Bulk Data Transfer

Colin Perkins

http://csperkins.org/teaching/2004-2005/gc5/

UNIVERSITY *of* GLASGOW

Motivation

- Scientific experiments and data analysis
  - Particle physics experiments like the Large Hadron Collider
    - Petabytes per year for approximately the next 15 years
  - Large scale distributed data repositories ("virtual observatories") for the astronomy and earth sciences communities
  - Human genome and similar biotechnology projects
- Commercial interests
  - Film and television production and special effects industries
    - Uncompressed HDTV is 120 megabytes/second; cinema content ~16× more
  - Large scale database replication and backup

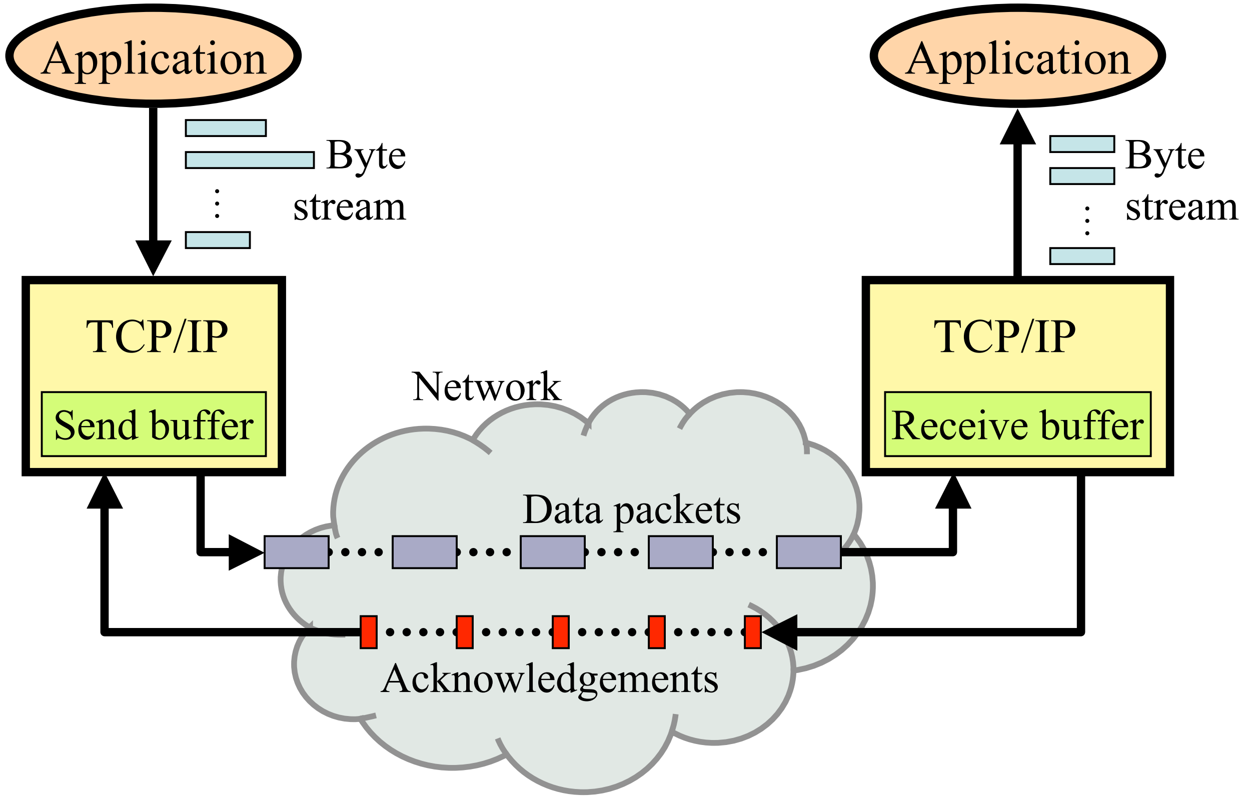
⇒ Requirement for predictable and high performance transfer of large data sets across TCP/IP networks

Lecture Outline

- Review of TCP/IP
- Bulk Data Transfer Using TCP/IP
  - Performance Limitations
- How to Improve Transfer Performance
  - Parallel Streams
  - Modifications to TCP/IP
  - New Transport Protocols
  - Enhanced Quality of Service
- Deployment and Standardisation of Alternatives

Review of TCP/IP

- TCP/IP ensures reliable, in-order, delivery of a byte stream over an unreliable packet network
  - Each packet is acknowledged, reliability through retransmission
  - Sliding window congestion control adapts sending rate to network capacity



The TCP protocol controls the transfer rate, and when packets are released to the application, based on the network conditions

TCP Reliability

- Packets contain a checksum to detect corruption
- Packets contain a sequence number to detect loss
- Receiver sends an acknowledgement containing the highest contiguous sequence number received each time a packet is received
- Sender uses duplicate sequence numbers to infer lost/reordered packets
  - Retransmits lost packets
  - Stalls receiver application until retransmission arrives; data delivered in order

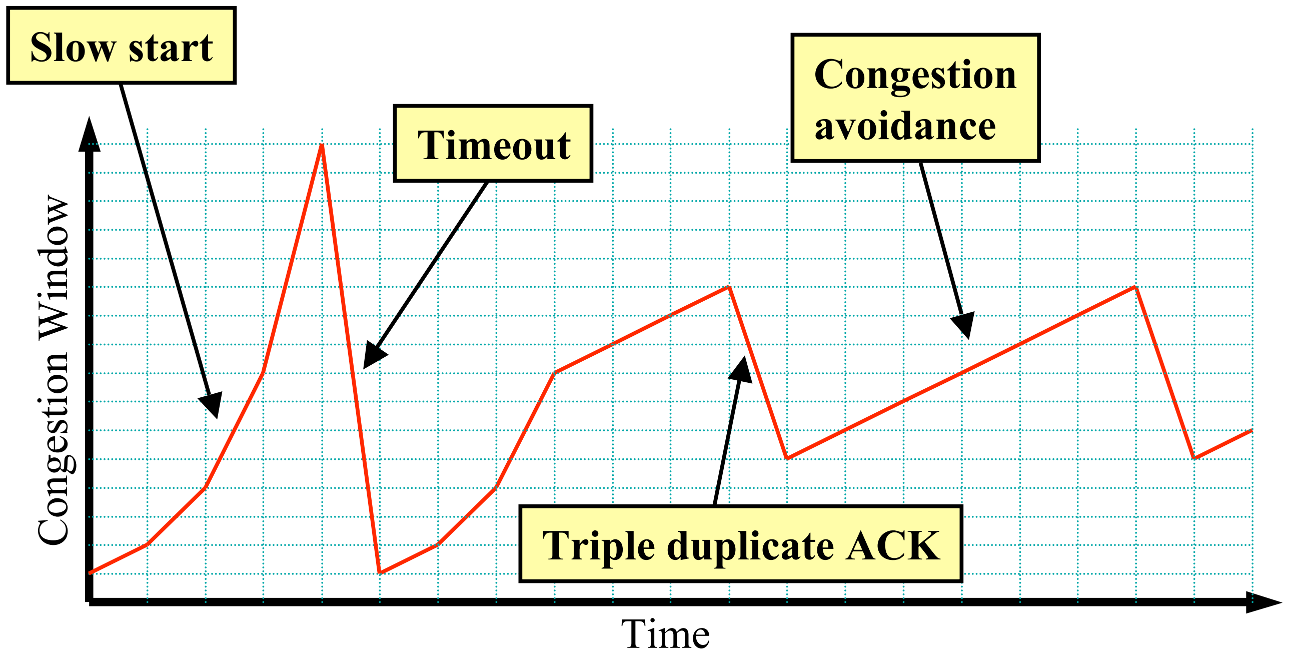
TCP Congestion Control

- Receipt of acknowledgements also drives TCP congestion control

- TCP is a sliding window protocol
  - A *congestion window* indicating the number of packets allowed in flight
    - Acknowledgements of new data increase the congestion window
      - *Slow start*
      - *Congestion avoidance*
    - Packet loss reduces the congestion window
      - *Triple duplicate ACK*
      - *Timeout*
  - A *receive window* to indicate how much data the receiver can handle
    - Flow control
  - Sender have max(*congestion window*, *receive window*) packets outstanding

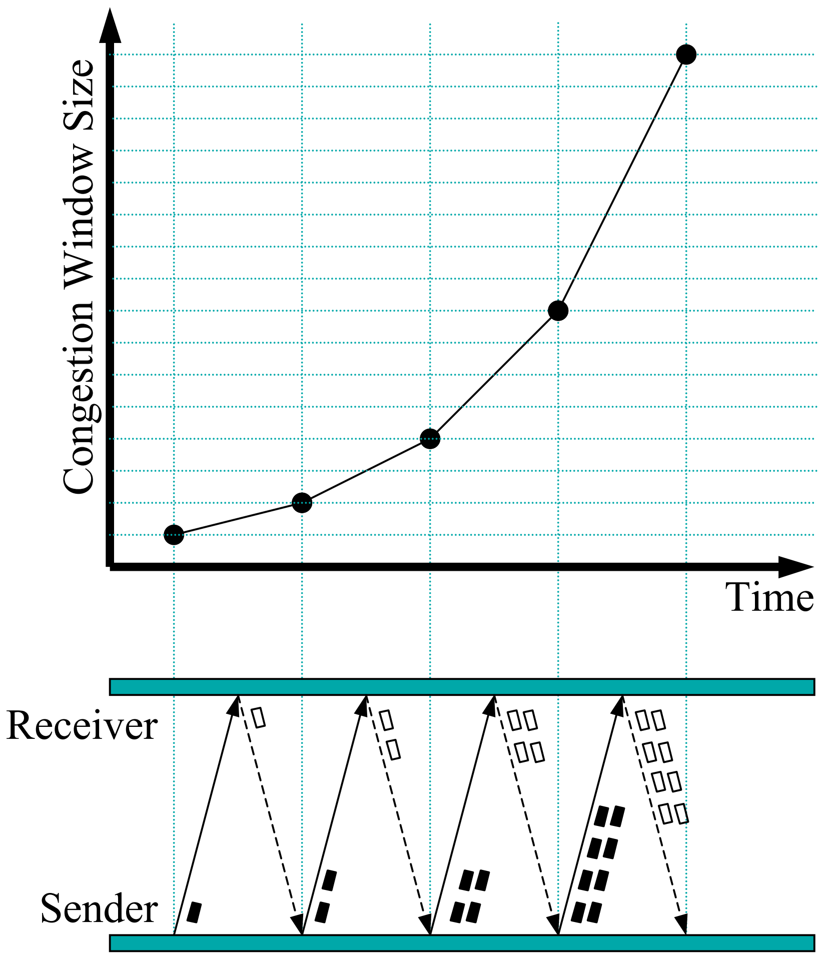
Evolution of the Congestion Window

- Additive increase/multiplicative decrease in the window
  - Linear probe of available capacity until momentary overload
  - Multiplicative back-off to safe sending rate
- Ensures capacity is used, avoids network overload
- Approximately equal share of bottleneck capacity between flows



Slow Start

- The *slow start* algorithm is used to rapidly probe network capacity
- Connections start with initial window of min(4M, max(2*M, 4380)) octets
  - M is maximum segment size
  - 3 packets on Ethernet where M=1460
  - Old implementations start at 1 packet
- Congestion window increases by one packet when an ACK is received that acknowledges new data
  - $cwnd_{new} = cwnd_{old} + 1$
  - Slow start, exponential growth
- Stops when a loss event occurs

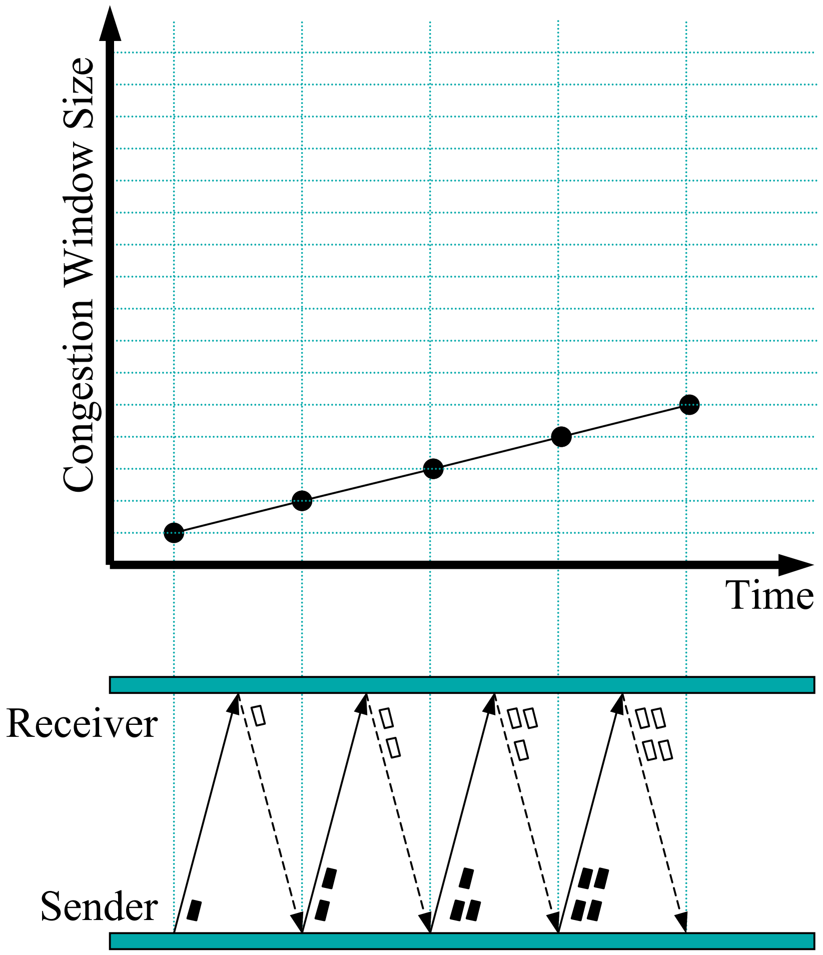


Congestion Avoidance

- Once in a stable regime, *congestion avoidance* takes over
- Additive increase in congestion window
- For each non-duplicate ACK received:

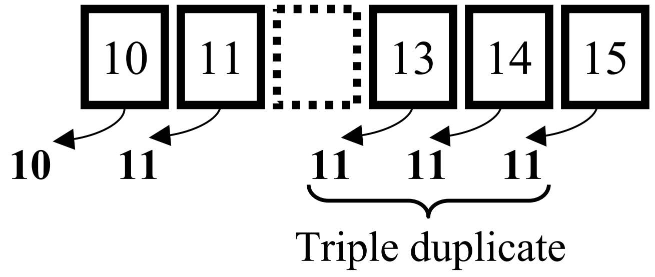
$$cwnd_{new} = cwnd_{old} + \frac{1}{cwnd_{old}}$$

- Equivalent to a linear increase in the congestion window by one segment per round-trip time

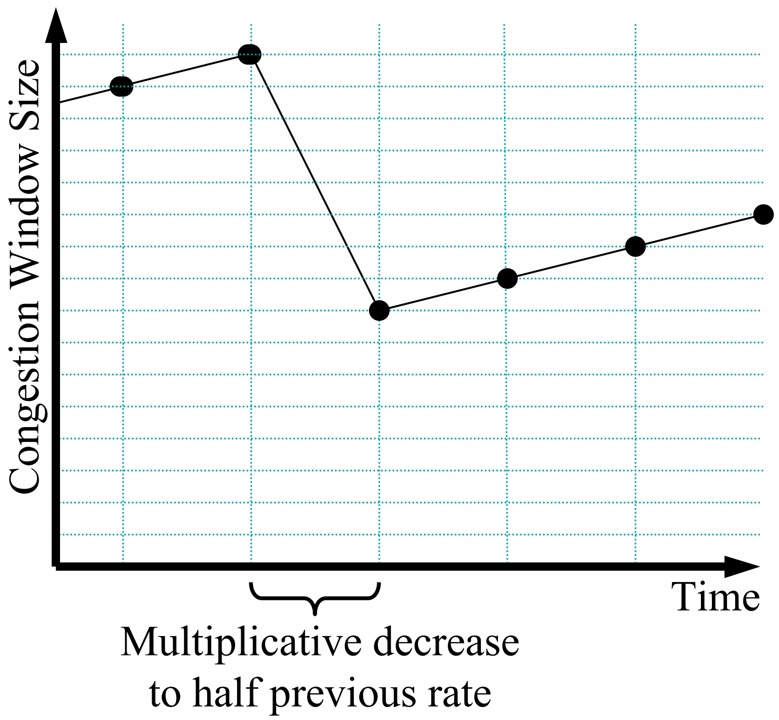


Reaction to Loss: Triple Duplicate ACK

- A single packet loss causes a triple duplicate ACK to be generated



  Sender waits for triple duplicate, to be robust to single packet reordering
- On receipt of triple duplicate, reduce to half previous sending rate, continue in congestion avoidance

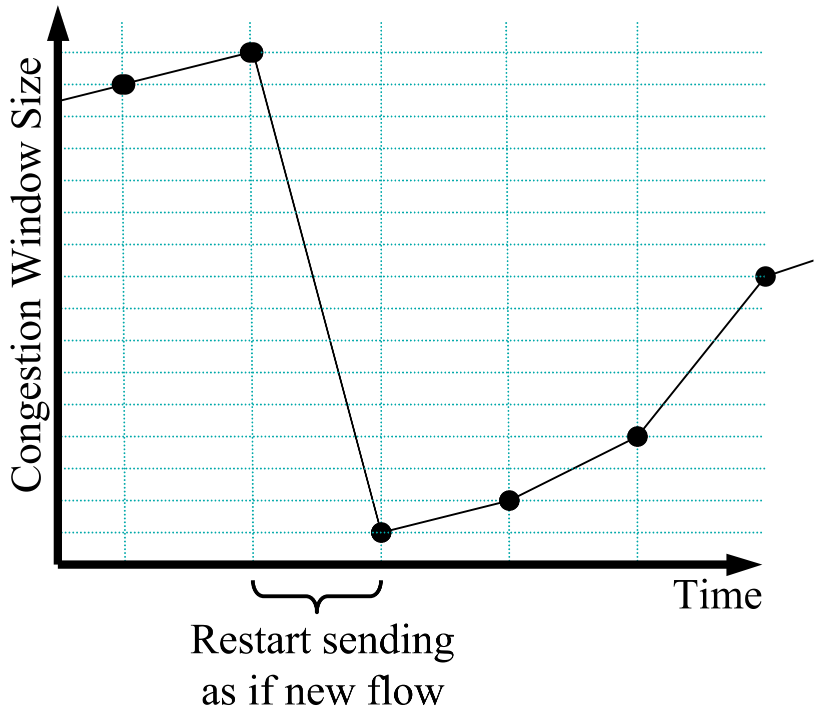


[Slightly simplified to omit “fast recovery” - See RFC2581]

Reaction to Loss: Timeout

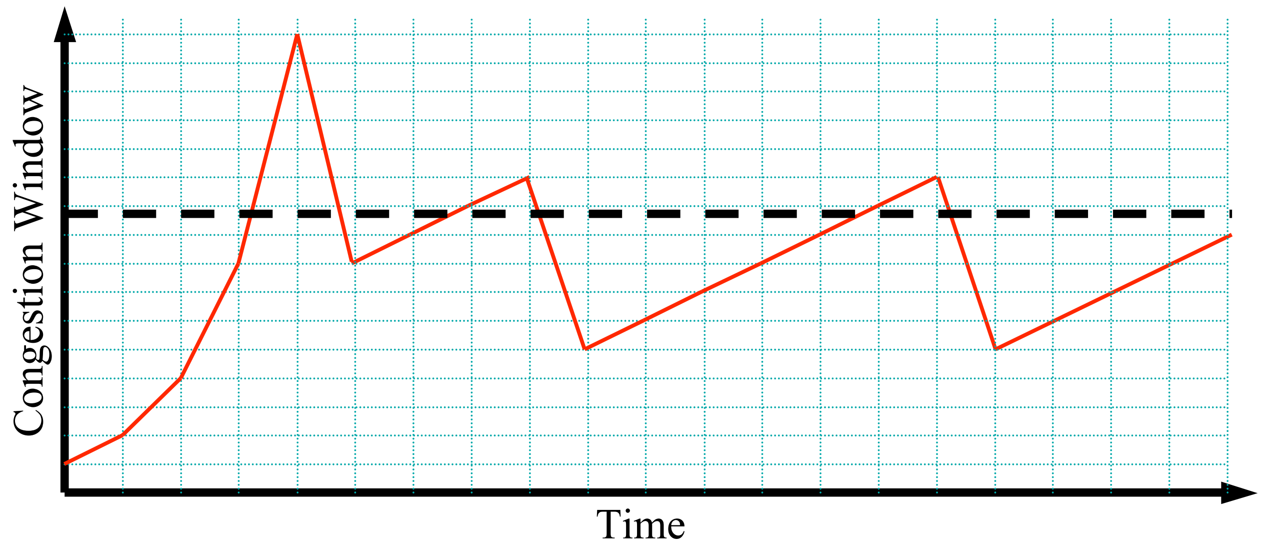
- Timeouts occurs when ACKs stop being received
- Two reasons:
  - Failure on the forward path; data doesn't reach receiver, so it stops generating ACKs
  - Failure on the reverse path; receiver is generating ACKs but the don't reach the sender

- On timeout, sender reduces congestion window to 1 segment, enters slow start until half previous rate then congestion avoidance



Evolution of the Congestion Window

- Ideally a flow slow starts to probe the capacity, then follows a saw tooth pattern of congestion avoidance and triple duplicate ACKs with back-off
- Steady state is to oscillate around the bottleneck link rate
  - Reductions in rate allow router queues to empty; receiver sees constant rate
  - Window ≈ bandwidth * delay of path



Other Features

- Selective acknowledgements (SACK) allow a receiver to indicate which packets arrived following a loss
  - Lets the sender only retransmit data that was actually lost
  - Faster recovery from loss
- Window scaling
  - TCP packet header has a 16 bit field to advertise the *receive window*, but 65536 bytes too small for modern networks
  - A *window scale* option conveys an integer multiple scale factor, to allow larger windows

TCP/IP Performance

- The slow start algorithm is designed to rapidly find the bottleneck link capacity
- The congestion avoidance algorithm will continually probe for changes in capacity during a connection
- Supposed to allow TCP to make effective use of network capacity

- But, many people complain that TCP/IP is slow…
- Why is this is case?
  - Poorly tuned hosts
  - AIMD behaviour, aggressive back-off, slow linear increase; poor performance on large fat pipes

TCP Performance Tuning

- Most operating systems choose TCP parameter defaults that are not optimized for high-performance, tuning often necessary:
  - Use large initial congestion window, window scaling, SACK, etc.
    - Gigabit wide area needs a window of ~10Mbytes
  - Tune system interrupt handling to reduce per-packet overhead
    - Interrupt coalescing, delayed interrupts
    - Polled rather than interrupt driven network devices (FreeBSD)
  - Use largest MTU possible, to reduce per-packet overhead

- Most operating systems have poor defaults
  - Optimized for many connections, not high performance
  - Need to tune system parameters!

Modelling TCP Throughput

- But, even when correctly tuned, performance can be poor…
- Need to use a mathematical model to understand the factors that limit TCP protocol performance
  - Model the fundamental behaviour of the protocol, rather than specific implementation issues
  - Much research conducted in this area, driven by the needs of the Grid computing community

Modelling TCP Throughput

- Current best model due to Padhye *et al.* [1]:

$$T = \frac{s}{R\sqrt{\frac{2p}{3}} + 3p(1+32p^2)\cdot T_{rto}\sqrt{\frac{3p}{8}}}$$

All parameters are measurable

T = average throughput | s = packet size
R = round trip time | p = loss event rate
T_{rto} = retransmission timeout (often approximated as $T_{rto} = 4R$)

- Makes certain assumptions so the analysis is tractable:
  - Saturated steady state TCP Reno sender
  - Packet loss correlated within sending window, uncorrelated long term
  - Packet reordering rare
- Models average behaviour on an idealised network; reasonable but not perfect fit with average behaviour of real systems

Modelling TCP Throughput

$$T = \frac{s}{\underbrace{R\sqrt{\frac{2p}{3}}}_{\text{Models triple duplicate ack}} + \underbrace{3p(1+32p^2)\cdot T_{rto}\sqrt{\frac{3p}{8}}}_{\text{Models timeout}}}$$

- When p is small, $3p(1+32p^2)$ tends to zero and response to triple-duplicate ACKs dominate; as p increases timeouts predominate

- Assuming low loss rates, can approximate: $T \approx \frac{s}{R\sqrt{p}}$

TCP and Large Fat Pipes

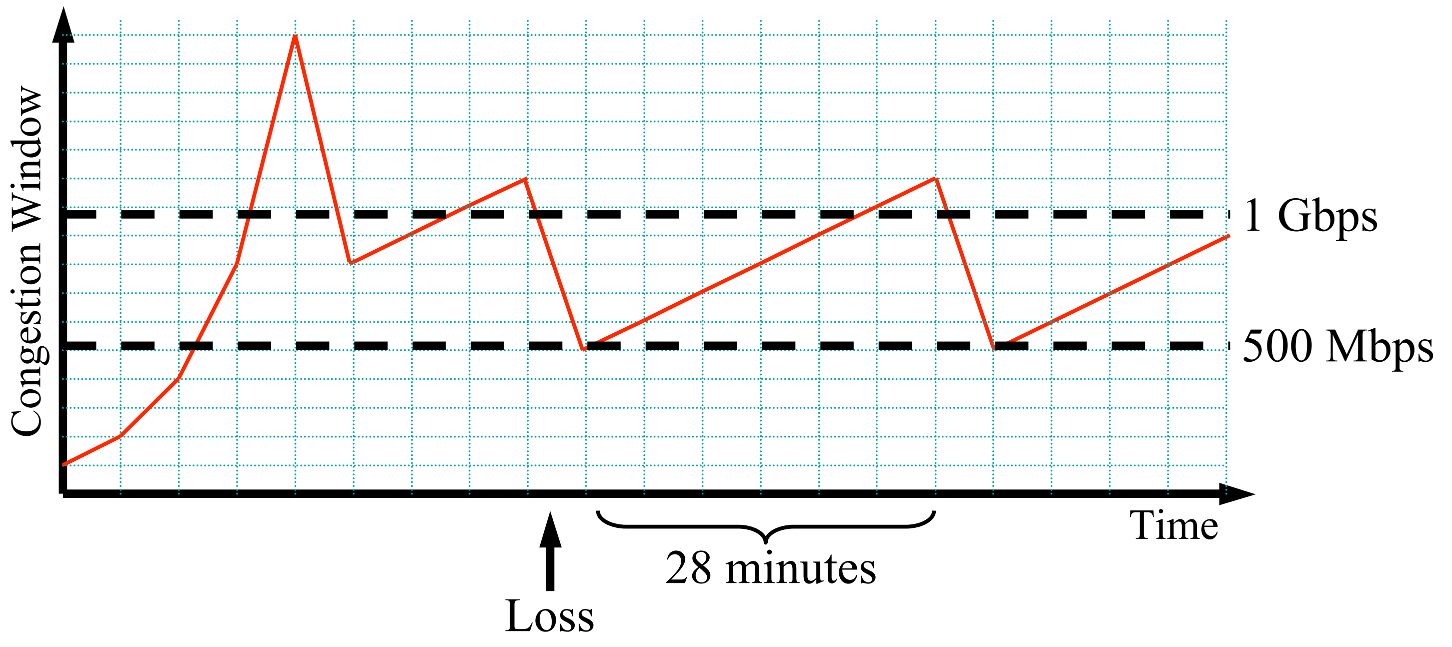
- If we invert the previous equation, can derive the loss event rate needed to sustain a particular throughput:

$$p \approx \left(\frac{s}{TR}\right)^2$$

- What if we want to send 10Gbps transatlantic?
  - s = 1500 octets, T = 10Gbps, R = 100ms $\Rightarrow p = 2*10^{-10}$
  - This corresponds to an error rate of about 1-in-10^{14} bits, which is more than the inherent bit error rate of optical fibre

TCP and Large Fat Pipes

- Worse: the AIMD behaviour of TCP means it recovers slowly from loss

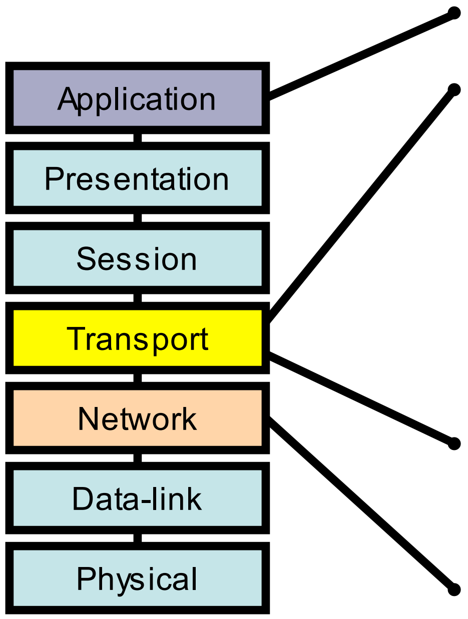


- Not an issue at low speeds, but big problem at high rates…

TCP and Large Fat Pipes

- Conclusion: TCP requires unrealistically low loss event rates to sustain high performance
- This is a fundamental limitation of TCP, not something that can be solved through host performance tuning

How to Improve Transfer Performance



- Parallel streams
- Modify the TCP congestion response
  - High Speed TCP
  - Scalable TCP
  - H-TCP
  - FAST TCP
- Replace TCP
  - XCP
- Modify the IP layer to avoid congestion
  - Integrated and Differentiated Services
  - MPLS and Optical Switching

- Can approach the problem at different layers in the protocol stack
- Trade-off ease of implementation and deployment versus potential for performance improvement

Parallel Connections

- Simple application level solution: if a single stream is too slow, open multiple connections
  - Web browsers using HTTP
  - GridFTP in Globus toolkit
- Simple to deploy; no changes to the operating system or network

- Small numbers of streams improve performance

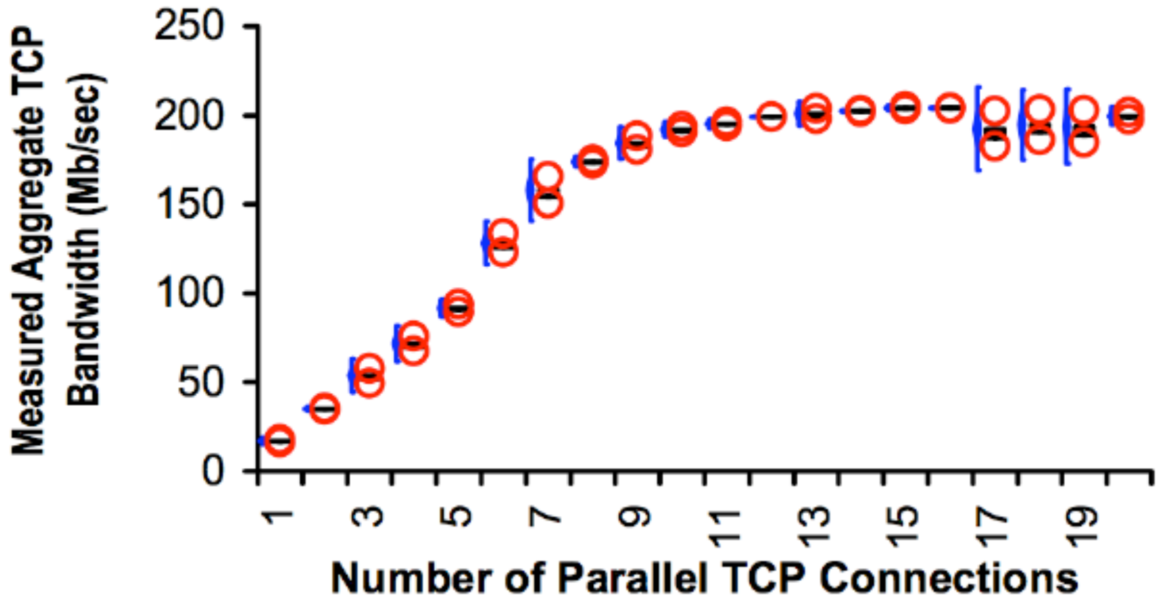
  …but larger numbers interfere with each other

Parallel Connections

- Can be shown that throughput of n parallel flows is:

$$T \leq \frac{s}{R}\left[\frac{1}{\sqrt{p_1}} + \frac{1}{\sqrt{p_2}} + \cdots + \frac{1}{\sqrt{p_n}}\right]$$

- Implies n flows achieve n times the capacity of a single flow
- But more flows will increase p_x, so full benefit not gained…



From [2]

Modify the TCP Congestion Response

- For standard TCP:
  - On ACK: $cwnd_{new} = cwnd_{old} + \alpha / cwnd_{old}$
  - On drop: $cwnd_{new} = cwnd_{old} - \beta * cwnd_{old}$
  - Where $\alpha = 1$ and $\beta = 0.5$

- Can make TCP more aggressive by increasing α and β
  - Naïvely doing so makes it unfair to standard TCP
  - Adjust these parameters as a function of the window size
    - Gradually make response more aggressive as the window increases
    - Same response at the rates standard TCP achieves, more aggressive at higher rates (to allow it to achieve higher rates)
    - Difficult to achieve stability, fairness
  - Many proposals: HighSpeed TCP [3], H-TCP, Scalable TCP, etc.
  - Active research area… no standard solution

Replace TCP

- If TCP works so poorly, can we replace it?
  - Yes, but not easily
  - Need to update all hosts, NAT boxes and firewalls

- Three protocols under serious development:

– SCTP	Telephony signaling; TCP-like congestion control with fail-over
– DCCP	Streaming media; TCP-friendly congestion control
– XCP	Alternative to TCP that needs router support; non-TCP congestion control, higher performance (Will be discussed in the tutorial this week - see the paper [4])

Modify the IP Layer to Avoid Congestion

- Alternative to making TCP faster, we can isolate our traffic to avoid congestion on the network
  - Quality-of-service (QoS)
    - Differentiated services
    - Integrated services/RSVP
  - Traffic engineering (MPLS)

- Or we can add *explicit congestion notification* so routers can inform connections of loss without dropping packets
  - A bit in the IP header to signal "congestion occurring, slow down"

- Hard to deploy:
  - Need to update routers; possible since ISPs apply software updates
  - Need to update NAT and firewalls (including home NAT boxes…)

Deployment and Standardisation Issues

- IETF has active work to standardize improvements to TCP:
  - http://www.ietf.org/html.charters/tcpm-charter.html
  - http://www.ietf.org/html.charters/tsvwg-charter.html
- Many experiments using Linux
  - Widespread deployment difficult unless you can persuade Microsoft…

- Much work on QoS - mature and developing standards
- Very limited deployment
  - Economic issues: no scalable solutions to billing, accounting, etc.

Lecture Summary

- You should know…
  - How TCP congestion control works
  - The limitations of TCP/IP for high performance networking
  - Outlines approaches to improving performance
    - Parallel connections
    - Modifying TCP
    - New transport protocols
    - Modifying IP

- The tutorial on Friday:
  - Discussion of HighSpeed TCP and XCP (see papers…)

References

1. J. Padhye, V. Firoiu, D. Towsley and J. Kurose, "Modelling TCP Throughout: A Simple Model and its Empirical Validation", Proceedings of SIGCOMM 1998, Vancouver, Canada, September 1998.
2. T. Hacker, B. Athey and B. Noble, "The end-to-end performance effects of parallel TCP sockets on a lossy wide-area network", Proceedings of the 16th IEEE/ACM International Parallel and Distributed Processing Symposium, Ft. Lauderdale, FL, USA, April 2002.
3. S. Floyd, "HighSpeed TCP for Large Congestion Windows", Internet Engineering Task Force, RFC 3649, Experimental, December 2003
4. Dina Katabi, Mark Handley, and Charles Rohrs, "Internet Congestion Control for High Bandwidth-Delay Product Networks", Proceedings of SIGCOMM 2002, Pittsburgh, August, 2002.

Plus any standard text on TCP/IP networking…