# HighSpeed TCP and Related Issues:

*

Sally Floyd

November 14, 2002

## Topics:

*

- HighSpeed TCP.
- Quick-Start.

# The Problem: TCP for High-Bandwidth-Delay-Product Networks

*

- Sustaining high congestion windows:

A Standard TCP connection with:

– 1500-byte packets;

– a 100 ms round-trip time;

– a steady-state throughput of 10 Gbps;

would require:

– an average congestion window of 83,333 segments;

– and at most one drop (or mark) every 5,000,000,000 packets (or equivalently, at most one drop every 1 2/3 hours).

This is not realistic.

## Is this a pressing problem in practice?

$*$

- Nope. In practice, users do one of the following:
  – Open up $N$ parallel TCP connections; or
  – Use MulTCP (roughly like an aggregate of $N$ virtual TCP connections).

- However, we can do better:
  – Better flexibility (no $N$ to configure);
  – Better scaling (with a range of bandwidths, numbers of flows);
  – Better slow-start behavior;
  – Competing more fairly with current TCP
(for environments where TCP is able to use the available bandwidth).

## The Solution Space:

*

- At one end of the spectrum:

Simplier, more incremental, and more-easily-deployable changes to the current protocols:

  – HighSpeed TCP (TCP with modified parameters);
  – QuickStart (an IP option to allow high initial congestion windows.)

- At the other end of the spectrum:

More powerful changes with a new transport protocol, and more explicit feedback from the routers?

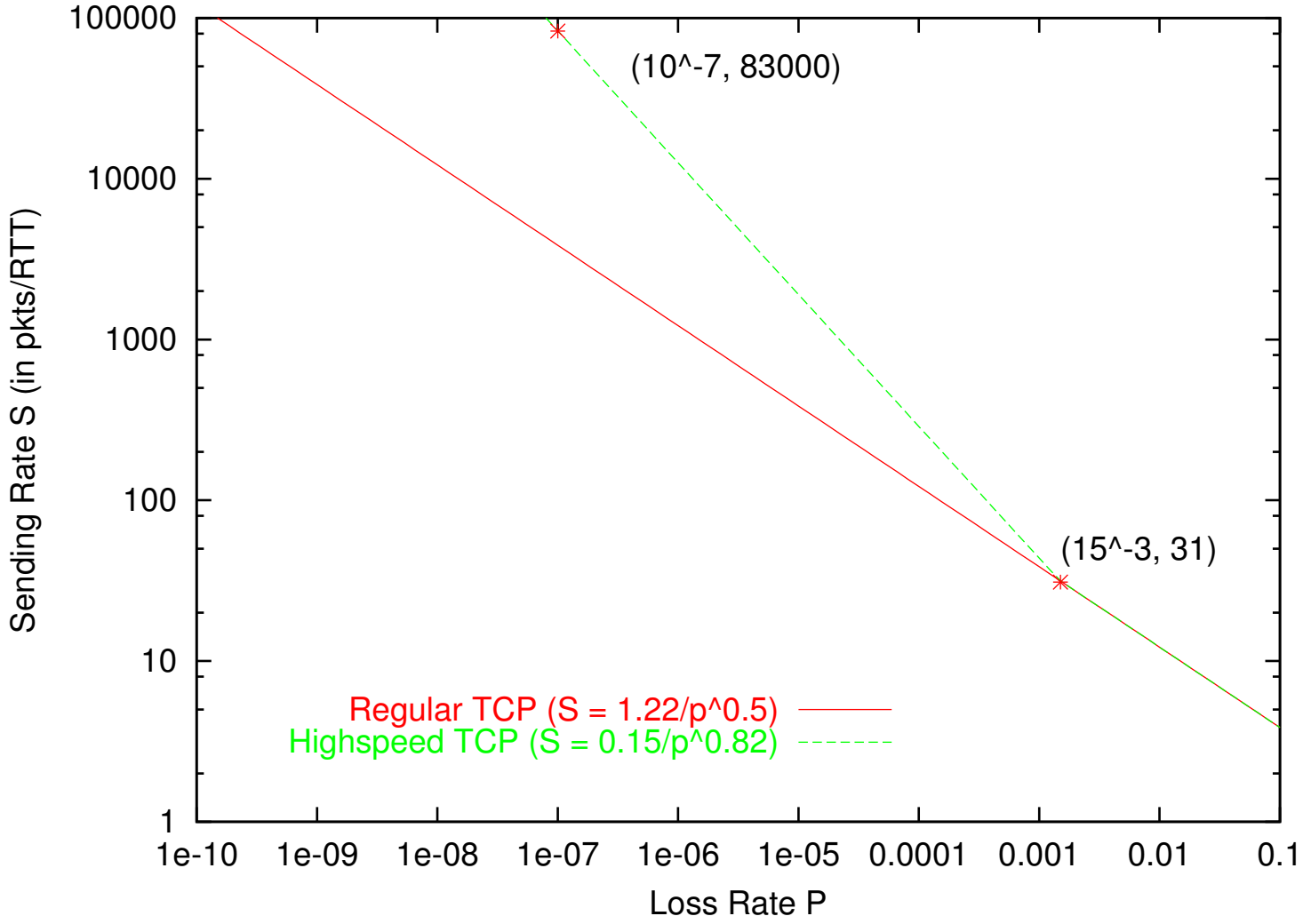
- And other proposals along the simplicity/deployability/power spectrums.

## What is HighSpeed TCP:

$*$

- Just like Standard TCP when cwnd is low.
- More aggressive than Standard TCP when cwnd is high.
  – Uses a modified TCP response function.
- HighSpeed TCP can be thought of as behaving as an aggregate of $N$ TCP connections at higher congestion windows.
- Joint work with Sylvia Ratnasamy and Scott Shenker, additional contributions from Evandro de Souza, Deb Agarwal, Tom Dunigan.

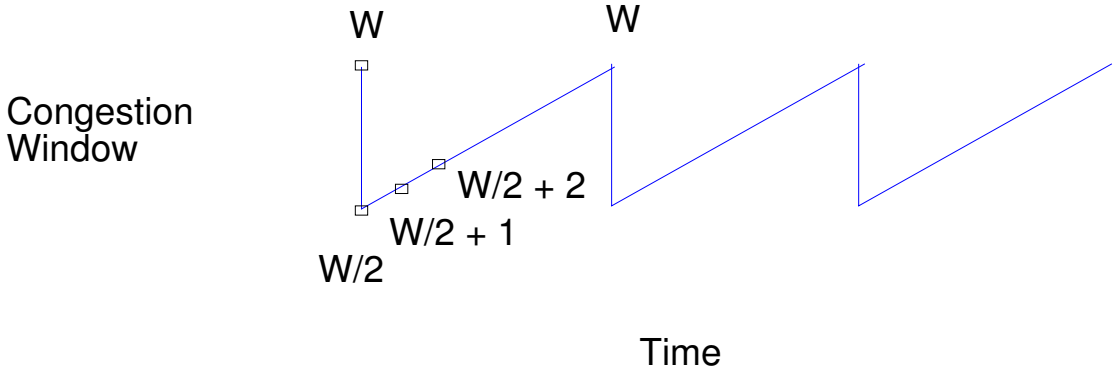
# T HighSpeed TCP: the modified response function.



# Digression: The derivation of the TCP response function:

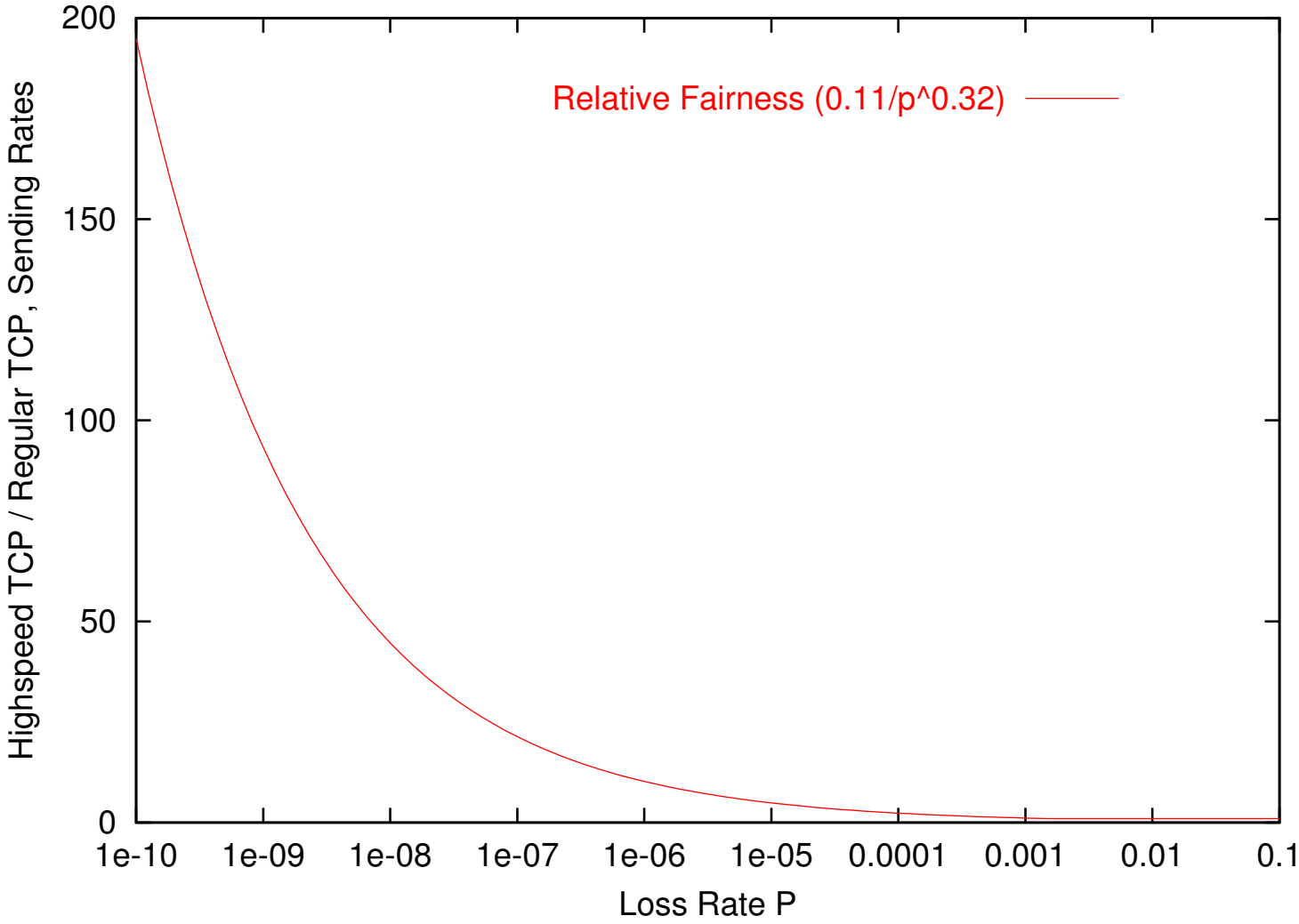
*

- The steady-state model:




- The average sending rate $S$ is $\frac{3}{4}W$ packets per RTT.
- Each cycle takes $\frac{W}{2}$ RTTS, with one drop in $\approx \frac{3}{8}W^2$ packets.
- Therefore, $p \approx \frac{1}{\frac{3}{8}W^2}$, or $S \approx \frac{\sqrt{1.5}}{\sqrt{p}}$, for packet drop rate $p$.

# T HighSpeed TCP: Relative fairness.



## HighSpeed TCP: Simulations in NS.

*

- ./test-all-tcpHighspeed in tcl/test.
- The parameters specifying the response function:
  - Agent/TCP set low_window_ 31
  - Agent/TCP set high_window_ 83000
  - Agent/TCP set high_p_ 0.0000001
- The parameter specifying the decrease function at high_p_:
  - Agent/TCP set high_decrease_ 0.1

# HighSpeed TCP: The Gory Details:

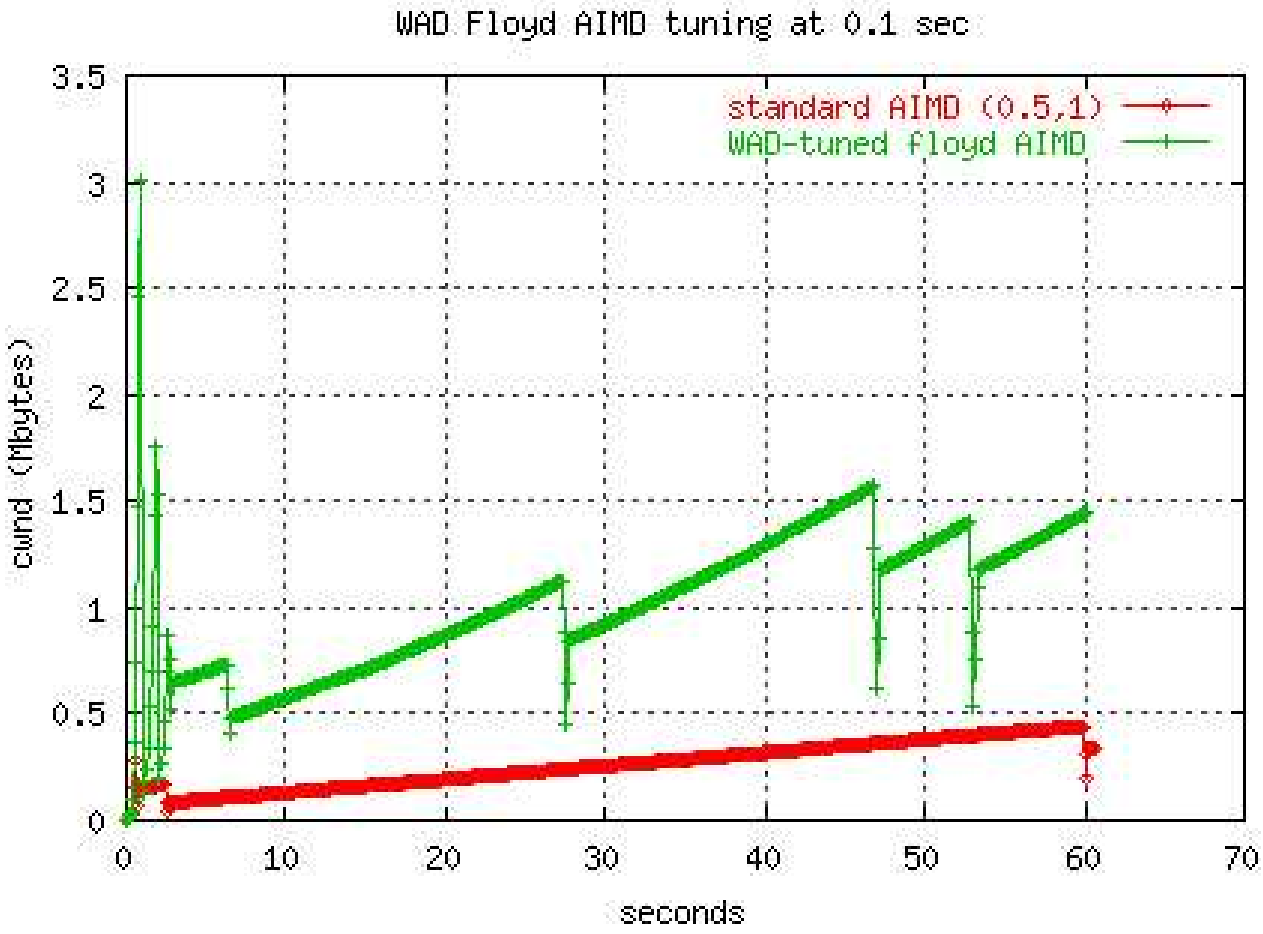
| w | a(w) | b(w) |
|---|---|---|
| 38 | 1 | 0.50 |
| 118 | 2 | 0.44 |
| 221 | 3 | 0.41 |
| 347 | 4 | 0.38 |
| 495 | 5 | 0.37 |
| 663 | 6 | 0.35 |
| 851 | 7 | 0.34 |
| 1058 | 8 | 0.33 |
| 1284 | 9 | 0.32 |
| 1529 | 10 | 0.31 |
| 1793 | 11 | 0.30 |
| 2076 | 12 | 0.29 |
| 2378 | 13 | 0.28 |
| ... | | |
| 84035 | 71 | 0.10 |

## Conclusions:

*

- This proposal needs feedback from more experiments.
- My own view is that something like this is the fundamentally correct path:
  – given backwards compatibility and incremental deployment.
- Experimental results from Tom Dunigan are on the HighSpeed TCP web page.
  – http://www.icir.org/floyd/hstcp.html
  – Experimental results from Brian Tierney coming shortly...

# Tests from Tom Dunigan:




This shows HighSpeed TCP with Limited Slow-Start (described next).

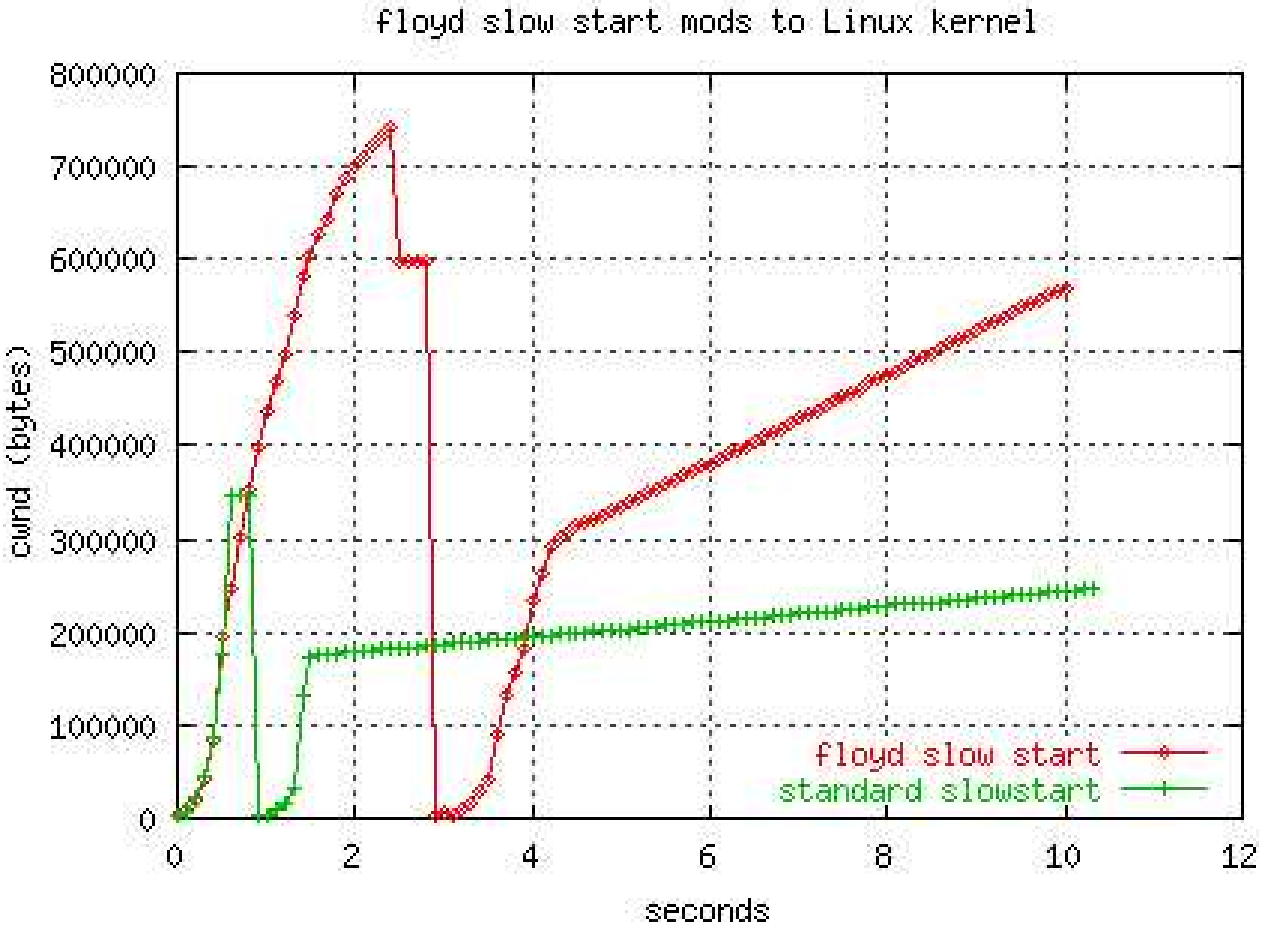
## HighSpeed TCP requires Limited Slow-Start:

*

- Slow-starting up to a window of 83,000 packets doesn't work well.
  – Tens of thousands of packets dropped from one window of data.
  – Slow recovery for the TCP connection.

- The answer: Limited Slow-Start
  – Agent/TCP set max_ssthresh_ N
  – During the initial slow-start, increase the congestion window by at most N packets in one RTT.

# Tests from Tom Dunigan:




This shows Limited Slow-Start, but not HighSpeed TCP.

## The pseudocode:

*

```
For each arriving ACK in slow-start:
  If (cwnd <= max_ssthresh)
     cwnd += MSS;
  else
     K = 2 * cwnd/max_ssthresh ;
     cwnd += MSS/K ;
```

## Other small changes for high congestion windows:

*

- Wait for more than three duplicate acknowledments before retransmitting a packet.
- Or, recover more smoothly when a retransmitted packet is dropped.

## Additional Problems:

*

- Starting up with high congestion windows?
- Making prompt use of newly-available bandwidth?

# What is QuickStart?

*

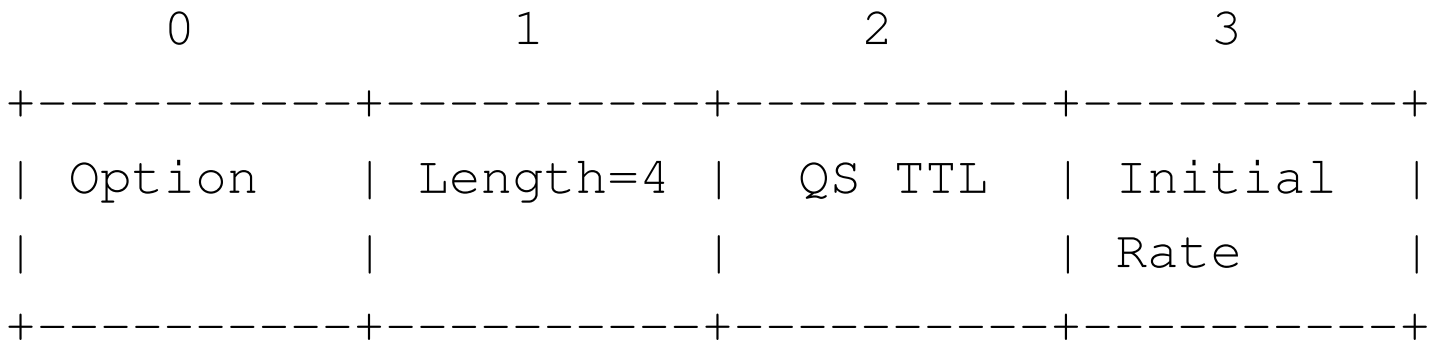
- In an IP option in the SYN packet, the sender's desired sending rate:
  – Routers on the path decrement a TTL counter,
  – and decrease the allowed initial sending rate, if necessary.

- The receiver sends feedback to the sender in the SYN/ACK packet:
  – The sender knows if all routers on the path participated.
  – The sender has an RTT measurement.
  – The sender can set the initial congestion window.
  – The TCP sender continues with AIMD using normal methods.

- From an initial proposal by Amit Jain

# The Quick-Start Request Option for IPv4

*




- Explicit feedback from all of the routers along the path would be required.
- This option will only be approved by routers that are significantly underutilized.
- No per-flow state is kept at the router.

## Questions:

*

- Would the benefits of Quick-Start be worth the added complexity?
  – SYN and SYN/ACK packets would not take the fast path in routers.
- Is there a compelling need to add some form of congestion-related feedback from routers such as this (in addition to ECN)?
- Is there a compelling need for more fine-grained or more frequent feedback than Quick-Start?
- Are there other mechanisms that would be preferable to Quick-Start?

## Architectural sub-themes favoring incremental deployment:

*

- A goal of incremental deployment in the current Internet.
- Steps must go in the fundamantally correct, long-term direction, not be short-term hacks.
- Robustness in heterogeneous environments valued over efficiency of performance in well-defined environments.
- A preference for simple mechanisms, but a skepticism towards simple traffic and topology models.
- Learning from actual deployment is an invaluable step.
- The Internet will continue to be decentralized and fast-changing.

## References:

*

- HighSpeed TCP and Limited Slow-Start:

http://www.icir.org/floyd/hstcp.html

- QuickStart:

http://www.icir.org/floyd/papers/draft-amit-quick-start-01.txt

## Extra viewgraphs:

*

## Problem Statement for DCCP

*

Datagram Congestion Control Protocol
draft-floyd-dccp-problem-00.txt
Sally Floyd, Mark Handley, and Eddie Kohler

## Requirements:

*

- Unreliable data delivery, but with congestion control.
- ECN-capable.
- A choice of TCP-friendly congestion control mechanisms.

## Constraints:

*

- Low overhead, for applications that send small packets.
- Traversing firewalls?
- Ability to negotiate congestion control parameters:
  – ECN.
  – type of congestion control.

## Three possibilities, for flows that now use UDP:

*

- Congestion control above UDP.
- Congestion control below UDP.
- Congestion control in another transport protocol.

## Congestion control above UDP:

*

- Burden on the application designer, or on RTP.
- The problems of firewall traversal and parameter negotiation remain.
- Application-level control over ECN?
- Evasion of end-to-end congestion control?

## Congestion control below UDP:

*

- If congestion control feedback is at the application layer:
  – CM does this.
  – Issues: parameter negotiation; ECN; firewalls; evasion of congestion control.

- If congestion control feedback is at the layer below UDP:
  – An additional packet header is needed.
  – To be most effective, the semantics of the UDP socket API would have to be changed, for late binding, and for communication of sequence numbers. Thus, we are already changing UDP.

## If a new transport protocol (other than UDP):

*

- Modify TCP?
  – We want a choice of congestion control mechanisms.
  – We want sequence numbers in packets rather than bytes.
  – Would we need a new protocol number anyway?

* Unreliable variants of SCTP?
  – Support for multiple streams is not needed for unreliable transfer, so we don't want to pay the price in extra packet overhead.
  – Separate control chunks for ECN feedback?
  – We want a choice of congestion control mechanisms.

* A new protocol?
  – Yep.

## Other design considerations:

*

- Mobility?

- Defense against DoS attacks:

server should not hold state for unacknowledged connection attempts.

- Interoperation with RTP.

- Interactions with NATs and firewalls:

- Explicit connection setup and teardown helps.

## Questions:

*

- Is this the right problem?
- Do we have the right set of constraints?
- Are there other requirements that we haven't considered?
- Feedback?

# What is XCP?

*

- Congestion Control for High Bandwidth-Delay Product Networks
  – by Dina Katabi, Mark Handley, and Charlie Rohrs.

- XCP (eXplicit Control Protocol) has the goals of stability,

fair bandwidth allocation, high utilization, small standing queue size, and near-zero packet drops.

- Specific goals:
  – Minimizing oscillations.
  – High delay-bandwidth-product connections.
  – Minimizing the transfer delay of short flows.
  – Fairness between flows with different RTTs.

- No per-flow-state is maintained in routers.

## XCP: the End Nodes

$*$

- The packet header contains:
  - current cwnd,
  - rtt estimate,
  - feedback
    (Initialized to the desired increase in bytes in the cwnd, per ACK.)

- Routers modify the feedback field.

- At the sender, for each ACK:

cwnd $<-$ max (cwnd + feedback, packet size)

# XCP: the Routers

*

- Routers deal with efficiency and fairness separately.
- The efficiency controller computes the desired change in the number of arriving bytes in a control interval (i.e., an average RTT), based on the spare bandwidth and persistent queue.
- The fairness controller uses AIMD to allocate the increase or decrease to individual packets.
- This requires a few additions and three multiplications per packet.
- Policing agents can be used at the edge of the network for security.