing with variant

Stochastic TCP similar to combined parallel TCP

Much fairer than most other TCP variants

Compound TCP is as fair as Stochastic TCP



$$\Phi=n$$

# Conclusions

- A network simulation testbed for the Teragrid network proved to be valuable and useful
  - Permitted detailed and exhaustive analysis of highspeed TCP protocols without affecting the production network
- Careful instrumentation of the simulation was critical to developing a *realistic* simulation
- Useful TCP assessment results
  - Aggressive TCP variants (Scalable, Fast, BIC, HTCP) can severely degrade performance of competing TCP streams from other users/applications
  - Fairer variants (Compound, Stochastic TCP) can improve bandwidth
    - Without unfairly stealing bandwidth from competing flows
- Network simulation revealed properties of Stochastic TCP
  - Stochastic TCP increases throughput compared with a single TCP SACK Stream
  - Stochastic TCP has performance characteristics similar to Parallel TCP
  - Stochastic TCP provides an appropriate balance among effectiveness, efficiency, and fairness

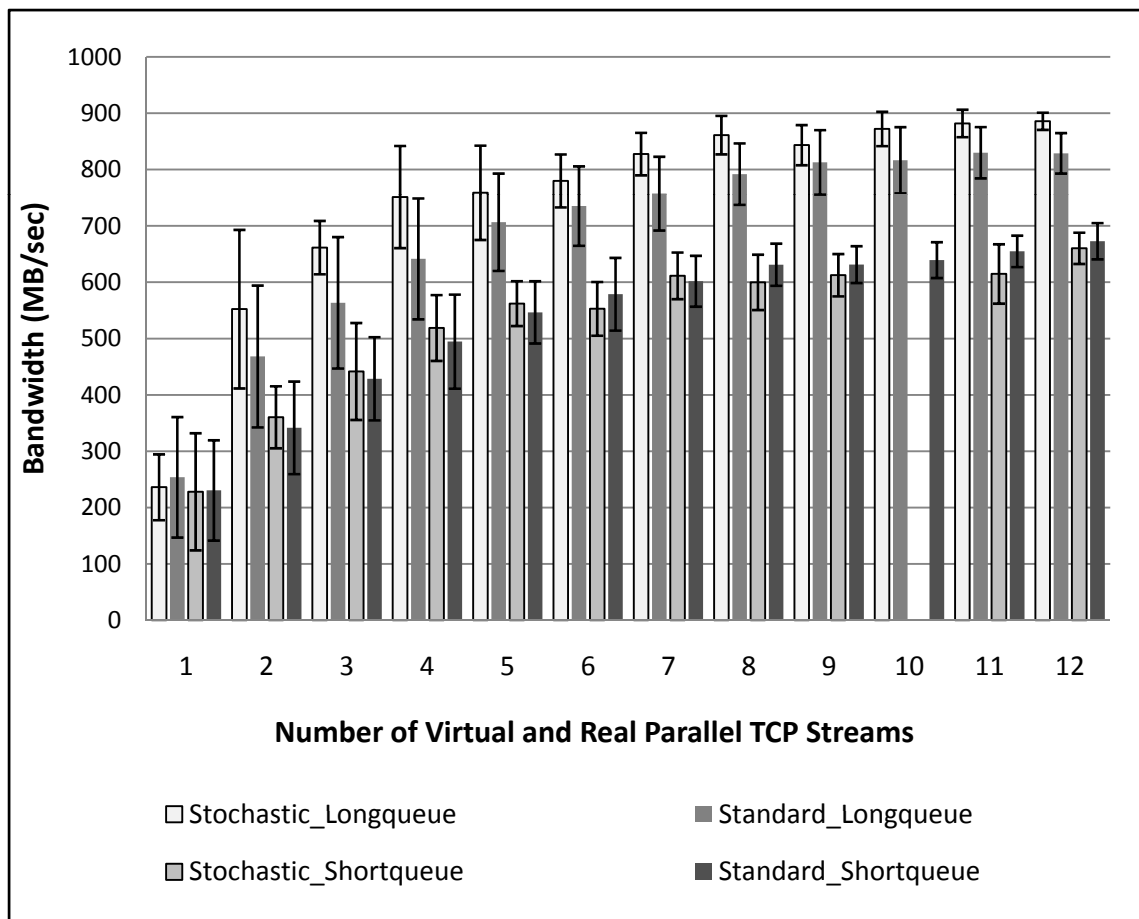
# Backup Slides

# Evaluation - Effectiveness

- Throughput with no TCP crosstaffic

Performance Improvement over single stream

Stochastic TCP is statistically equivalent to Parallel TCP



$$\Phi=1$$

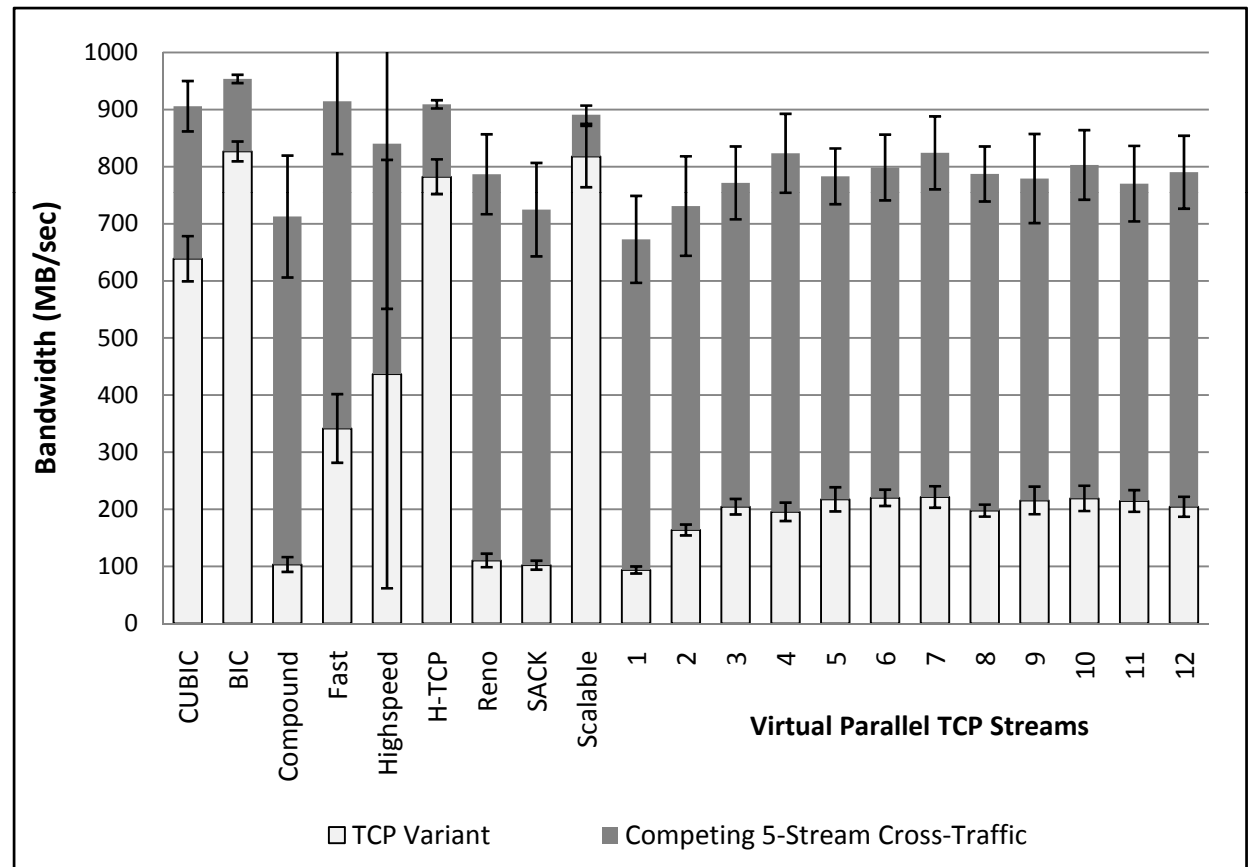
# Evaluation - Effectiveness

- Throughput increased beyond a single SACK stream
  - For  $\Phi = 1, n$
  - Effectiveness decreases with  $N$  for  $\Phi = N^2, N^3$ 
    - Low aggression streams affected by non-congestive loss
- Stochastic TCP throughput was statistically equivalent to Parallel TCP
  - For  $\Phi = 1$

# Evaluation - Efficiency

- Throughput of Variant & 5 TCP crosstraffic streams

Improves throughput over single SACK/Reno stream, but NOT at the expense of fairness



$$\Phi=n$$

# Assessment Methodology

- Developed an accurate simulation model of the Teragrid network
  - A high speed wide-area network regularly used by the research community
  - Purdue is a member of the Teragrid
  - Several research projects (such as the LHC CMS) use the Teragrid network
- Created an ns-2 simulation of the Teragrid network
- Link speeds and latencies from SDSC to Purdue TG network path

# Implementation of Stochastic TCP

- Changes made to the TCP networking code in the kernel
  - Implemented as a pluggable congestion avoidance module
  - Sysctl call added to set number of virtual TCP streams and aggression factor  $\Phi$
  - Data structure (*virt\_stream*) added to represent virtual streams
    - Each virtual stream maintains *snd\_cwnd*, *snd\_cwnd\_cnt*, *ssthresh*
  - virtual *cwnd* updates are also applied to real *cwnd*
- Initialization and Slow start of Stochastic TCP
  - Each virtual *cwnd* set to one packet, total *cwnd* set to number of virtual streams
  - Slow Start: Virtual *cwnd* incremented by  $1/\Phi$  for each received ACK
    - $\Phi=1 \Rightarrow$  Standard TCP Stream
    - $\Phi > 1 \Rightarrow$  Reduced aggression: more ACKs required to increase *cwnd* - equivalent to longer RTT

# Implementation of Stochastic TCP

- Timeout
  - One stochastically selected virtual *cwnd* reduced to 2 packets
- Linear Increase phase
  - Stochastically selected virtual *cwnd* incremented by  $1/(cwnd * \Phi)$  for each ACK
  - Increased by one packet when the number of received ACKs that were assigned to the virtual stream  $> snd\_cwnd * \Phi$
  - Mathematically equivalent to longer RTT when  $\Phi > 1$
- Responding to packet loss
  - Stochastically selected virtual *cwnd* reduced by  $\frac{1}{2}$
  - Total *cwnd* reduced by the same amount
- Miscellaneous Bookkeeping
  - Several places in *tcp\_input.c* where the congestion window is tweaked for various reasons
  - Code added to keep  $\Sigma$  (virtual congestion windows) = real congestion window

# Implementation of Stochastic TCP

- Stochastic stream selection process
  - Generating a sequence of random numbers in the Linux kernel
    - Used *random32()* in the kernel
    - Didn't seem to effect CPU load
- Investigated two kinds of stochastic processes
  - Weighted
    - Selection probability is a function of the size of the virtual congestion window
  - Unweighted
    - Selection probability unrelated to virtual congestion window size
- Weighted stochastic process favored large streams
  - Did not correctly emulate parallel TCP behavior
- Unweighted stochastic selection process reproduced observed Parallel TCP behavior

# Conclusions (2)

- Does Stochastic TCP meet goals?
  - ✓ Provides single TCP stream solution
  - ✓ Improves performance over Reno/SACK
  - ✓ Matches Parallel TCP or other high speed TCP variant performance when appropriate
  - ✓ Can be fairer to competing TCP streams, with tunable  $\Phi$
  - ✓ Can improve throughput, but not at the expense of fairness