

Implementing High Speed TCP (aka Sally Floyd's)

Yee-Ting Li & Gareth Fairey

1st October 2002

DataTAG Presentation

@ CERN (Kinda!)

What is High Speed TCP?

- Changes the way TCP behaves at high speed (ie large cwnd)
- Standard TCP has two modes
 - Slow start (not very slow...)
 - Congestion Avoidance
- Focuses on Congestion Avoidance Mode
 - ie when TCP knows (thinks it knows...) how well the network behaves...
- BUT only when we are at high speeds, else do what normal Standard TCP does...
- Readily deployable 1st step towards Equation Based Congestion Control

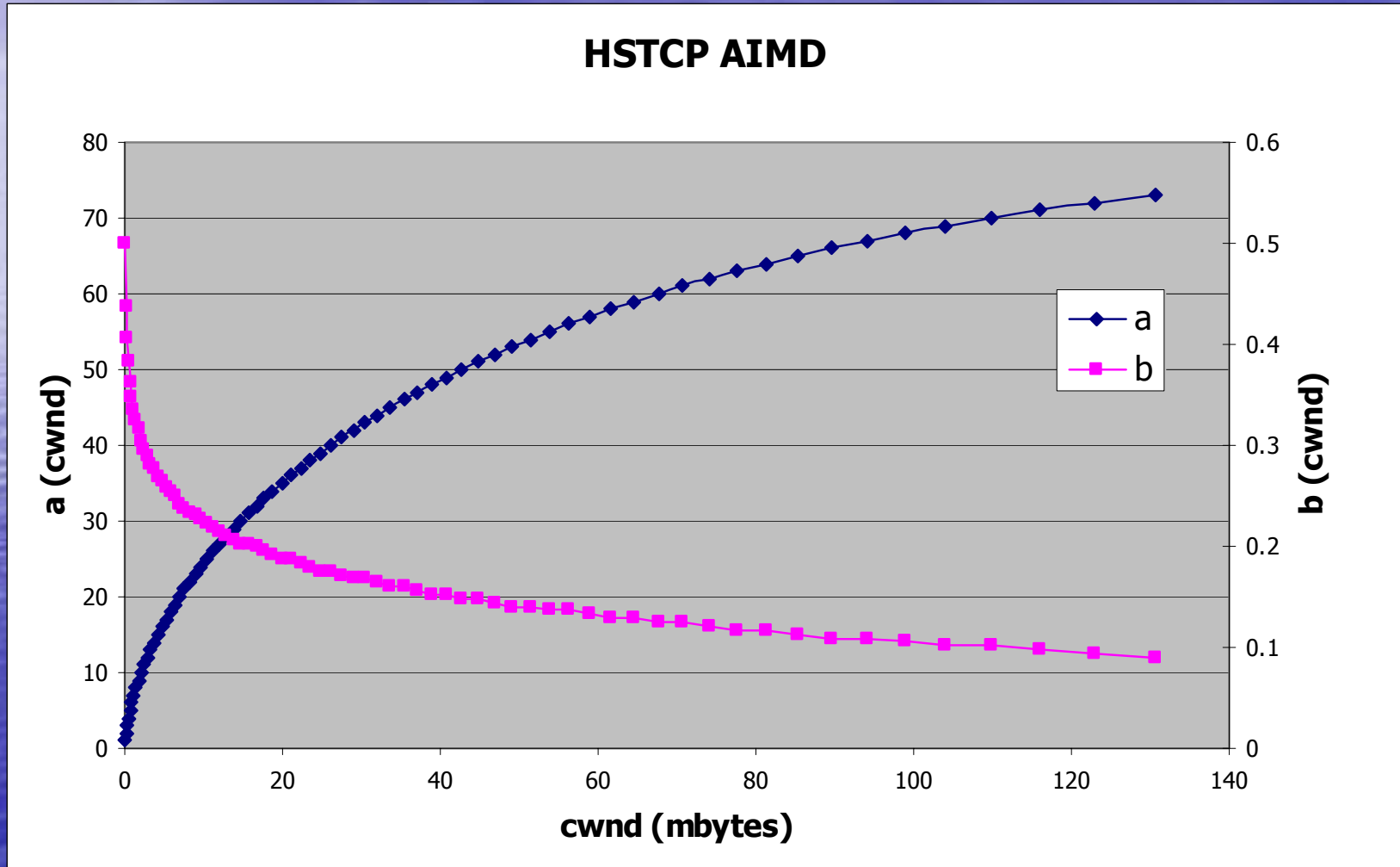
What does it do?

- TCP uses two parameters during Congestion Avoidance - AIMD(a, b)
 - Increase parameter, a
 - Decrease parameter, b
- StandardTCP uses
 - $a=1$
 - $b=0.5$
- High Speed TCP introduces a dependance of a and b depending on the current $cwnd$
 - $a \rightarrow a(cwnd)$
 - $b \rightarrow b(cwnd)$
- If we increase a more with larger $cwnd$ we can get back up to our 'optimal' $cwnd$ size for the network path
- If we decrease b less we don't lose as much bandwidth due to a small congestion window

What exactly does it do?

- Based on the TCP response function
 - Relates loss and throughput
- Uses the TCP response function to investigate certain parameters
 - **High_Window, High_Loss**; largest cwnd needed for x throughput and the required loss for that throughput
 - **Low_Window, Low_Loss**; smallest cwnd when we actually switch from Standard TCP and the required loss rate for that cwnd size
 - **High_B**; the smallest decrease in b when we are at a large cwnd
- Equations to transform this information into a table for $a(\text{cwnd})$ and $b(\text{cwnd})$

$a(\text{cwnd})$ & $b(\text{cwnd})$



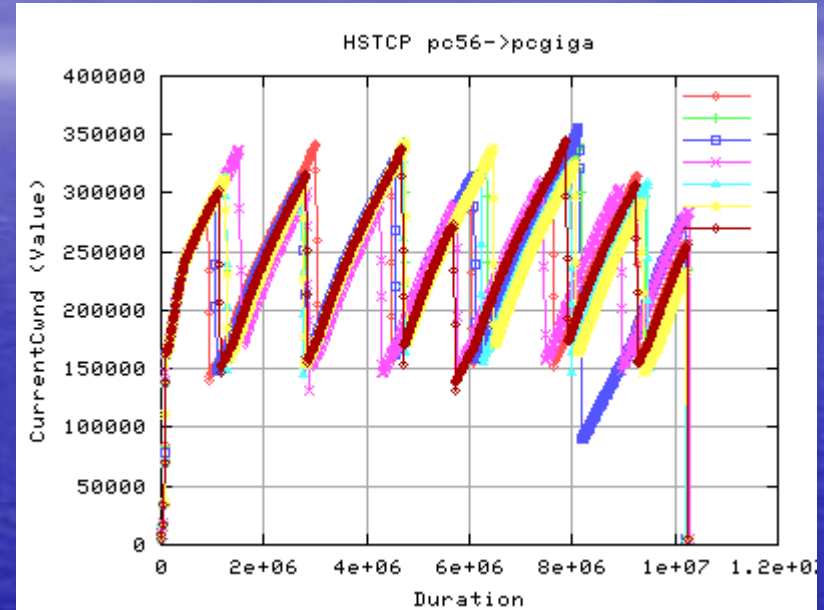
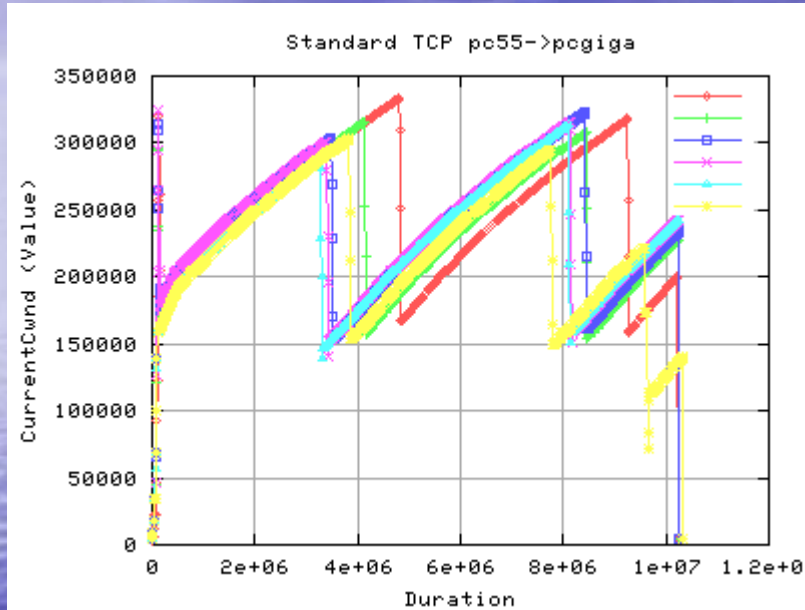
Kernel Modifications

- Focus on investigating phase space of HSTCP variables
- 3 Phase approach
 - Hard code HSTCP as compile-time option – ‘recommended’ HSTCP by Sally Floyd
 - Kernel hooks to enable alteration of HSTCP variables – from user space
 - Allows easier testing/exploration
- Only a few changes necessary:
 - Code for calculating the $a(\text{cwnd})$ and $b(\text{cwnd})$ values
 - Modification of existing code for changing cwnd (during the Congestion Avoidance phase only)
- Implementation on kernel 2.4.16 with web100 (currently alpha1.1)

a(cwnd) & b(cwnd)

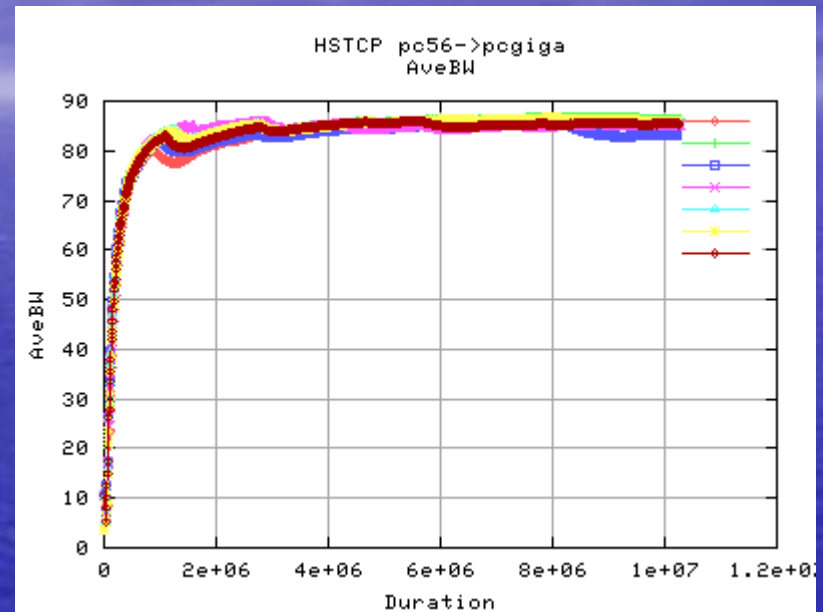
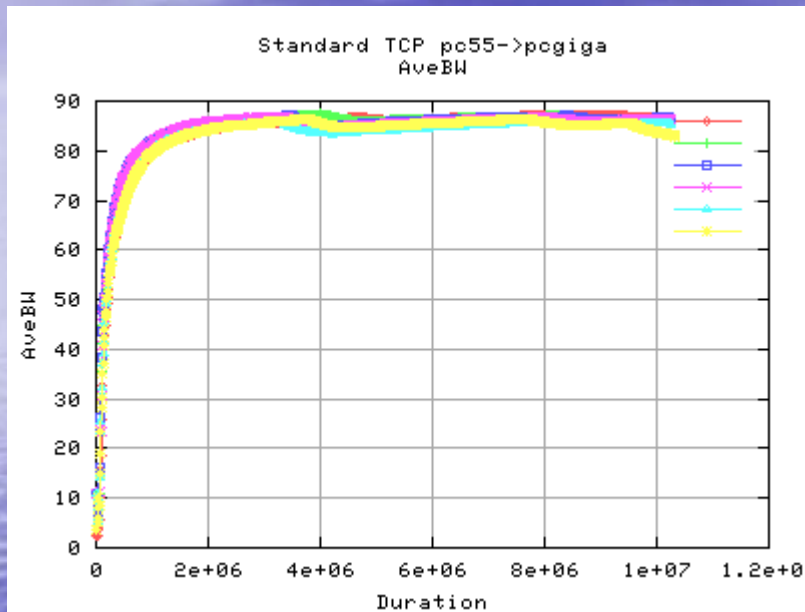
- a(cwnd) and b(cwnd) implement via a static look-up table
- Using access function `get_hstcp_val`
- Changes to congestion avoidance algorithm required only slight modification in `tcp_output.c`

UCL->CERN



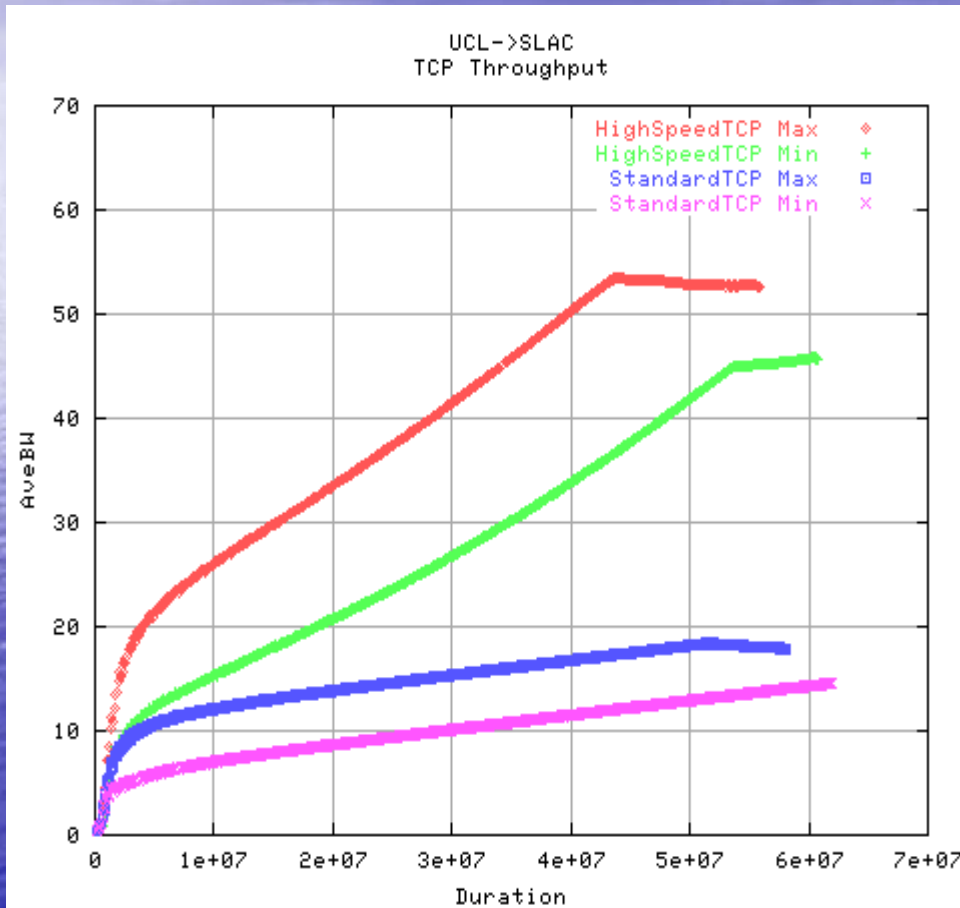
- 100mbit link from UCL->CERN
- Saw tooth increase is greater!
- Ie a is larger
- Limited by 100mbit link... (bandwidth delay \sim 180kbytes)

UCL->CERN



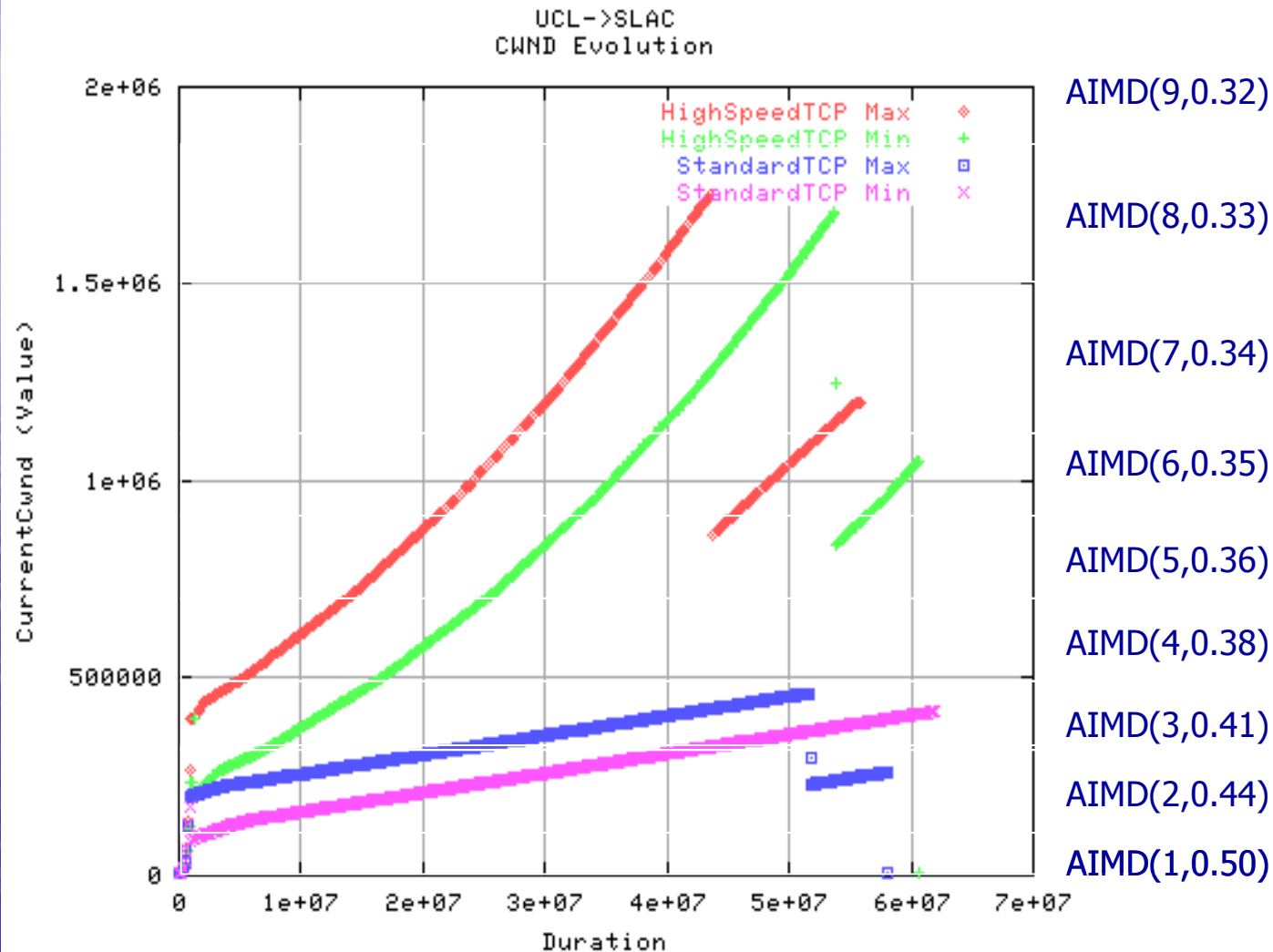
- RTT: 20ms
- Not much difference!
- No clear advantage to using HSTCP for this link at this speed (100mbit)!

UCL->SLAC



- Graph shows min and max of about 8 separate flows
- RTT: 135ms
- Standard TCP performance poor
- HighSpeedTCP gives performance boost of about 2.5!

UCL->SLAC



Performance Issues

- Slow Start
 - High drop rates when we send too much out during Slow Start
 - Means that we have to linearly increment cwnd
 - As HSTCP has a much steeper increase, we get up to a large cwnd faster

Controlling Loss Rates

- Difficult to control what is happening on real life networks
- Need to be able to control packet drops to quantify effects of TCP
- Need facility for Sender side to 'unacknowledged' acknowledgements
 - Artificial packet drops
- Easier to test as only requires Sender side (already modified with HSTCP) to be changed
- Investigate into
 - Constant drop rates – unrealistic but easy to implement
 - Statistically predictable drop rates
- Can analyse implementation of b parameter easier!

Test Plan

- Need to verify $a(cwnd)$ and $b(cwnd)$ are functioning correctly
- Comparison with Theoretical Results with Network Simulator by Evandro de Souza @ Berkely Lab
- Investigation of 'recommended' HSTCP
 - Quantify advantages over StandardTCP
 - Investigate into Fairness
- Investigation of RTT independence of HSTCP
 - Proposed by Tom Kelly & Glenn Vinnicombe
- Comparison against Multiple Stream TCP
 - If we can do it with Multiple Stream TCP, why bother with HSTCP?
- Investigation into Limited Slow Start
 - Important if we're using large cwnds (UCL->SLAC) to better guess the network state at start up

Conclusions

- Not much use for low latency, low throughput networks
- Much improved for high latency, high throughput networks
- Important!
 - Fairness (and hence internet stability) needs much investigation
 - Comparisons to existing methods (ie Multistream TCP)