H-TCP: TCP for high-speed and long-distance networks

D.J.Leith, R.N.Shorten
Hamilton Institute
Ireland

High-speed Networks

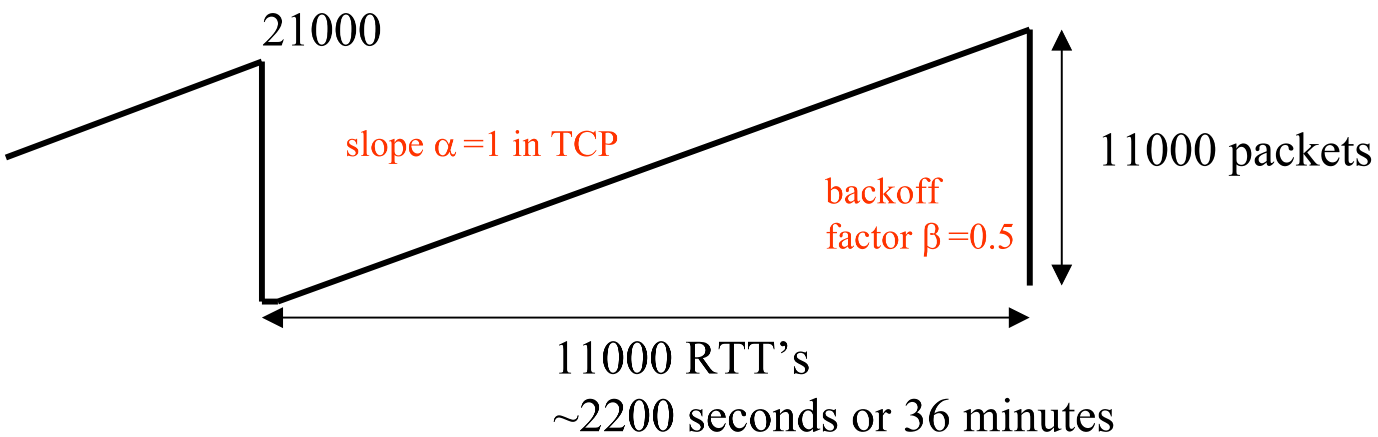
The pipe size of a link is roughly $BT+q_{max}$

where B is the the link rate (packets/s), T is the propagation delay and q_{max} is the queue size.

On a long distance gigabit link, B=100,000 packets/s, T=200ms, q_{max}=1000 and

$$BT+q_{max}=21,000$$

Note that the pipe size determines the peak window size of a TCP source.



- TCP becomes sluggish, and requires v.low drop rate to achieve reasonable throughput.

High-speed Networks

Rather than a complete redesign of TCP, is it possible to devise a small modification that fixes it in high-speed networks ?

Simply making the increase parameter α larger is inadmissable – on low-speed networks we require backward compatibility with current sources.

Large α in high-speed regimes, $\alpha=1$ in low-speed regimes suggests some sort of mode switch.

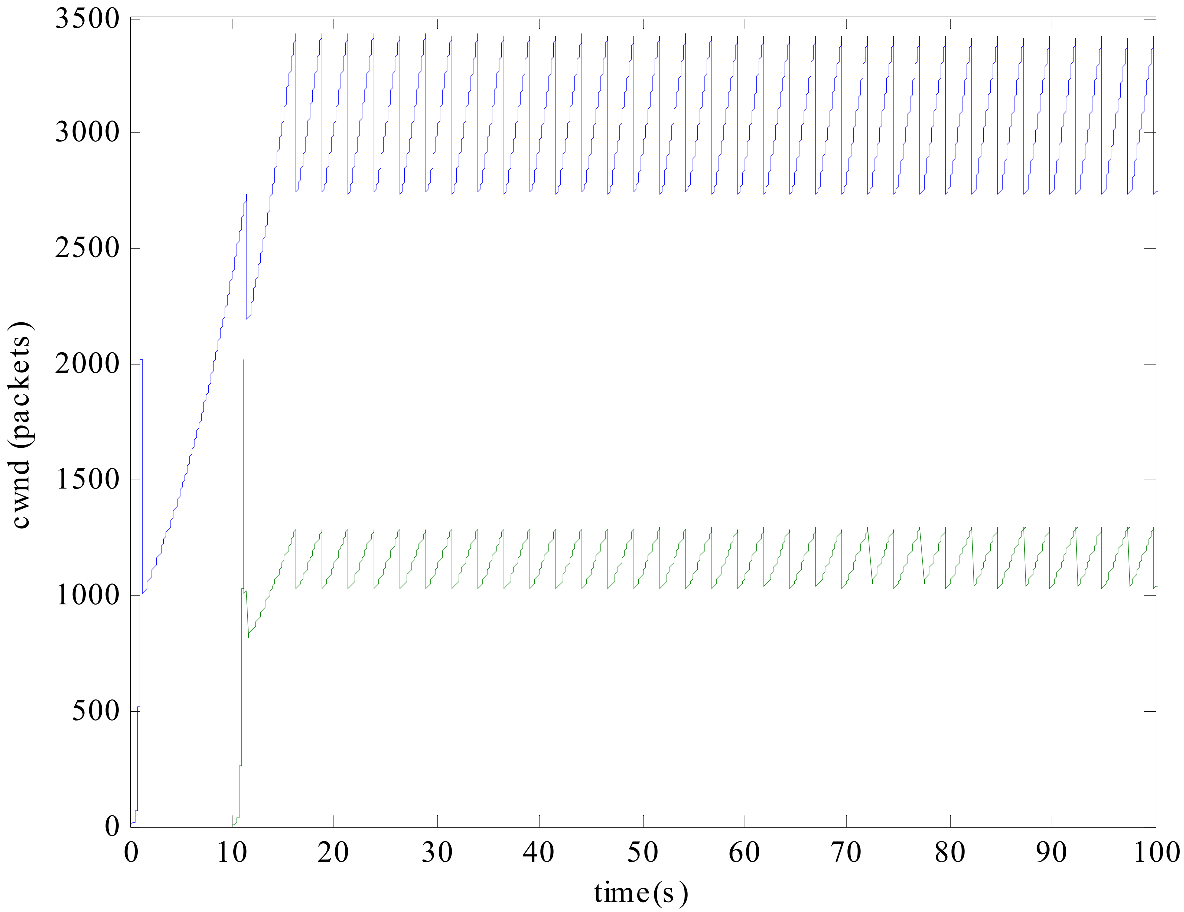
One approach – Scalable TCP – use a multiplicative increase rule and smaller backoff parameter β.

Another – HS-TCP – is to vary AIMD parameters as a fiunction of *cwnd* …

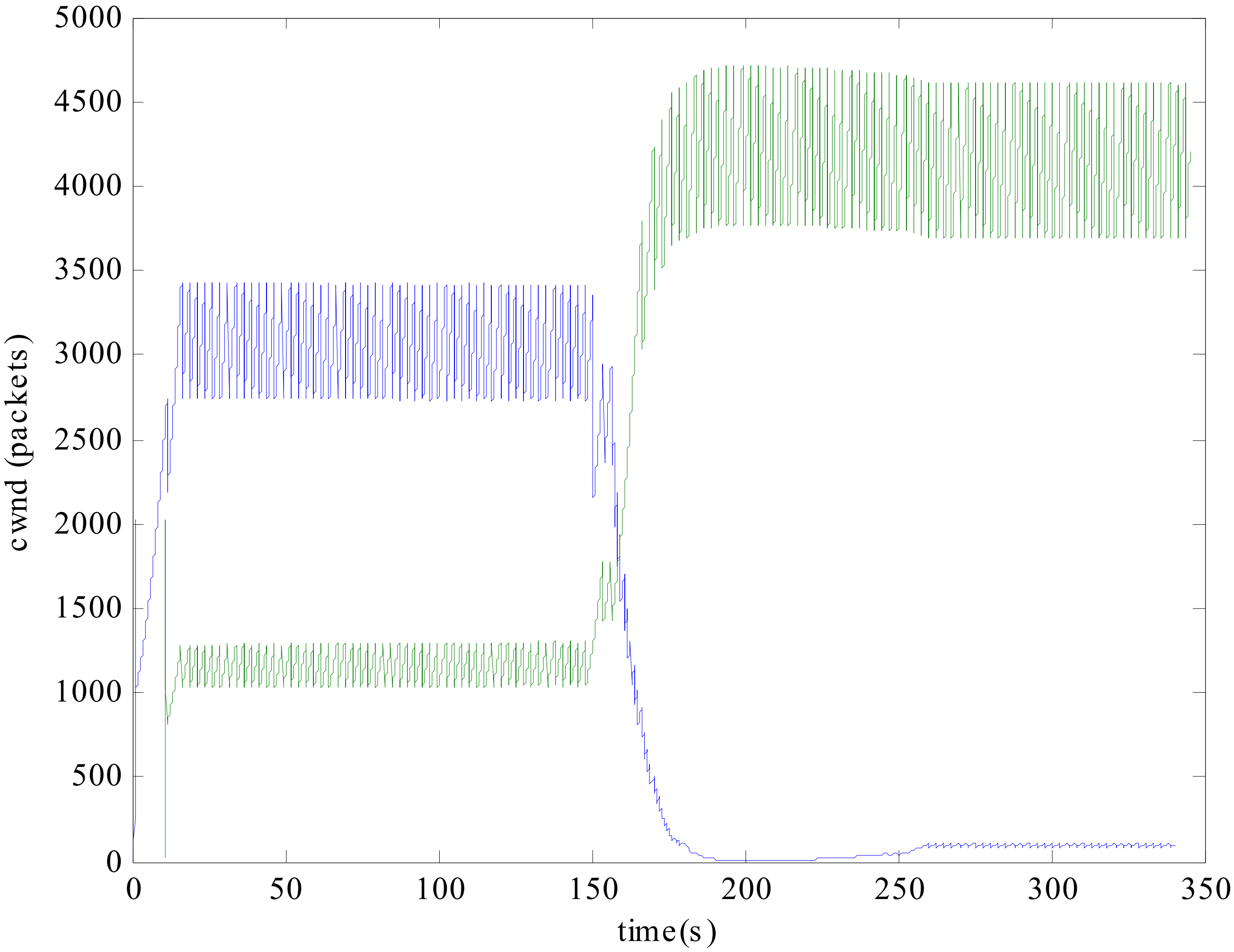
(increase α, decrease β as *cwnd* becomes large)

High-speed Networks **Scaleable TCP**

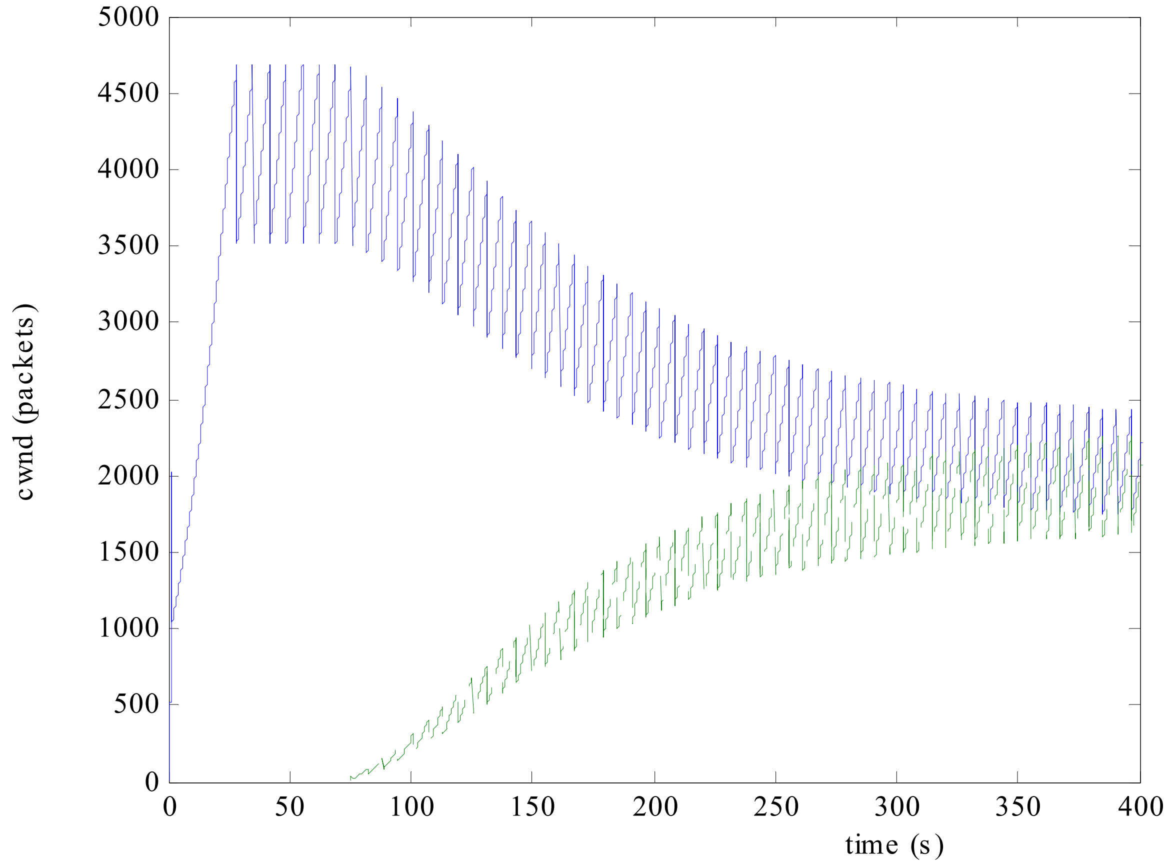
Scaleable TCP has convergence issues …



High-speed Networks Scaleable TCP



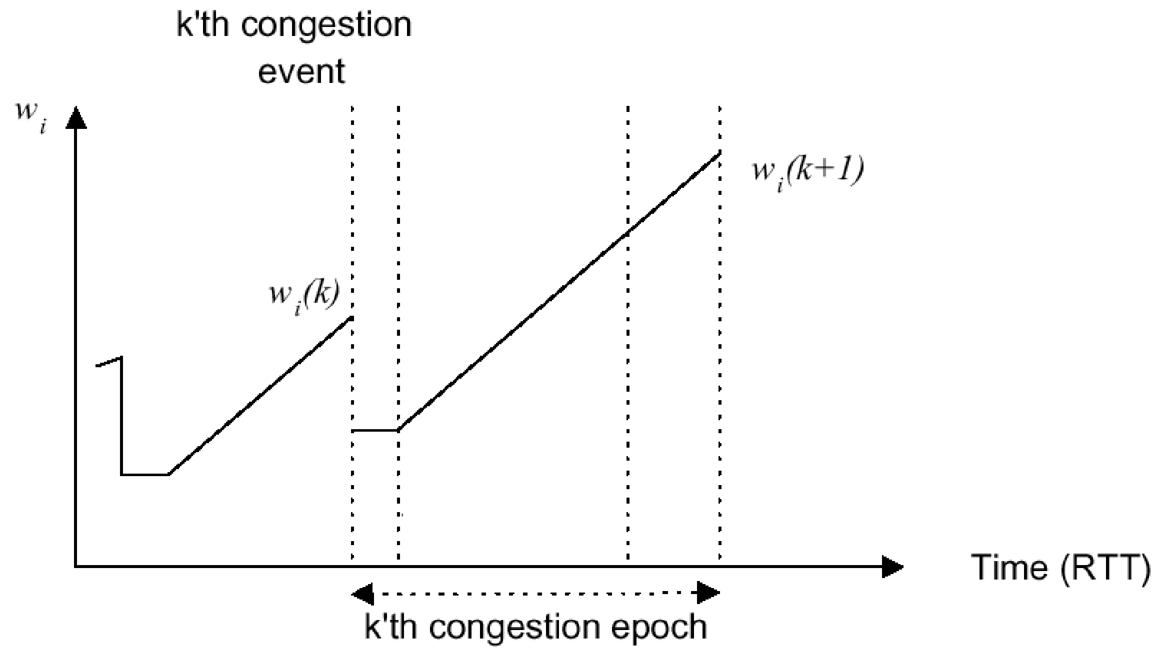
High-speed Networks HS-TCP also has convergence issues



Example of two HS-TCP flows - the second flow experiences a drop early in slow-start focussing attention on the responsiveness of the congestion avoidance algorithm.

(NS simulation: 500Mb bottleneck link, 100ms delay, queue 500 packets)

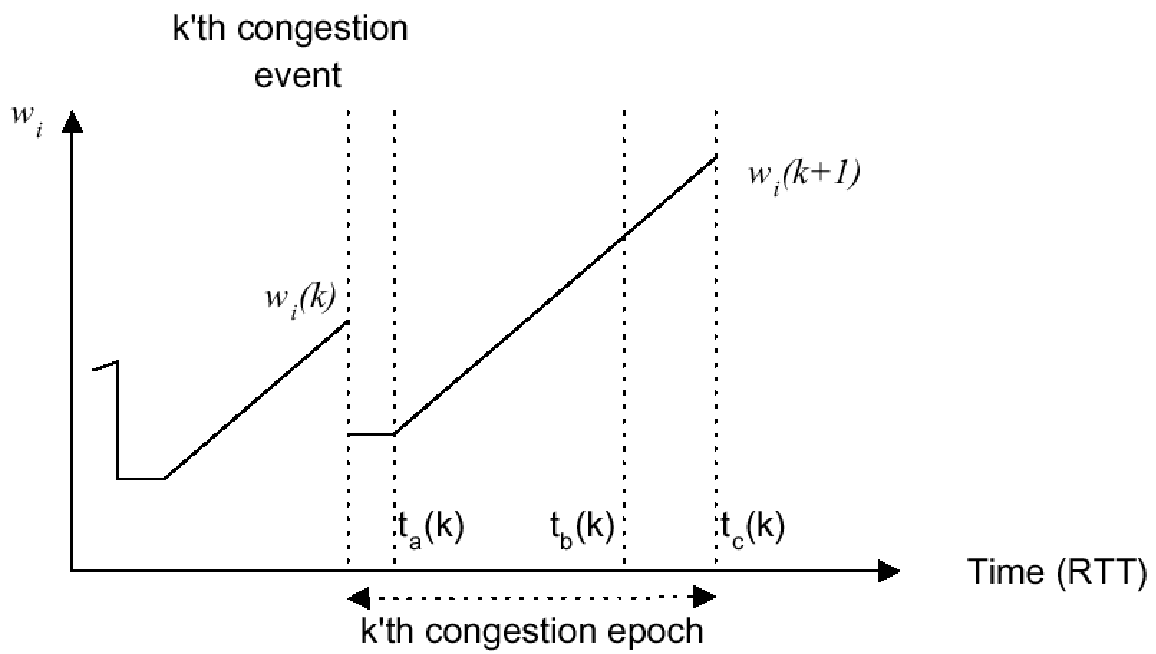
Current TCP congestion control algorithm revisted.



Note: *cwnd* never converges to a steady value with this probe/back-off approach. Also, we are ignoring slow-start, timout's etc here so as to focus on the congestion avoidance behaviour.

Synchronisation Model

Typical congestion window evolution for a TCP source in congestion avoidance:



Synchronisation assumption: t_a, t_b, t_c are the same for all sources.
e.g. when a shared bottleneck link, RTT is the same for all sources, each source transmits at least one packet every RTT ($\alpha \geq 1$)

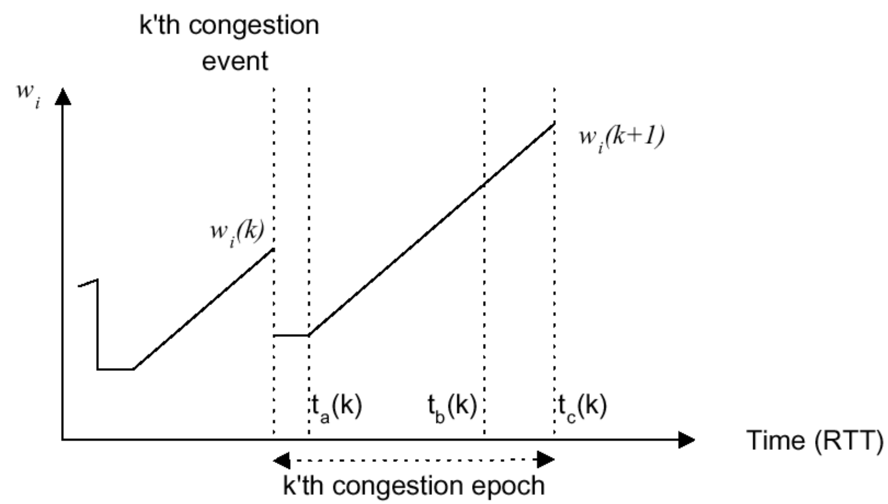
Synchronisation Model

The source congestion windows are subject to constraints:

$$w_i \geq 0, \sum_{i=1}^{n} w_i = P + \sum_{i=1}^{n} \alpha_i$$

Number of packets in pipe is non-negative

At congestion, total number of packets in pipe matches pipe size, P



For source *i* we have:

$$\begin{aligned} w_i(k+1) &= \beta_i w_i(k) + \alpha_i [t_c(k) - t_a(k)] \\ t_c(k) - t_a(k) &= \frac{1}{\sum_{i=1}^{n} \alpha_i}[P - \sum_{i=1}^{n} \beta_i w_i(k)] + 1 \end{aligned}$$

Synchronisation Model

Collecting the evolution equations for all n sources yields the network dynamics:

$$W(k+1) = AW(k)$$

where $W^T(k) = [w_1(k), \cdots, w_n(k)]$ is the vector of window sizes at congestion and

$$A = \begin{bmatrix} \beta_1 & 0 & \cdots & 0 \\ 0 & \beta_2 & 0 & 0 \\ \vdots & 0 & \ddots & 0 \\ 0 & 0 & \cdots & \beta_n \end{bmatrix} + \frac{1}{\sum_{j=1}^{n} \alpha_i} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \cdots \\ \alpha_n \end{bmatrix} \begin{bmatrix} 1-\beta_1 & 1-\beta_2 & \cdots & 1-\beta_n \end{bmatrix}$$

where α_i is the AIMD increase parameter for source i, β_i the decrease parameter.

Observe that:

- The dynamics are linear
- **A** is a positive matrix with very special structure
- This model incorporates important network features such as the hybrid nature of AIMD, time-varying delay and drop-tail queueing.

Synchronisation Model

Analysis *A network of synchronised AIMD sources:*

(i) *possesses a unique stationary point,* $W_{ss}=\Theta x_p$ *where* Θ *is a positive constant and*

(ii) *the stationary point is globally exponentially stable. The rate of convergence depends on the second largest eigenvalue of A.*

Fairness

Stationary point: $W_{ss}= \Theta\, x_p$ where Θ is a positive constant and $x_p^T = \gamma[\frac{\alpha_1}{1-\beta_1}, ..., \frac{\alpha_n}{1-\beta_n}]$

$\alpha_i = \lambda(1-\beta_i)\ \forall i$ and for some $\lambda > 0$ $\Rightarrow$ $W_{ss}^T = \Theta/n\ [1, 1, ..., 1]$ i.e. $w_1 = w_2 = ... = w_n$

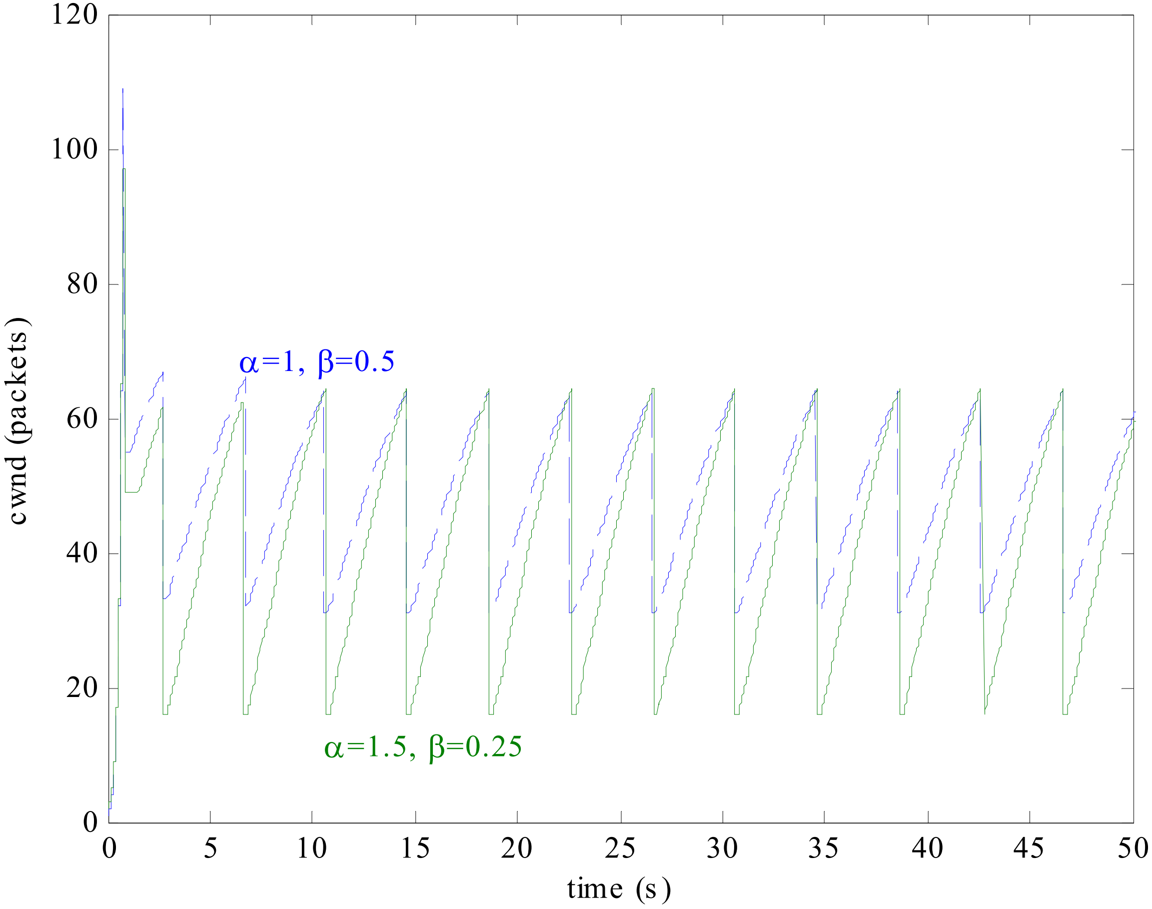
For standard TCP, α=1, β=0.5 so λ=2 and

$$\boxed{\alpha_i=2(1-\beta_i)}$$

is the condition for fair co- existence of AIMD flows with TCP.

Synchronisation Model

Fairness – Example (NS simulation 10Mb link, 100ms delay, queue 40 Packets)



Synchronisation Model

Responsiveness

Special case: All of the sources have the same decrease parameter: $\beta_i = \beta \ \forall i$.

Then the eigenvalues of A (other than the Perron eigenvalue) are equal to β

$\Rightarrow$ rate of convergence is β^k, where k is the congestion epoch.

95% rise time (measured in congestion epochs) is $\log(0.05)/\log \beta$

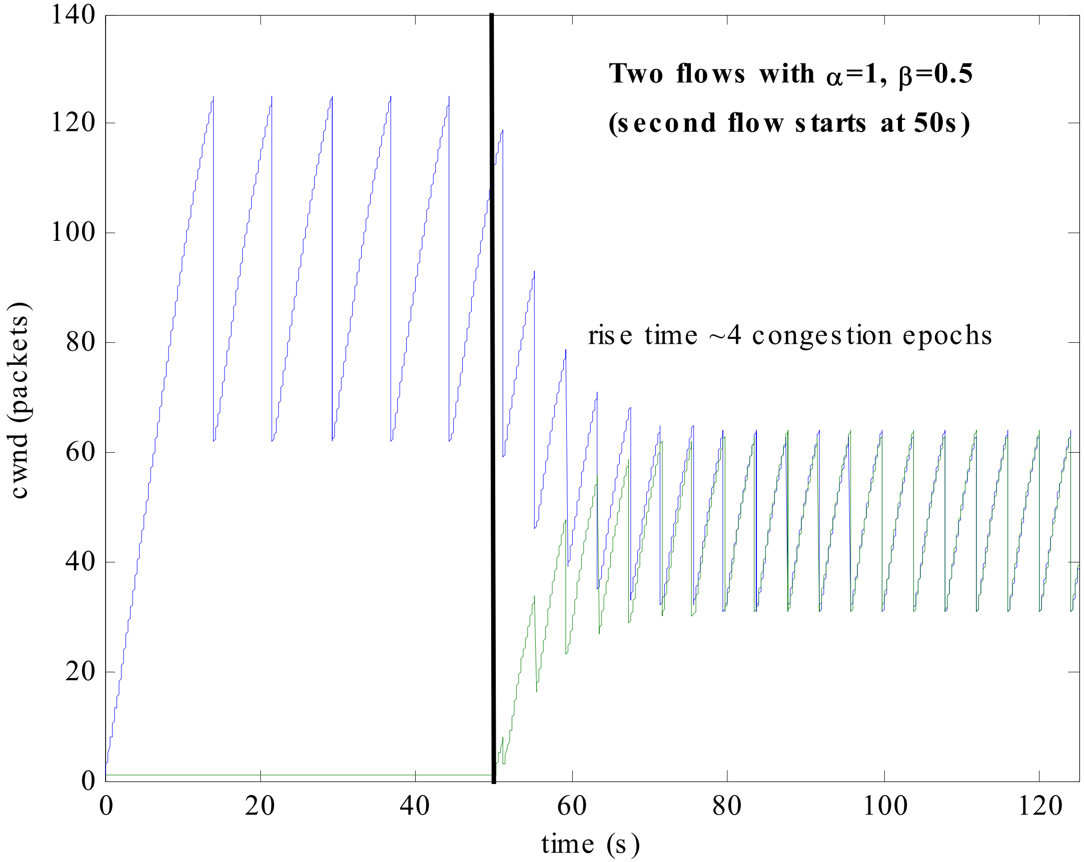
e.g. for β=0.5, rise time is 4 congestion epochs.

Note, *duration* of congestion epochs depends on increase parameters α_i.

Synchronisation Model

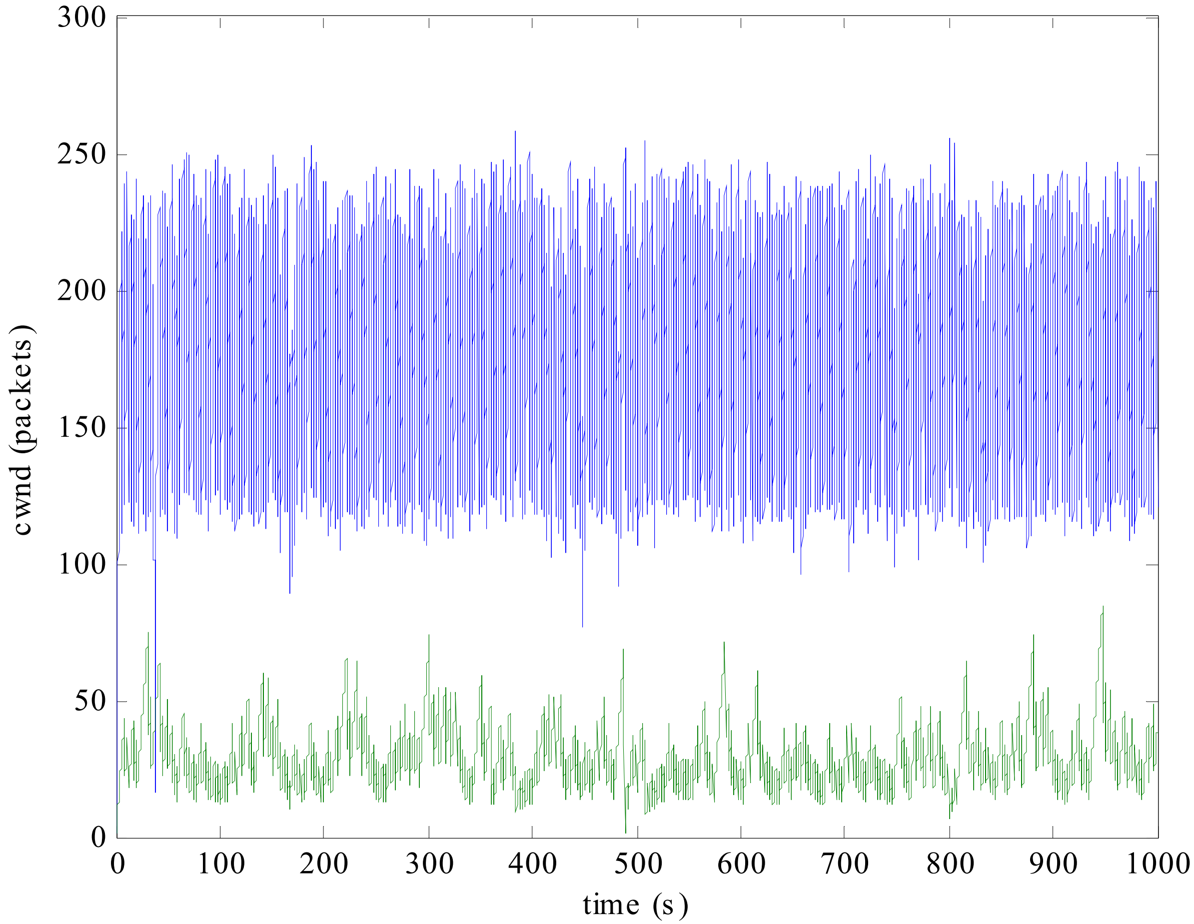
Responsiveness (cont)

e.g.

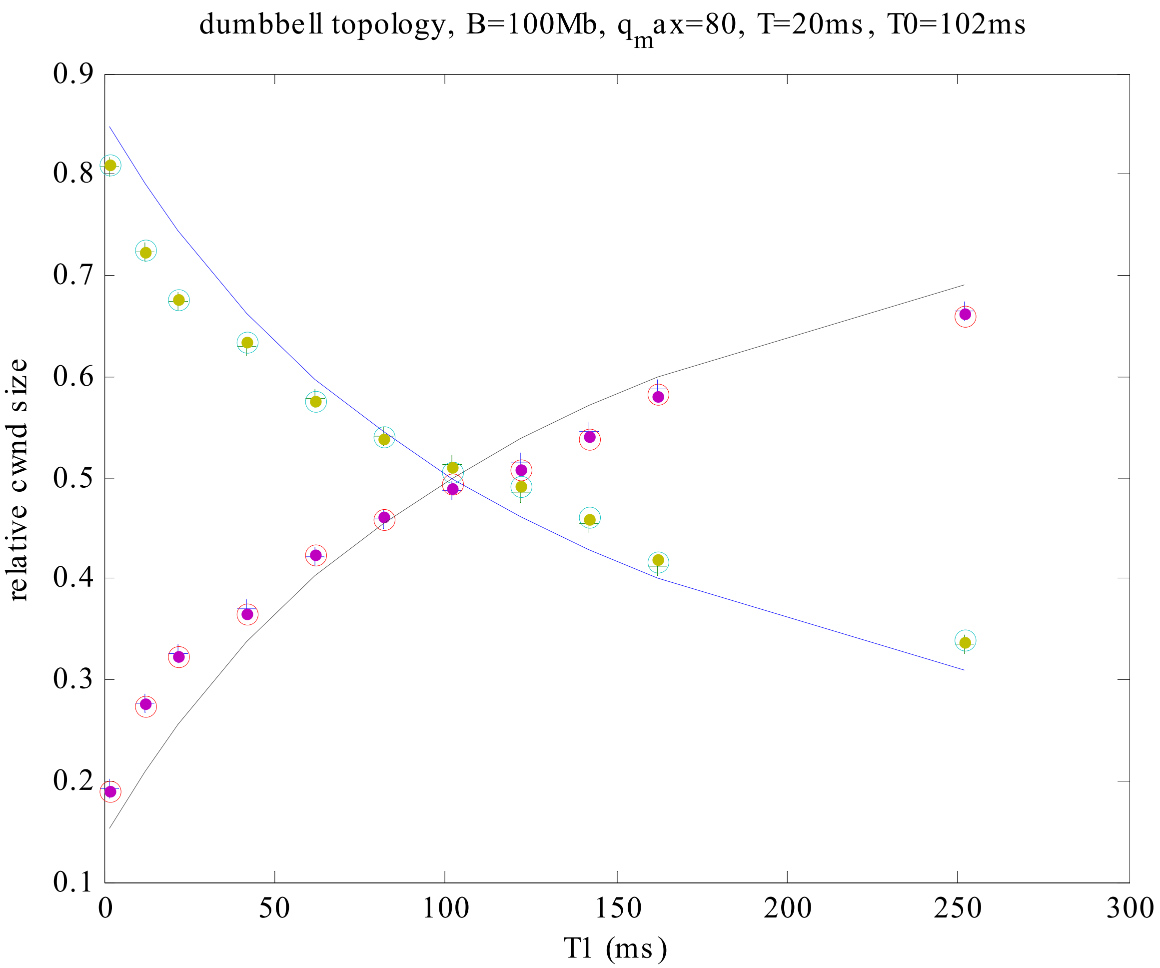


Unsynchronised TCP Flows ?

Example: Congestion window time histories (B=100Mb, T_0=20ms, T_1=2ms, T_2=162ms, queue 80 packets)



Unsynchronised TCP Flows ?



H-TCP

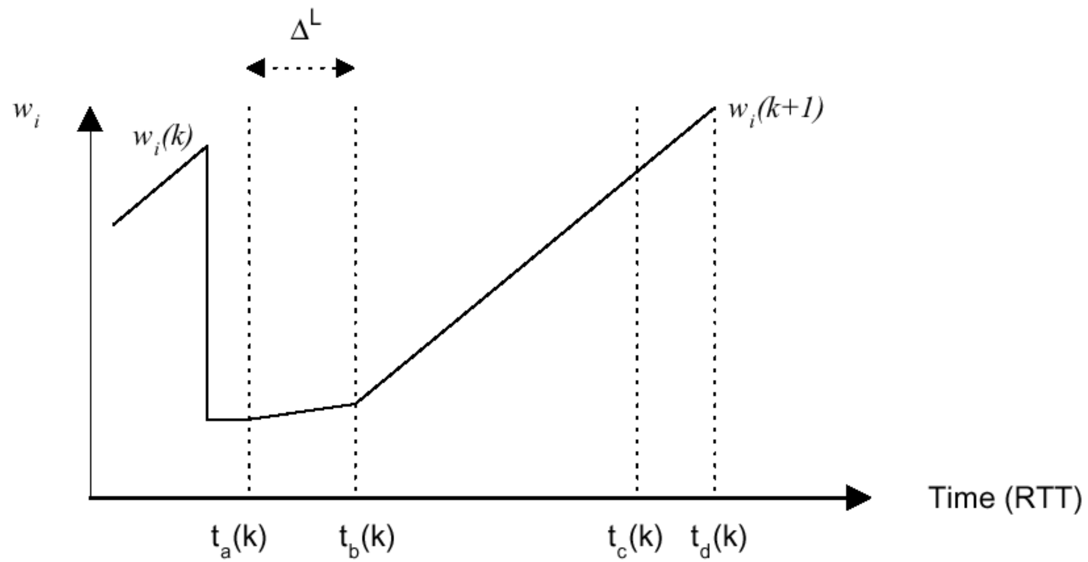
Simply making the increase parameter α larger is inadmissable – on low-speed networks we require backward compatibility with current sources.

Large α in high-speed regimes, α=1 in low-speed regimes suggests some sort of mode switch.

E.g.

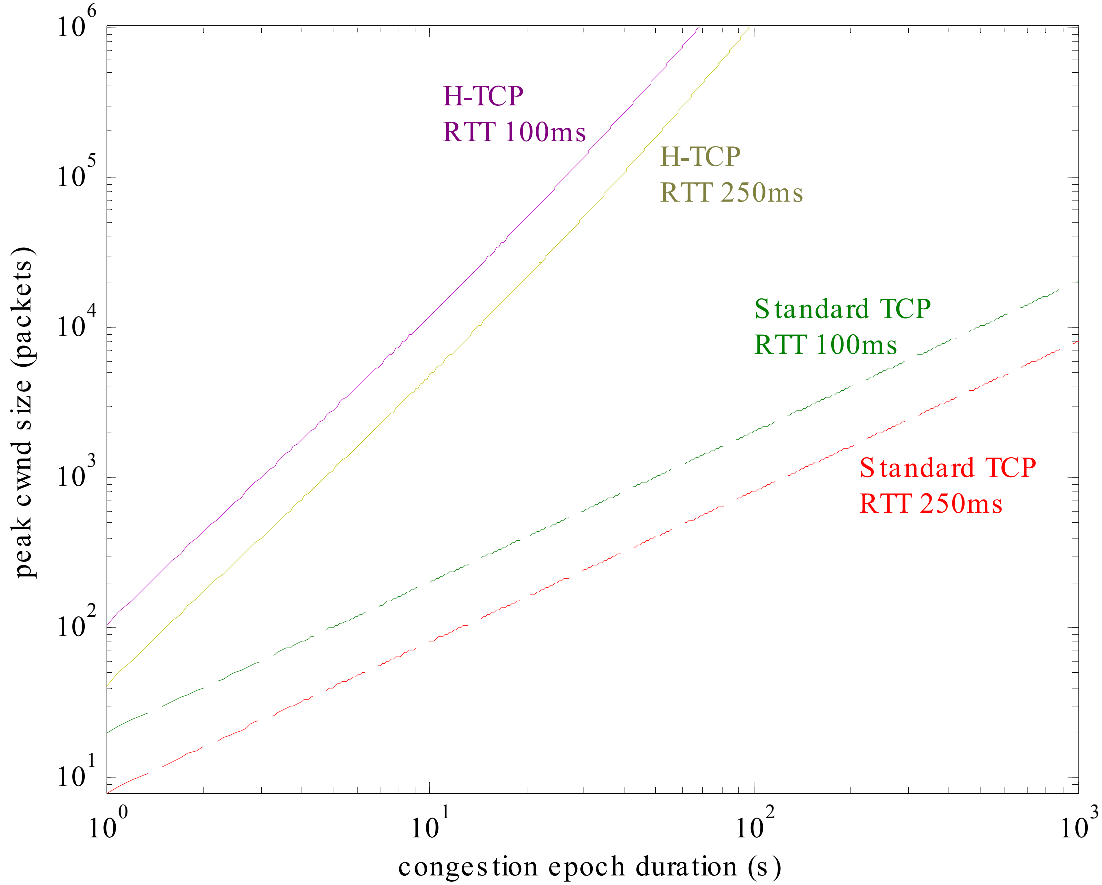
$$\alpha_i = \begin{cases} \alpha_i^L & \Delta_i \leq \Delta^L \\ \alpha_i^H(\Delta_i) & \Delta_i \geq \Delta^L \end{cases}$$

where Δ_i is the time since the last backoff.



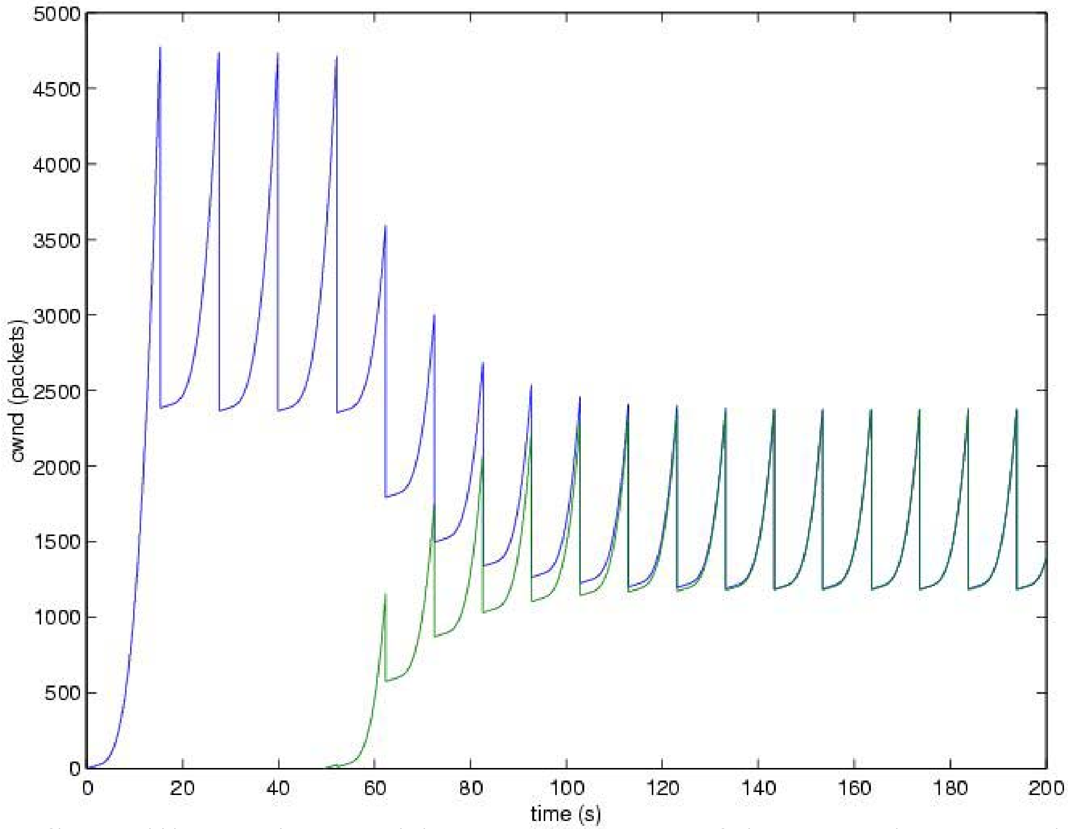
H-TCP

$$\alpha_i^H(\Delta_i) = 1 + 10(\Delta_i - \Delta^L) + \left(\frac{\Delta_i - \Delta^L}{2}\right)^2$$



H-TCP

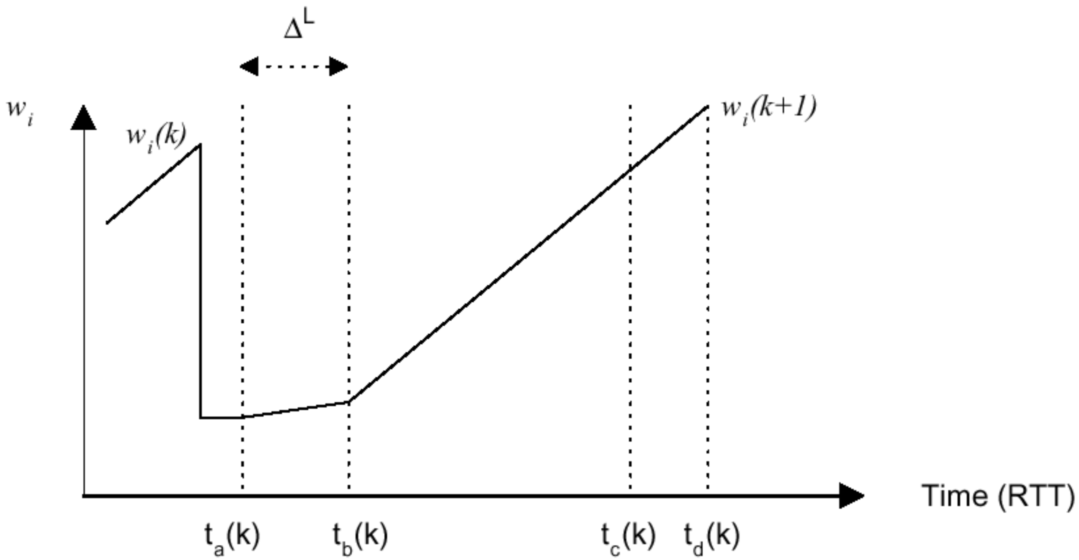
Rate of convergence



Example of two H-TCP flows illustrating rapid convergence to fairness - the second flow experiences a drop early in slow-start focusing attention on the responsiveness of the congestion avoidance algorithm.

(NS simulation: 500Mb link, 100ms delay, queue 500 packets; H-TCP parameters: α^L=1, α^H=20, β=0.5, Δ^L=19 – corresponding to window size threshold of 38)

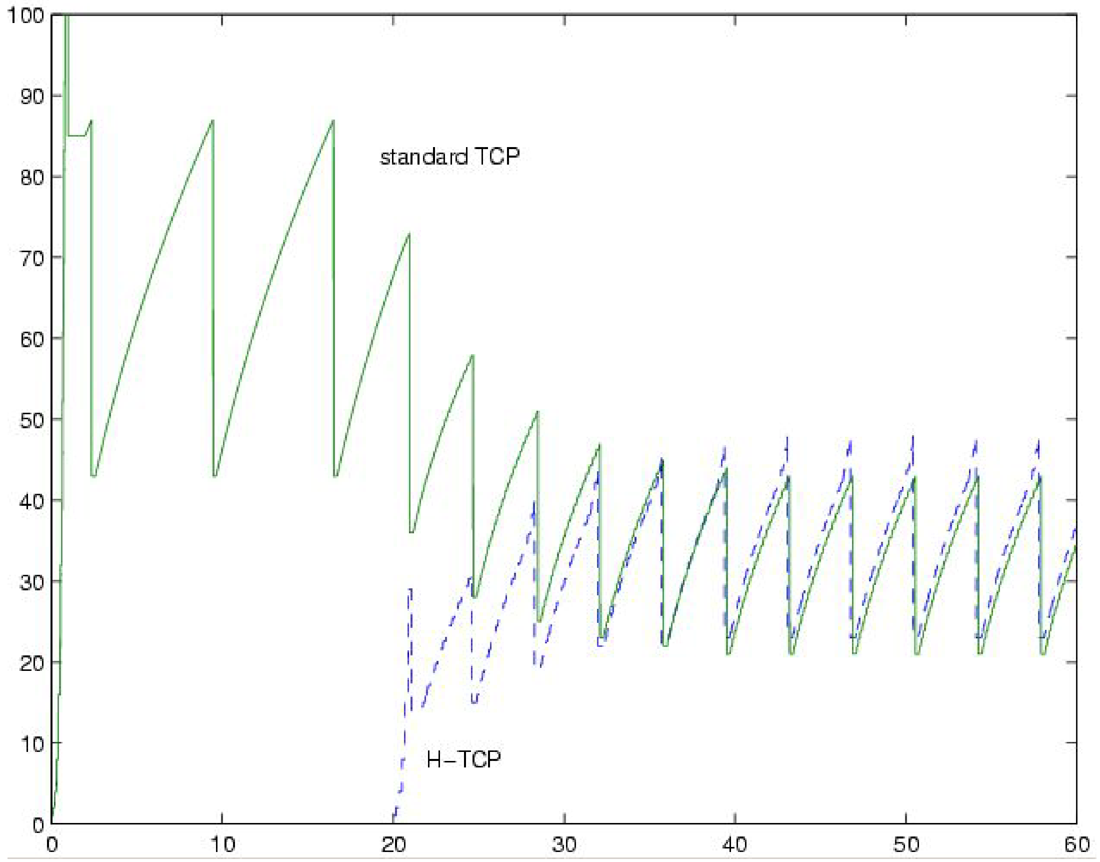
Backward compatibility



On low-speed links where duration of congestion epoch is less than Δ^L, H-TCP is identical to standard TCP.

As the duration increases above Δ^L, the effective α of H-TCP increases and so does the degree of unfairness with standard TCP.

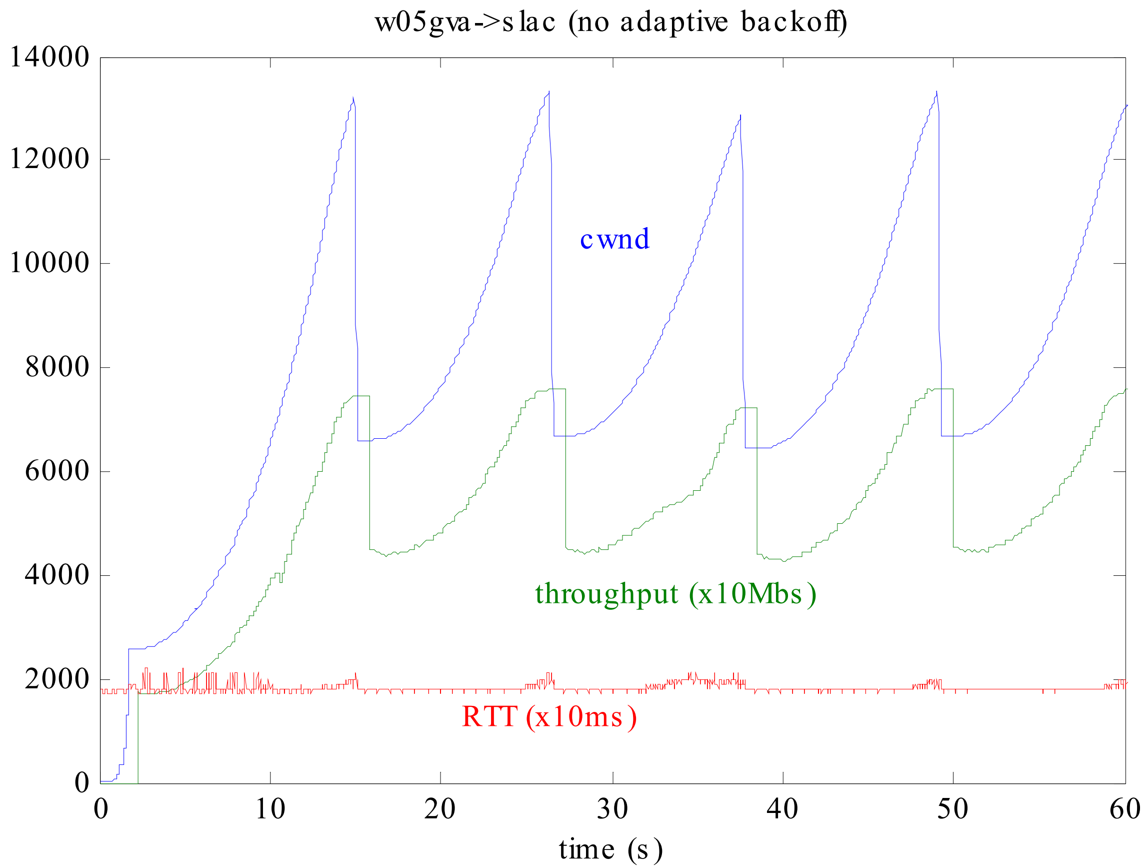
H-TCP



Example of standard TCP and H-TCP flows co-existing on a low speed link

(NS simulation, network parameters: 5Mb link, 100ms delay, queue 44 packets; H-TCP parameters: α^L=1, α^H=20, β=0.5, Δ^L=19 – corresponding to window size threshold of 38)

H-TCP **Adaptation to achieve efficient bandwidth utilisation**



H-TCP Adaptation to achieve efficient bandwidth utilisation

At congestion, the bottleneck link is operating at capacity and the overall throughput is given by:

$$R(k)^{-} = \sum_{i}^{n} \frac{w_i(k)}{RTT_{max,i}}$$

where w_i is the window size of source i at congestion and $RTT_{max,i}$ is BT_i+q_{max}. After backoff, the overall throughput is

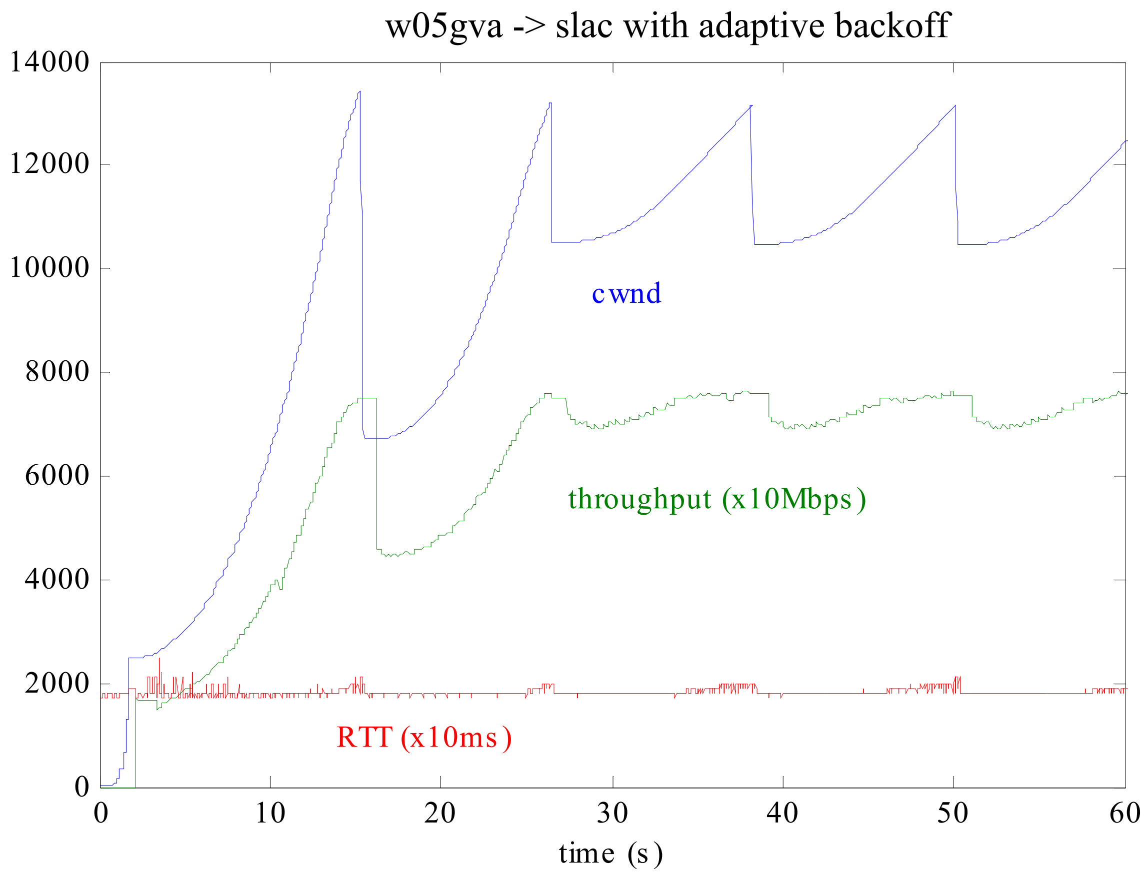
$$R(k)^{+} = \sum_{i}^{n} \frac{\beta_i w_i(k)}{RTT_{min,i}}$$

where $RTT_{min,i}$ is BT_i assuming the queue empties at backoff.

Simple approach is to equate both rates by using backoff factor

$$\beta_i = \frac{RTT_{min,i}}{RTT_{max,i}}$$

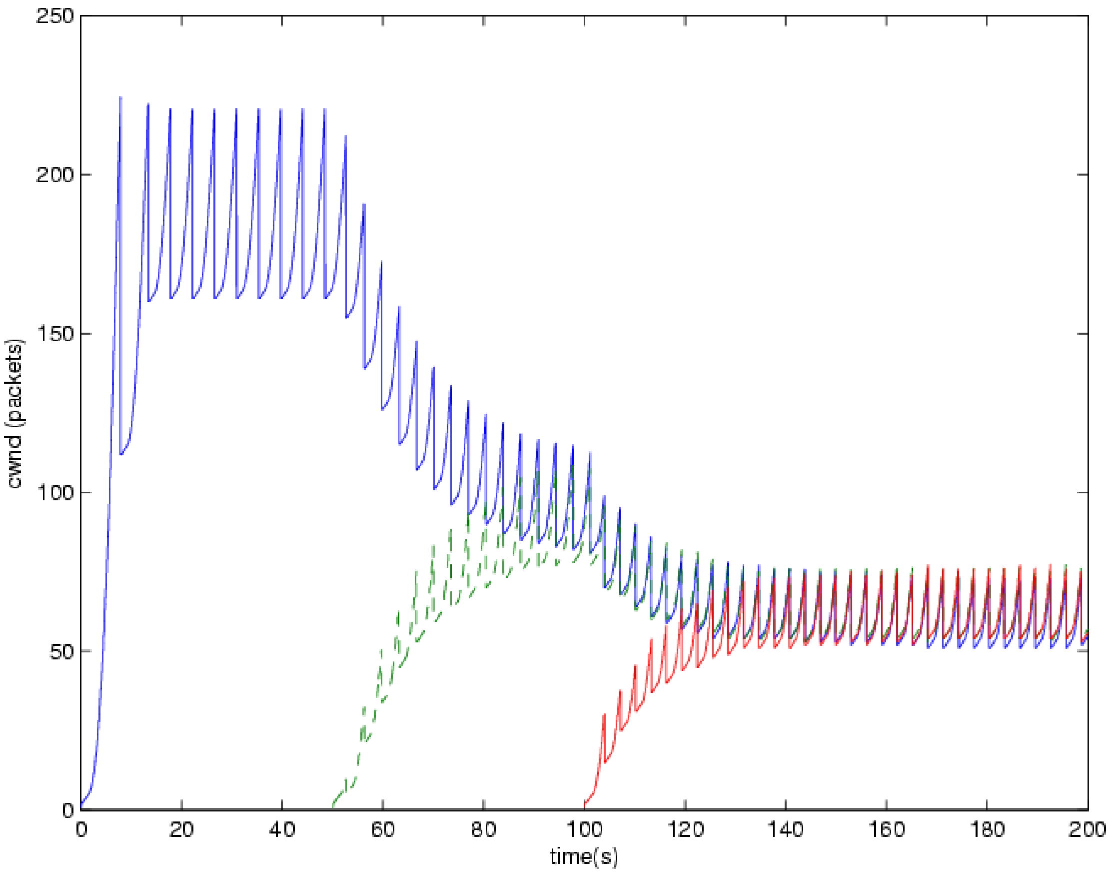
H-TCP **Adaptation to achieve efficient bandwidth utilisation**



Note: for prudence we restrict the backoff factor β to lie in the interval [0.5, 0.8] here – in this example a backoff factor >0.8 is needed to completely prevent the queue emptying.

H-TCP **Adaptation to achieve responsiveness**

Our previous analysis indicates that as the backoff factor β is increased the responsiveness of AIMD-like flows becomes more sluggish. For β-=0.5, the 95% rise time is 4 congestion epochs; for β=0.8 it is 13 epochs; for β=0.9 it is 28 epochs.



H-TCP Adaptation to achieve responsiveness

By clamping $\beta<0.8$, we ensure that the convergence time is no more than 13 congestion epochs.

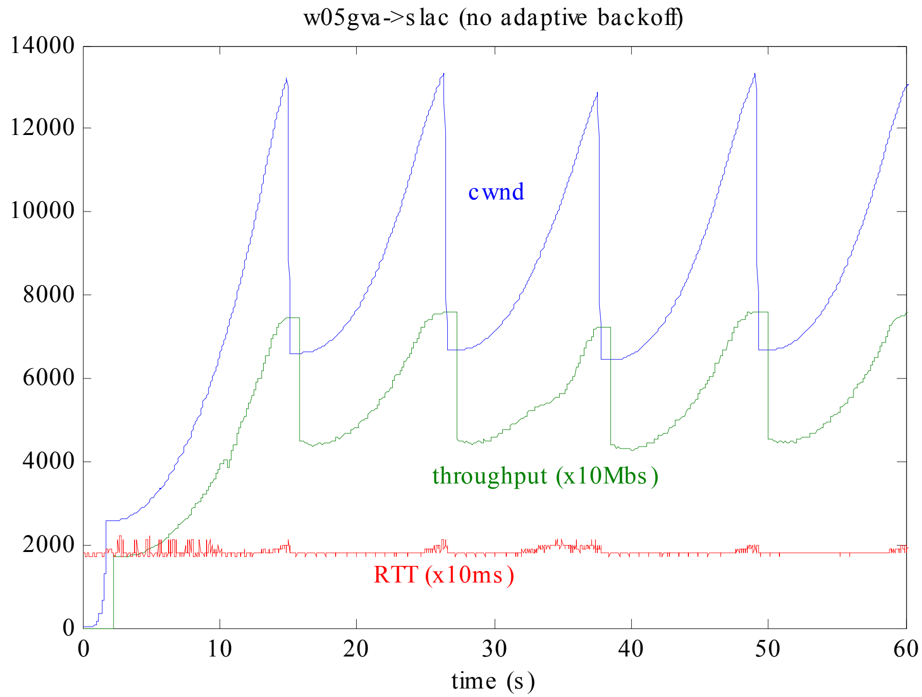
Is this ok or is it too long ?

If too long, it is straightforward to adapt the source back-off factors to reflect the need to respond rapidly to changes in network conditions or to utilise bandwidth efficiently.

We need a network quantity that changes sensibly during disturbances and which can be used to trigger an adaptive reset that adjusts the β_i to ensure responsiveness …

… we consider $\bar{B}_i^{max}$ the maximum throughput achieved over a congestion epoch (throughput is obtained by averaging packets acknowledged over an RTT).

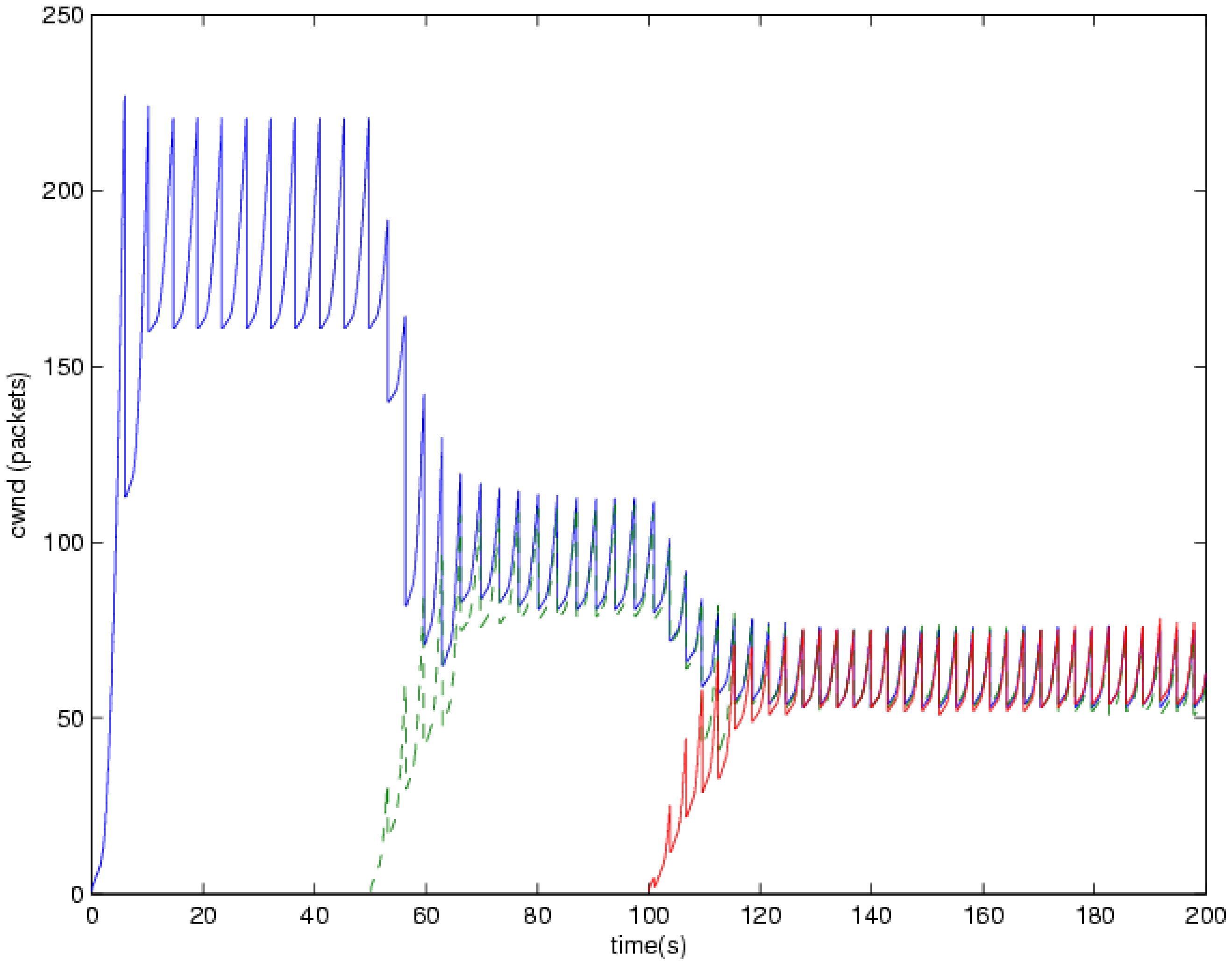
H-TCP **Adaptation to achieve responsiveness**



On each congestion event set:

$$\beta_i(k+1) \leftarrow \begin{cases} 0.5 & \left|\frac{\bar{B}_i^{max}(k+1)-\bar{B}_i^{max}(k)}{\bar{B}_i^{max}(k)}\right| > 0.2 \\ \frac{RTT_{min,i}}{RTT_{max,i}} & \text{otherwise.} \end{cases}$$

H-TCP **Adaptation to achieve responsiveness**



H-TCP **Complete Algorithm**

(a) On each acknowledgement set:

$$\alpha_i \leftarrow \begin{cases} 1 & \Delta_i \le \Delta^L \\ 1 + 10(\Delta_i - \Delta^L) + (\frac{\Delta_i - \Delta^L}{2})^2 & \Delta_i > \Delta^L \end{cases}$$

and then set

$$\alpha_i \leftarrow 2(1 - \beta_i)\alpha_i.$$

(b) On each congestion event set :

$$\beta_i(k+1) \leftarrow \begin{cases} 0.5 & \left|\frac{\bar{B}_i^{max}(k+1) - \bar{B}_i^{max}(k)}{\bar{B}_i^{max}(k)}\right| > 0.2 \\ \frac{RTT_{min,i}}{RTT_{max,i}} & \text{otherwise.} \end{cases}$$

Δ_i is the time elapsed since the last congestion event

$\bar{B}_i^{max}$ maximum throughput (where throughput is averaged over an RTT) achieved over a congestion epoch

H-TCP Complete Algorithm

(a) On each acknowledgement set:

High-speed mode switch
- short congestion epochs, backward compatibility, fairness among flows

$$\alpha_i \leftarrow \begin{cases} 1 & \Delta_i \leq \Delta^L \\ 1 + (\Delta - \Delta^L) + (\frac{\Delta - \Delta^L}{20})^2 & \Delta_i > \Delta^L \end{cases}$$

Ensure fairness regardless of β

$$\alpha_i \leftarrow 2(1 - \beta_i)\alpha_i.$$

(b) On each congestion event set :

Maintain responsiveness

$$\beta_i(k+1) \leftarrow \begin{cases} 0.5 & \left|\frac{\bar{B}_i^{max}(k+1) - \bar{B}_i^{max}(k)}{\bar{B}_i^{max}(k)}\right| > 0.2 \\ \frac{RTT_{min,i}}{RTT_{max,i}} & \text{otherwise.} \end{cases}$$

Efficient utilisation of links with small queues

Implementing H-TCP

•Initial test results – SLAC in late sept/early oct 2003.
•Algorithmic issues – throughput. Revised H-TCP implementation now available.
•Follow up exploratory tests – UCL, SLAC jan/feb 2004

But … evaluating TCP proposals is not so straightforward.

•Major software implementation issues (relevant to any TCP proposal and unrelated to congestion control strategy) can mean that we are not really comparing congestion control *algorithms*.

•How do we design good experiments for TCP proposals to bring out key issues (responsiveness, fairness, friendliness, efficiency etc) over a range of network conditions (which topologies and traffic mixes) ?

e.g. one key issue in testing behaviour of congestion control algorithms is that bottleneck lies in network rather than NIC.

Evaluating TCP proposals

Initial tests – CERN-Chicago.

Bottleneck in NIC and with web100: throughput max's out regardless of congestion avoidance algorithm used.