New TCP Stack Comparisons

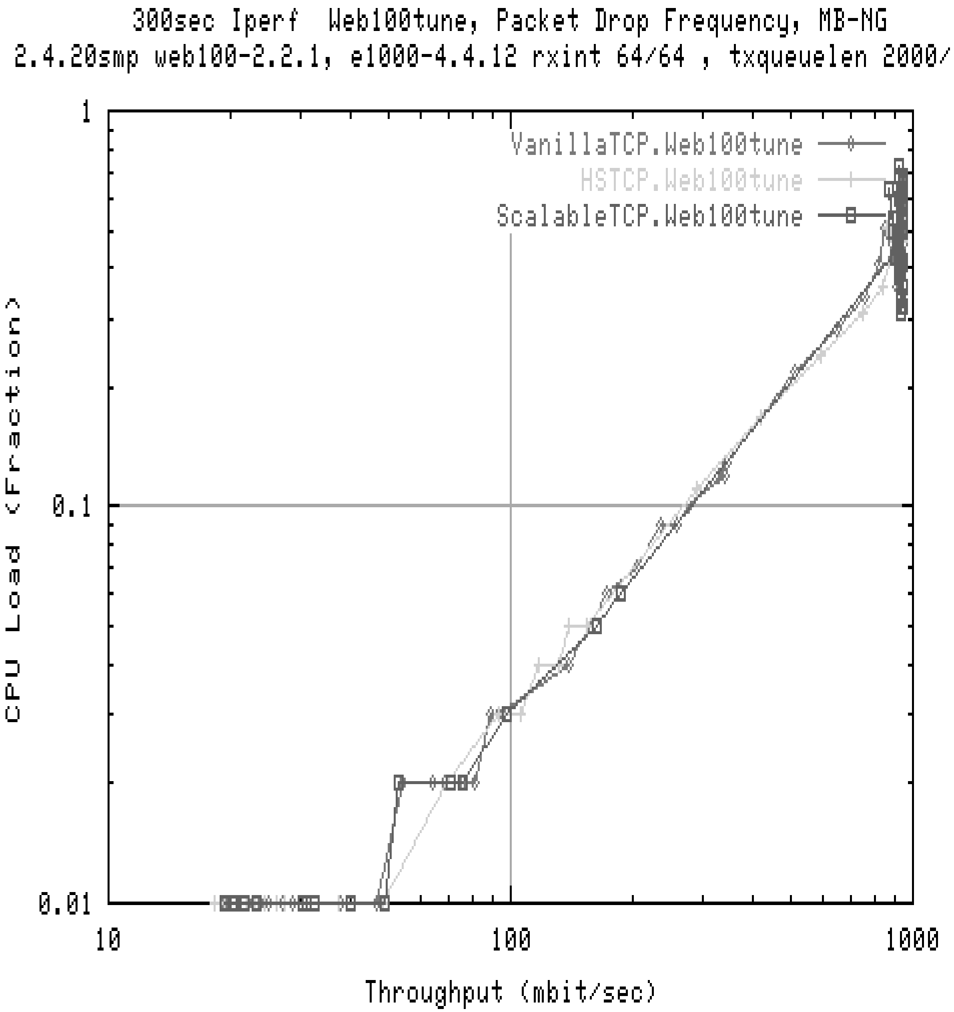
2nd Year PhD Presentation
30th September 2003

Yee-Ting Li, UCL

Introduction

- The internet relies on ‘protocols’ to transfer data
- Grid relies on movement of A LOT of data
  - LHC predicts a petabyte of data per year
- End to end throughput depends on many factors
  - Hardware (CPU, NIC, PCI bus)
  - Software (kernel version, scheduling mechanisms)
  - Network (Routers, network paths, other traffic/congestion)

Hardware

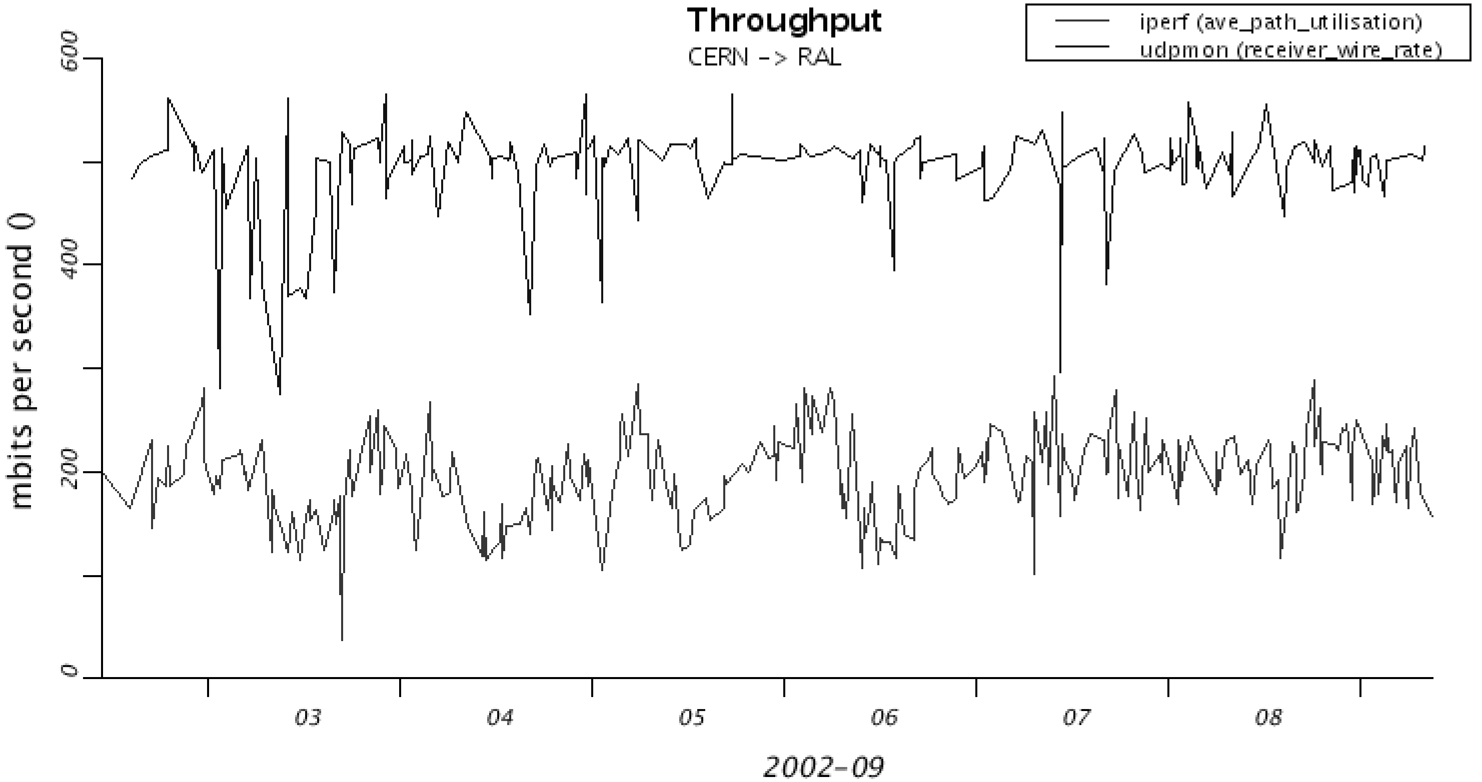


- CPU load on a server-class pc
  - Dual 2Ghz
  - 1Gbit/sec NIC
  - 64bit dual channel PCI
- Hardware is up to the job

Transport Level Protocols

- TCP (HTTP, FTP, GridFTP) used for most things
  - Gives guarantee on delivery
  - All data is copied precisely
  - Performance can be poor
  - Respects other internet users
- UDP (Real, H323) used for video conferencing
  - Gives no guarantees on delivery
  - Data may be incomplete
  - Performance good
  - Doesn’t respect other internet users

UDP versus TCP



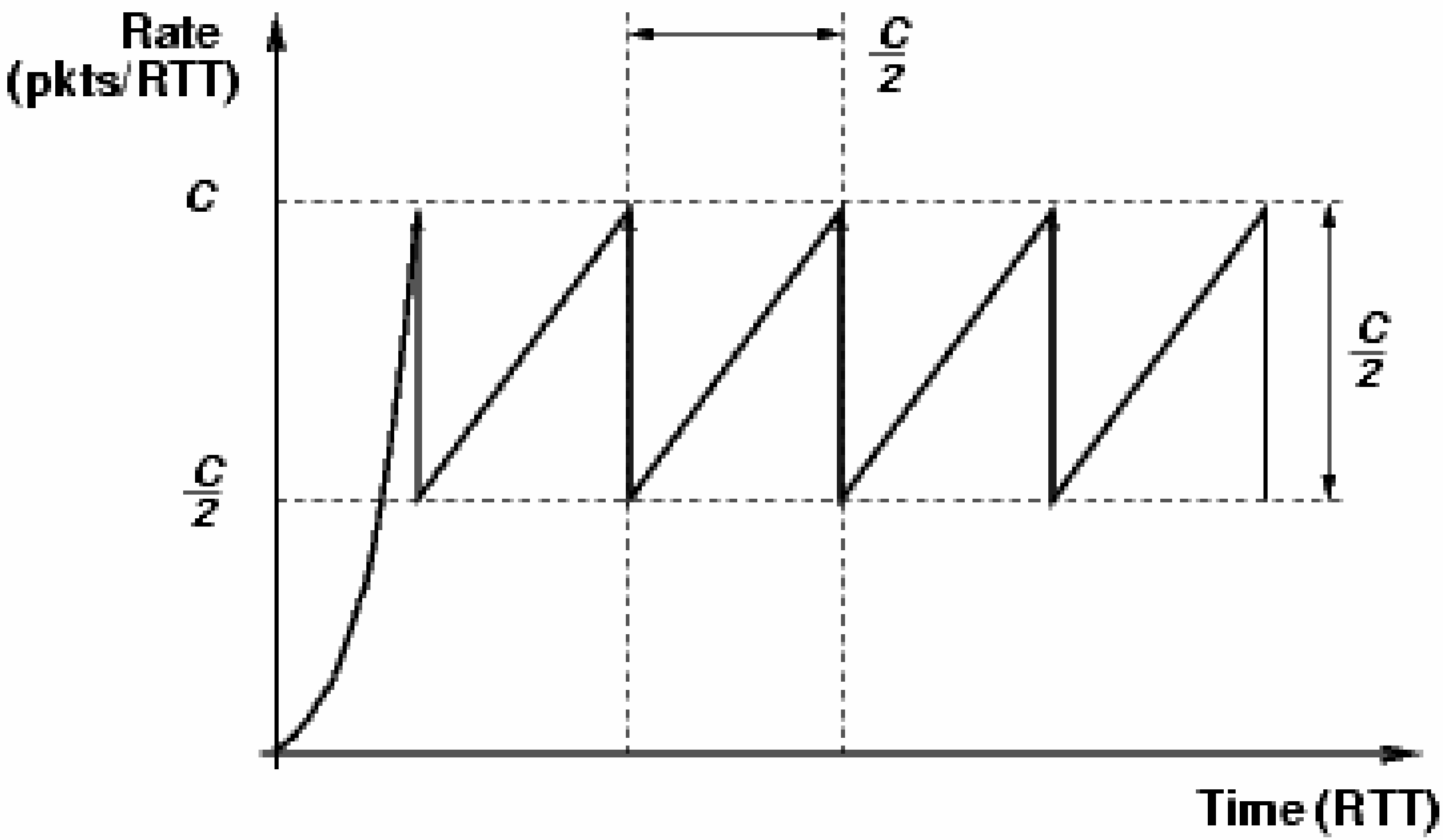
Transmission Control Protocol

- Main transport protocol used in the internet
- Relies on internal algorithms to
  - Enable reliable byte delivery
  - Flow control (to prevent overrunning senders and recvs)
  - Congestion Control (to prevent internet collapse)

Congestion Windows

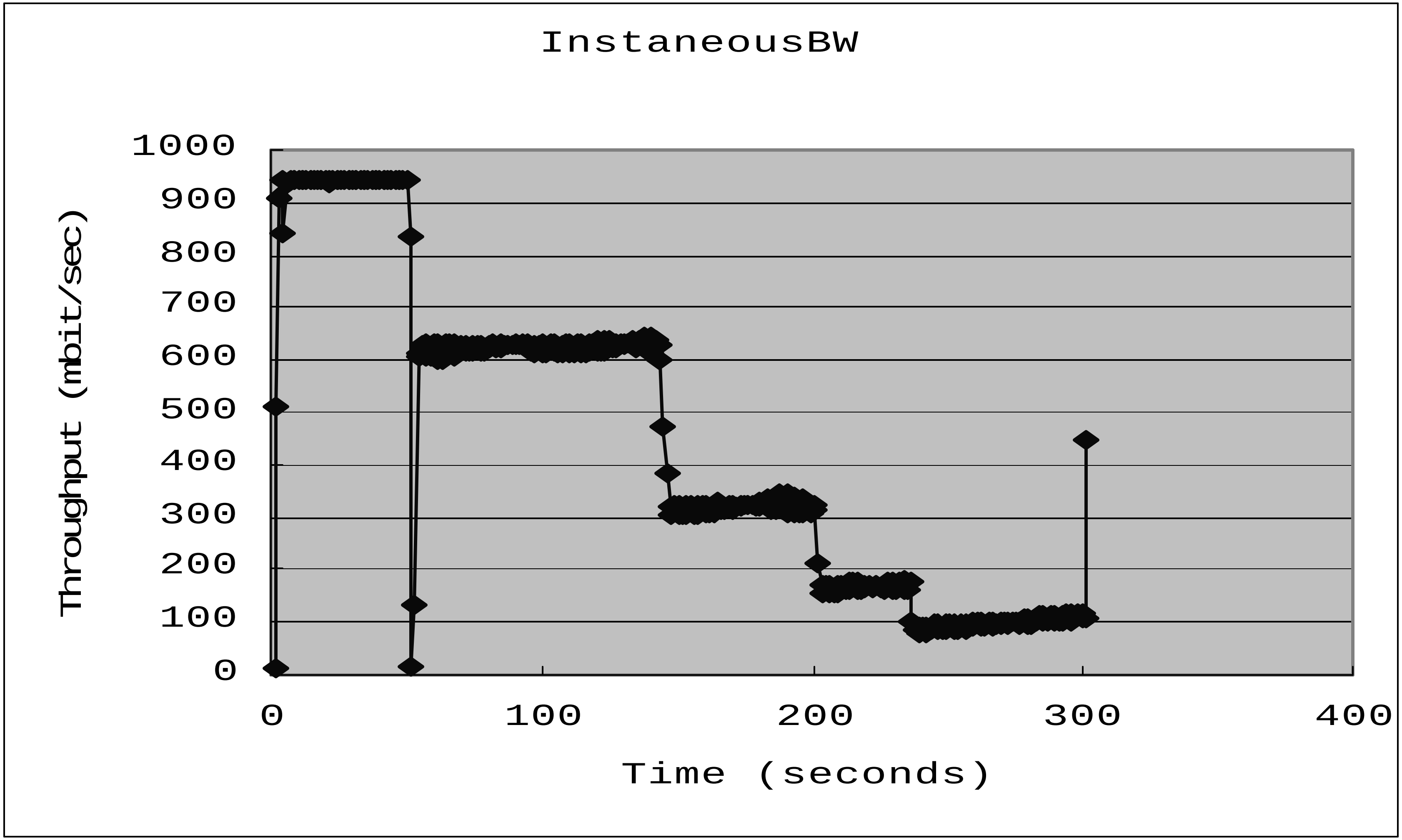
- Uses a variable ‘congestion window’ (cwnd) to enable congestion control
  - Idea is to regulate the rate of packets out so we don’t put *too much* into the network
  - Purpose: maintain a steady throughput, but at same time
    - see if we can get more throughput we increase
    - If we experience congestion (packet loss), then decrease sending rate

Typical TCP



Traditional TCP with large capacity

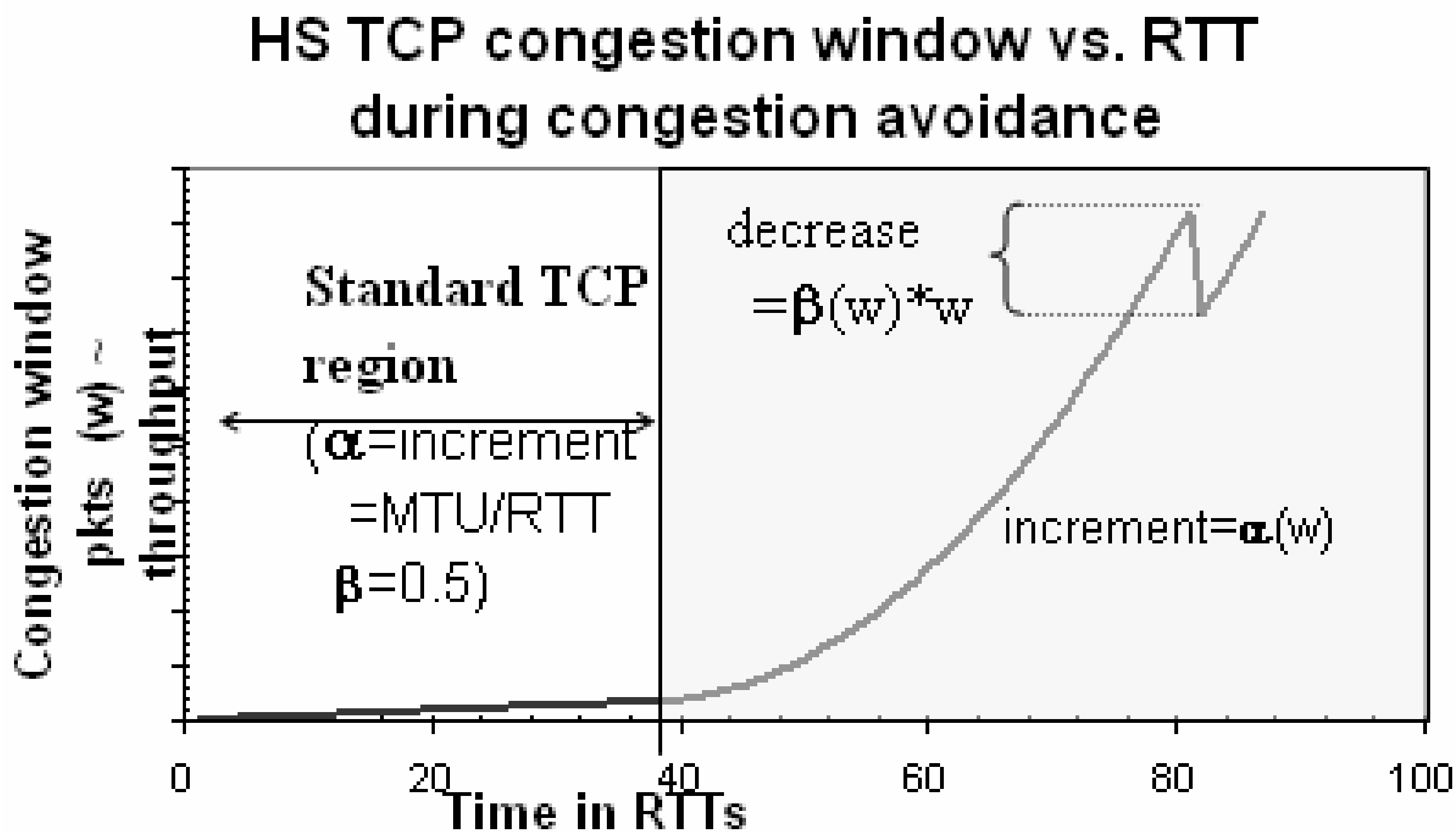
New Networks – TCP performance



New TCP Protocols

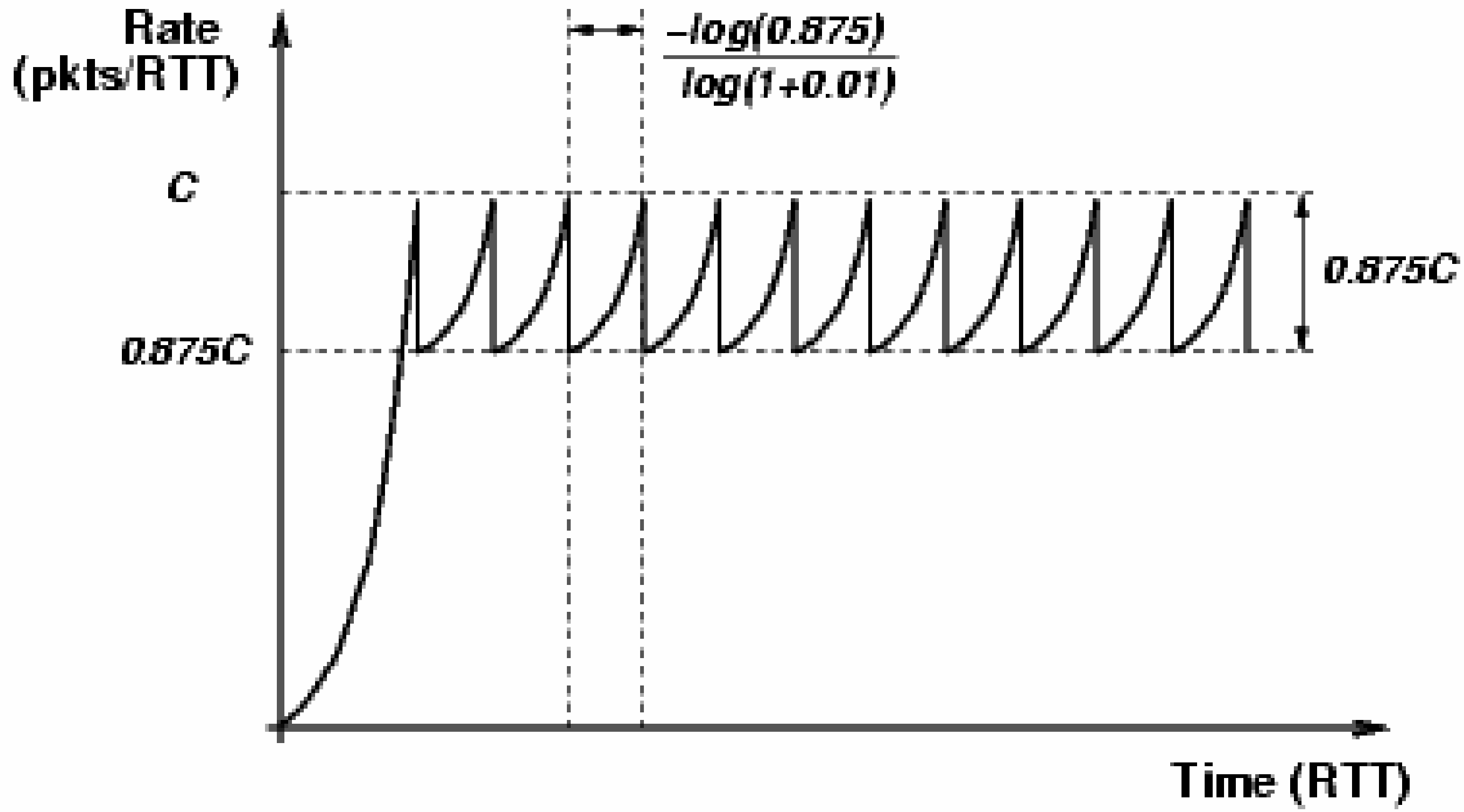
- Problem
  - Current (Vanilla) TCP doesn’t perform well under high bandwidth (and high delay) networks because of cwnd algorithms
- Solution
  - New algorithms for updating cwnd!
    - HSTCP
    - ScalableTCP
    - FAST

HSTCP



HS TCP congestion window vs. RTT
during congestion avoidance
Congestion window pkts (w) ~ throughput
decrease
=β(w)*w
Standard TCP
region
(α=increment
=MTU/RTT
β=0.5)
increment=α(w)
0
20
40
60
80
100
Time in RTTs

ScalableTCP



Scalable TCP with large capacity

Outline

- ***Determine*** and ***Quantify*** the performance of next generation TCP based transport protocols

What is Performance?

- Ultimately the throughput achieved
  - Based in terms of single stream under different network conditions
    - Vary the Packet Drop Rate (shown before)
    - Vary the CBR background rate
    - Vary the Self Similar background rate
- Other factors:
  - Fairness
    - Define as the ‘equivalence’ to other protocols
    - If we can say that the new TCP performance like n-VanillaTCP streams, then we can use all the existing research on parallel TCP as basis
  - How ‘stable’ the TCP flow is – ie the stdev of throughput

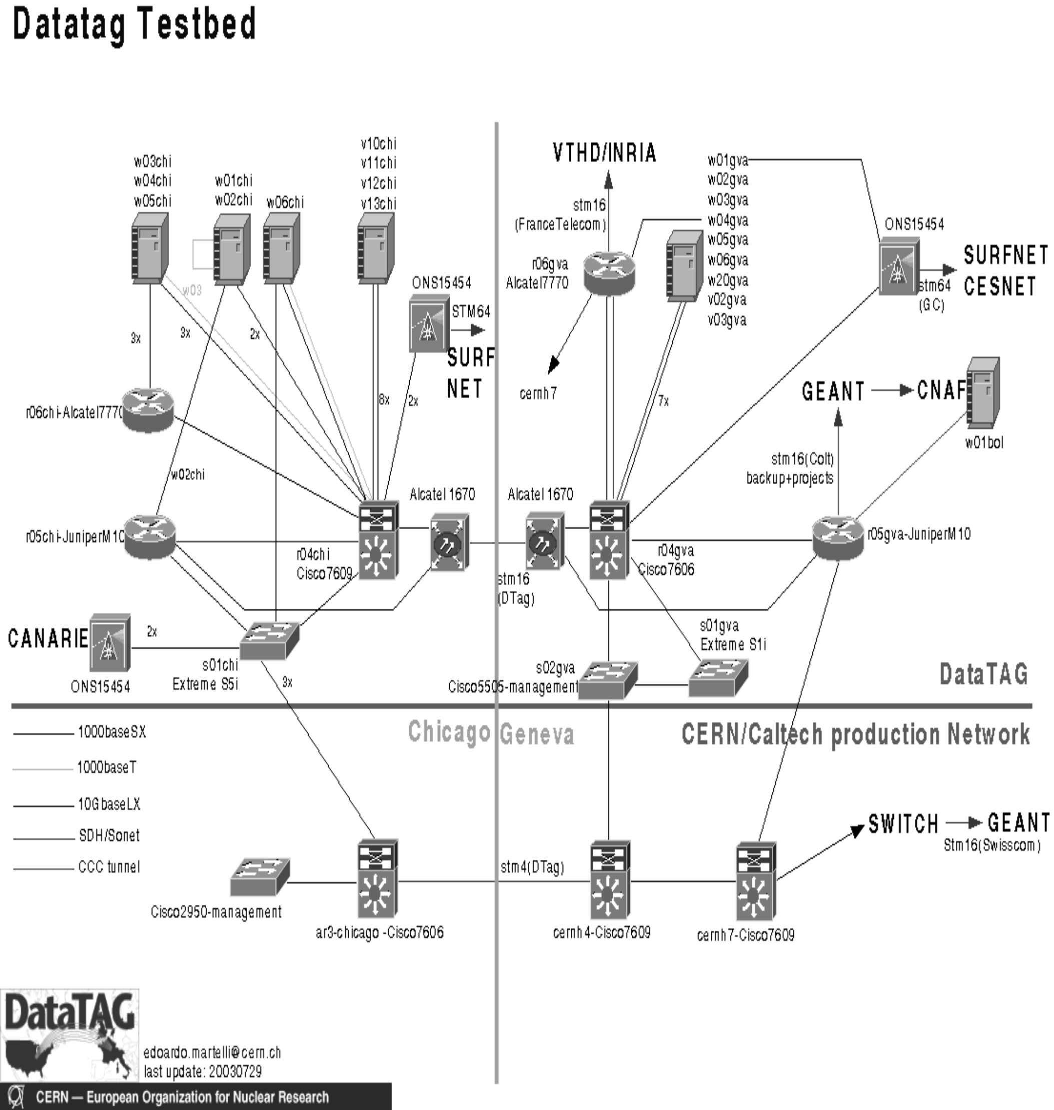
Background

- All tests conducted with the 2.4.20 ‘Alternative-AIMD’ kernel
- Need to define metrics
  - Based on the fact we are interested in how much throughput the flow gets and the stability of the flow
- Coefficient of Variance =

$$\frac{\text{Stdev Throughput}}{\text{Mean Throughput}}$$

  - Low CoV means better ‘performance’

Network Configuration



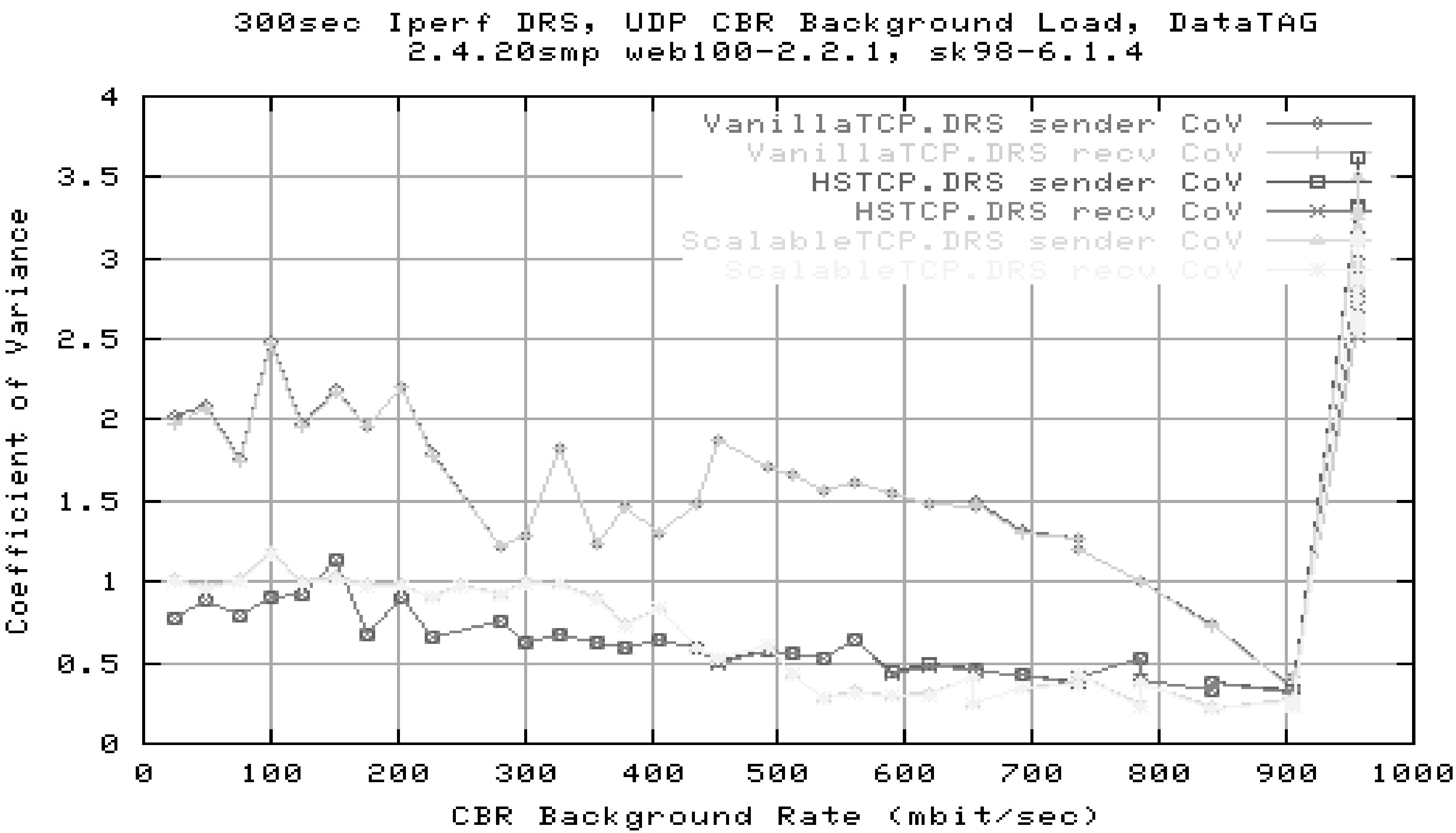
DataTAG

- High throughput (1Gb/sec)
- Long latency (120ms)
- Test network – no other competing traffic

CBR Background Tests

- Two pairs of hosts
  - UDP CBR with iperf at various rates
  - TCP stream with other pair
  - same bottleneck (1Gb/sec)

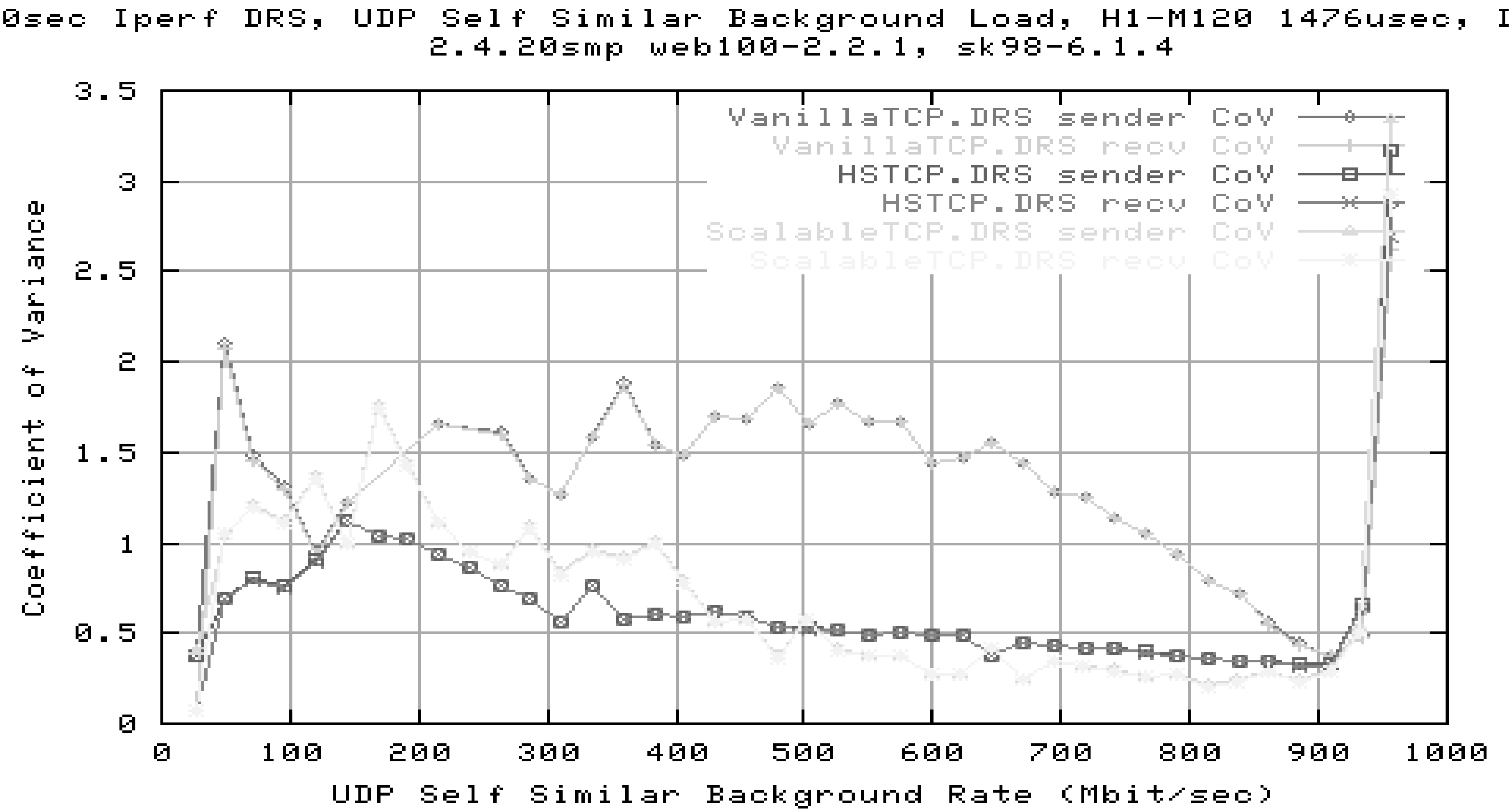
CBR Background Tests



Self Similar Background Tests

- In the 90’s computer scientists prooved that aggregate internet traffic is not Poisson but Self Similar in nature
  - Traffic profile generate with an FFT-FGN
  - Frank’s FlashIP program used to put traffic profile onto network
- Parameters:
  - Hurst 1
  - Max packets 120 per 1476usec
  - Variance of 1 packet
- Define that 120 is at line rate (1Gb/sec) and extrapolate average number of packets for each self similar background throughput

Self Similar Background Tests



Summary – Background Load

- Vanilla
  - Is rubbish ☺
  - Backs off too readily, takes too long to recover (increase throughput)
  - so throughput is very poor
- HSTCP
  - Better with CBR and SS tests – especially with low background loads

Summary – Background Load

- Scalable appears to be best at obtaining good throughput
  - Not so good when it has an empty(ish) pipe (CoV lower for low bg rates)
    - Caused by the fact that ScalableTCP is too aggressive and induces losses within itself
    - HSTCP performs better in this region

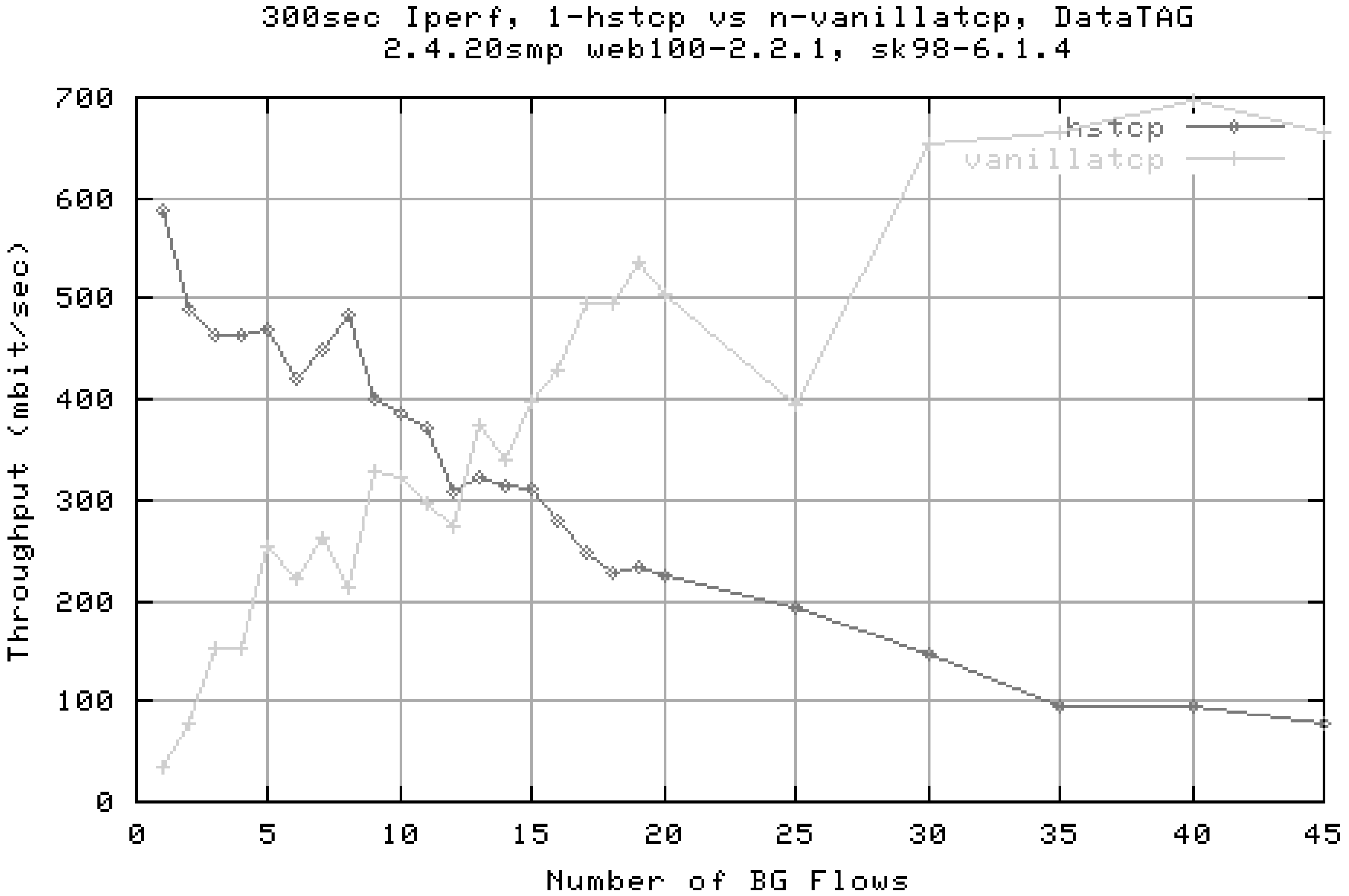
Fairness Tests Overview

- Two types
  - Equivalence Tests
    - n-flows Comparison against 1-flow new stack
    - how many flows of one stack gives the same accumulative throughput of one flow of another
    - Idea: use existing research in parallel tcp flows to determine capability and fairness of new tcp stacks
  - Transient Fairness
    - Start one flow until ‘stable’
    - Start another flow; how long does the average throughput take to become 50/50 ratio

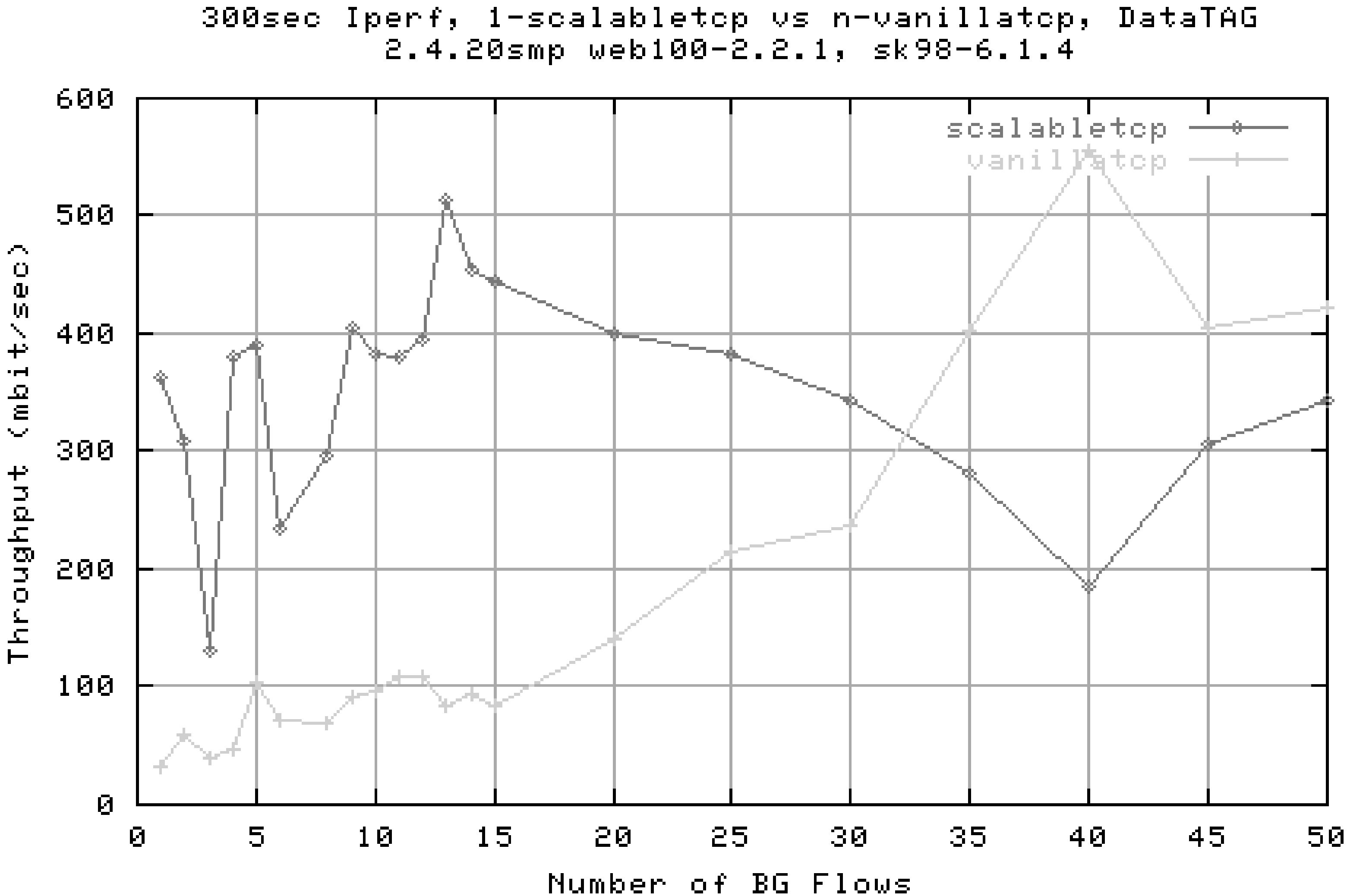
Equivalence Tests

- Currently people use parallel TCP streams to obtain high throughput
  - Aggregates the congestion window dynamics so it is no longer standard TCP
- Lots of research already...
  - Why not use that as comparison of the new stacks?

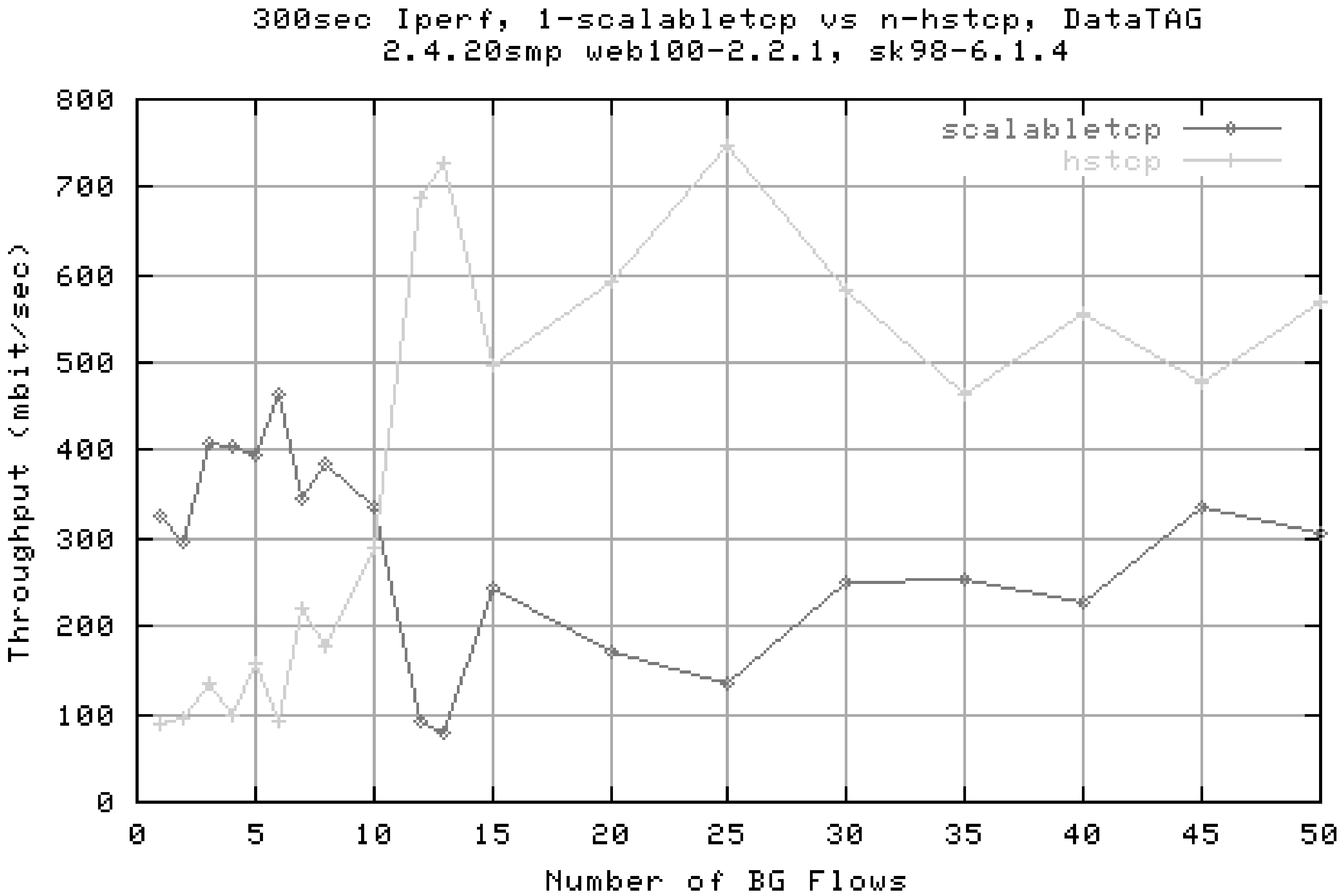
Equivalence Test - HSTCP



Equivalence Test - ScalableTCP



Equialence Tests



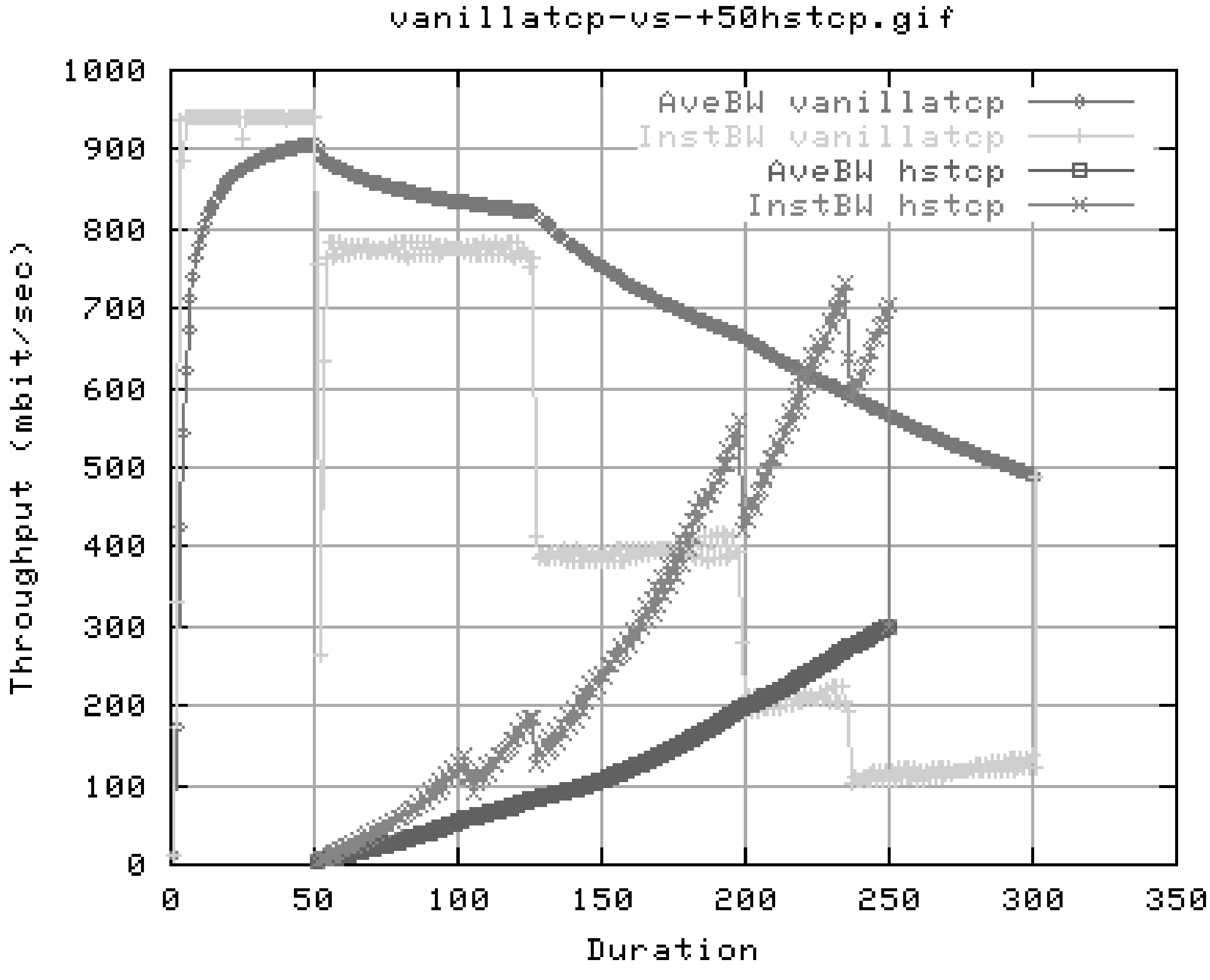
Summary - Fairness

- HSTCP is roughly equivalent to 12 Vanilla flows
  - Theory states that at 500Mps, HSTCP is ~ 15 times relatively fair
- Scalable is very unfriendly
  - Equivalent of about 30 odd VanillaTCP flows and 10ish HSTCP flows

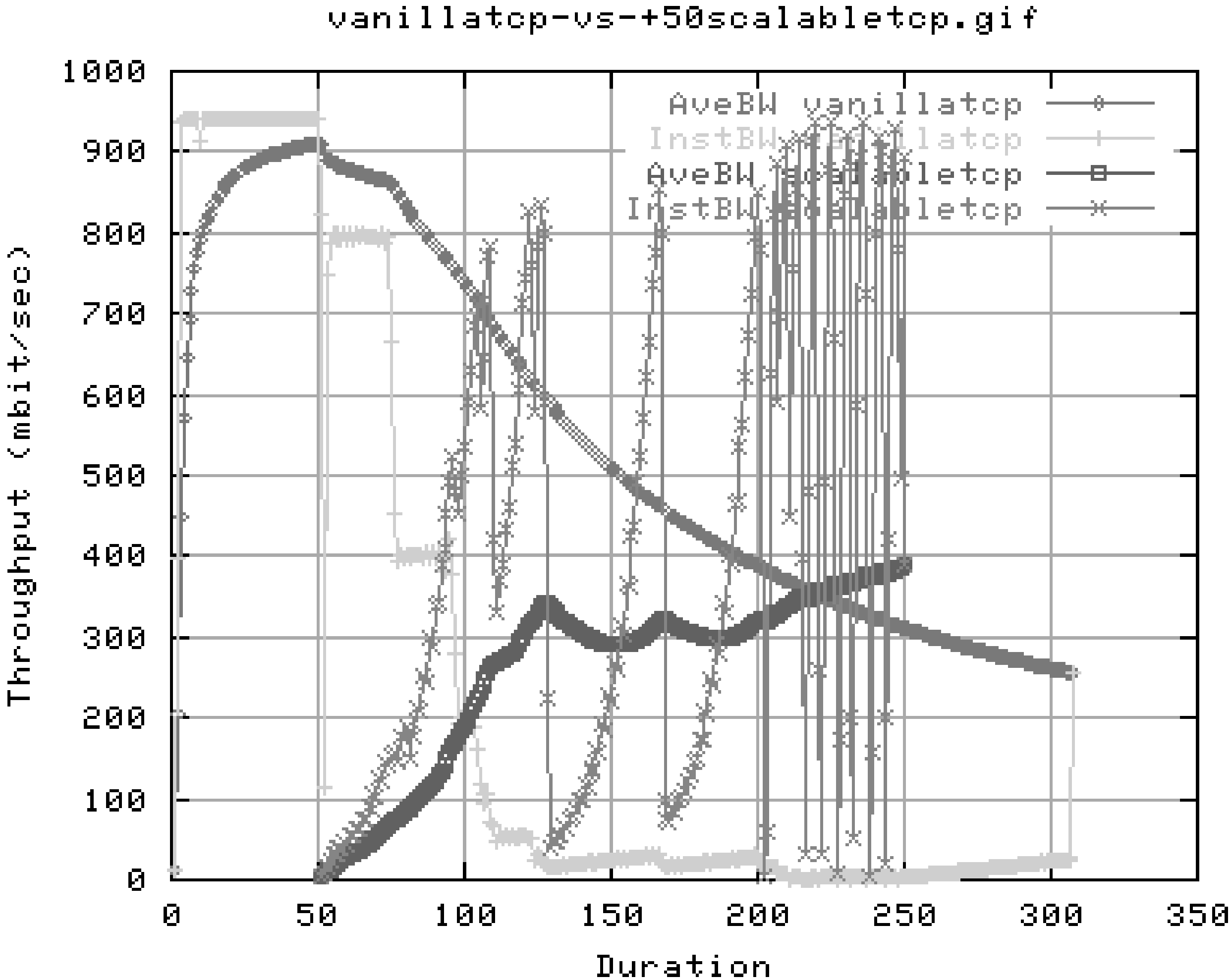
Transient Fairness Tests

- To give an idea of the stability of having these new stacks at a microscopic level
- Start a single tcp flow
  - After a while (50sec), initiate another flow
- Determine how long it takes for the two streams to ‘converge’
  - Work in progress: use running averages and ratio of streams to determine convergence point

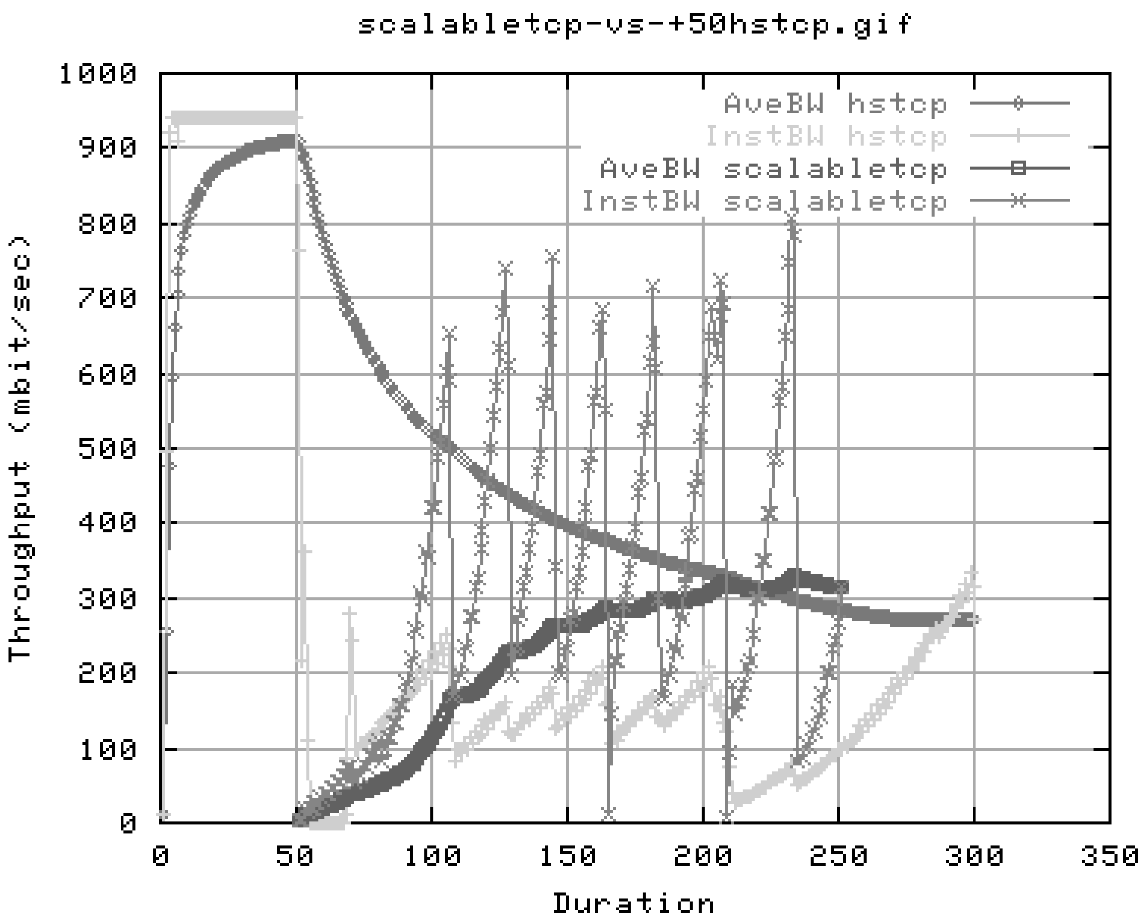
VanillaTCP vs HSTCP



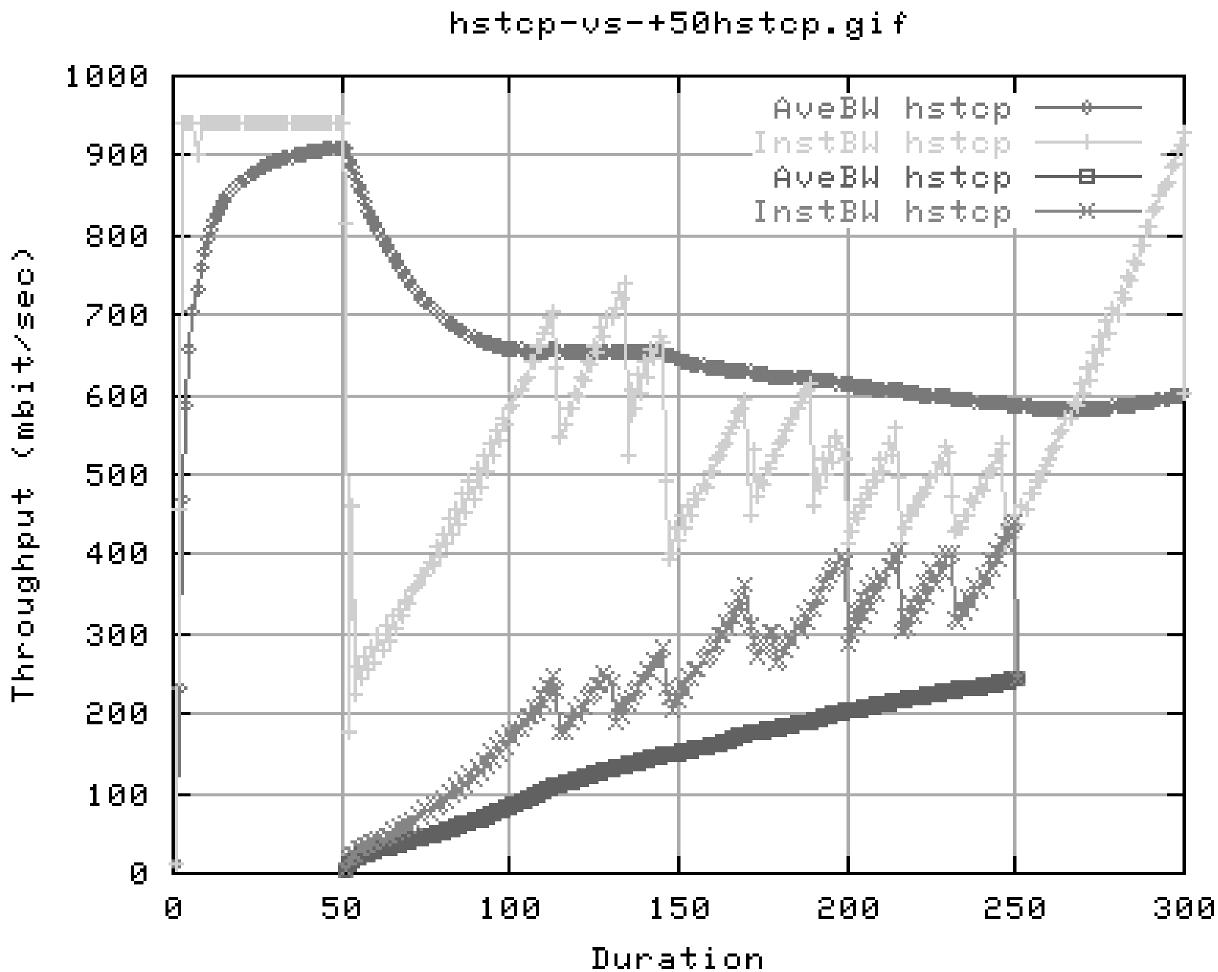
VanillaTCP vs ScalableTCP



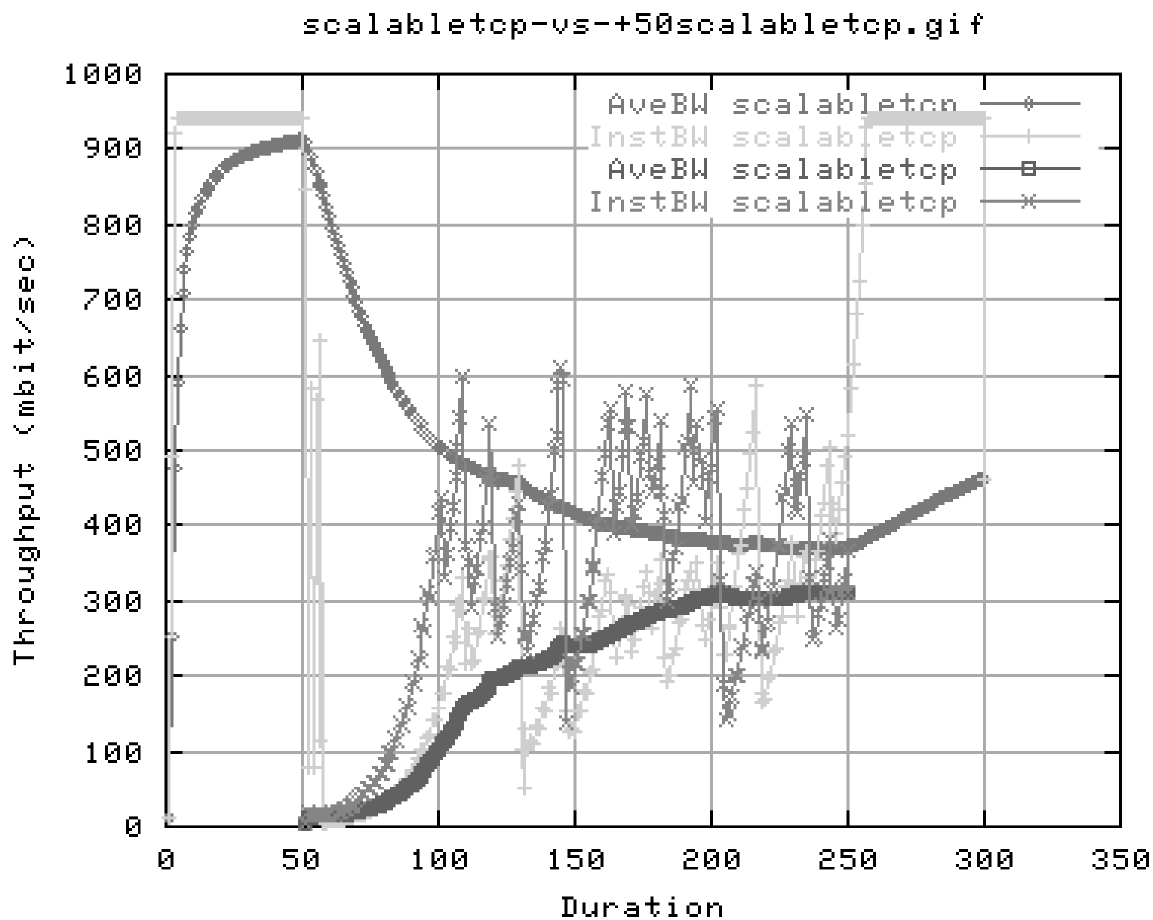
HSTCP vs ScalableTCP



HSTCP vs HSTCP



ScalableTCP vs ScalableTCP



Summary – Transient Fairness

- Need to run tests for longer than 5 min
- Very unstable; can rarely reproduce graphs – need to run many many times to get average
- Preliminary Summary
  – ScalableTCP grabs BW from VanillaTCP a lot quicker than HSTCP
  – ScalableTCP actually forces HSTCP down such that the Transient Fairness is ~VanillaTCP
  – HSTCP vs HSTCP takes a long time to converge

Other Stuff

- PhD:
  - Internet Protocols for Grid Transport
- Current focus is at lowest layers (TCP/IP)
- Now focus on higher layers
  - Non TCP Transport (Tsunami, SABUL)
  - GridFTP, BBCP, BBFTP (program efficiencies, cpu utilisation, etc)