COMP312-09A Communications and Systems Software

Lecture 7 – TCP III

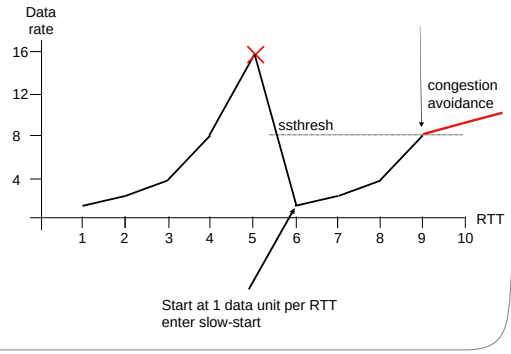
Matthew Luckie
mluckie@cs.waikato.ac.nz

THE UNIVERSITY OF WAIKATO

Overview

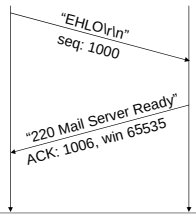
- Last lectures
  - TCP receive window
  - Window scaling
  - TCP options
  - Delayed acknowledgements
- This lecture
  - Fast retransmit
  - Fast recovery

TCP congestion avoidance



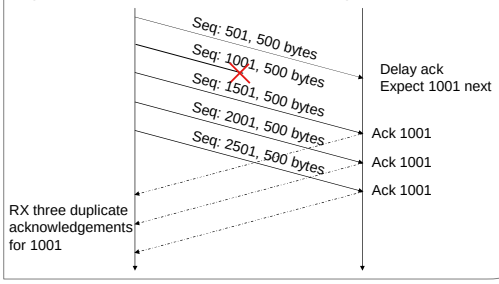
Delayed acknowledgements

- TCP implementations delay sending an acknowledgement immediately
  - Main idea is to avoid redundant acknowledgements
  - Receiver application is likely to have something to send shortly
- TCP acknowledges data received in order



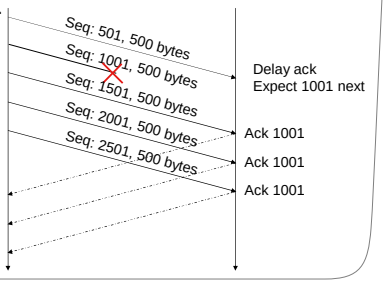
Exception to delaying an acknowledgement

- If TCP receives a data segment out of order (i.e. there is a hole in the receive window) it is required to generate an **immediate** acknowledgement



Duplicate acknowledgements

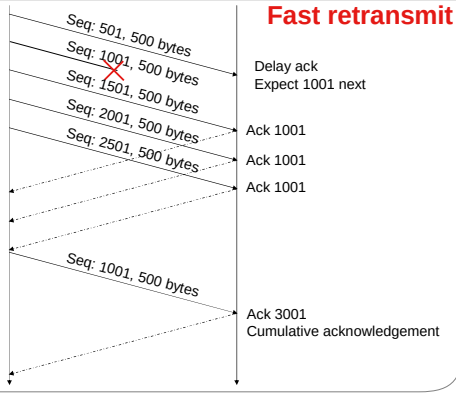
- Notice that the receiver has a fairly good idea that a segment has been lost by the receipt of the third duplicate acknowledgement, asking for data beginning with byte 1001.



Retransmission schemes

- Recall that TCP retransmits when the Retransmit Timer (RTO) expires
  - i.e. the absence of an acknowledgement is used to infer packet loss
- Fast retransmit: on the third duplicate ack, retransmit the assumed missing segment
  - i.e. infer packet loss by an apparent hole in the receiver's window
  - the fact that we are receiving duplicate acknowledgements means data is still being received, so the congestion event was not severe
    - don't want to go into slow-start again, just want to avoid congestion

Fast retransmit



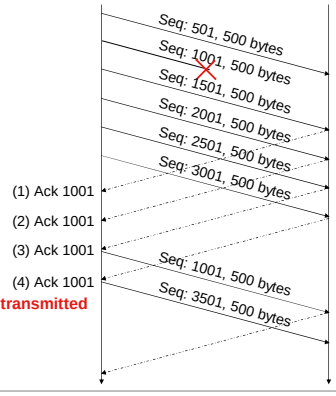
Fast retransmit

- Why wait for three duplicate acknowledgements?
  - Have more confidence that packet was lost, rather than just reordered
- Some work is going into adaptive algorithms to adjust the number of duplicate acknowledgements required before the sender retransmits
  - if the receiver has always observed duplicate acknowledgement run lengths of three (and never of length two) then chances are the path does not reorder packets
    - Reduce number of duplicate acks required
  - if the receiver has fast-retransmitted a packet, only to receive an acknowledgement for it before one RTT after the retransmission, then the segment was retransmitted too early
    - Increase number of duplicate acks required

Fast Retransmit + Fast Recovery

- Step One:
  - When third dup-ack is received, set ssthresh to half of the current value of cwnd or half of the receiver's window, which ever is smaller
  - Retransmit the segment assumed to be missing
  - Set cwnd to ssthresh plus 3 times the segment size
- Step Two:
  - Each time another dup-ack is received, increment cwnd by the segment size and transmit a new segment, if allowed by the value of cwnd

Step two

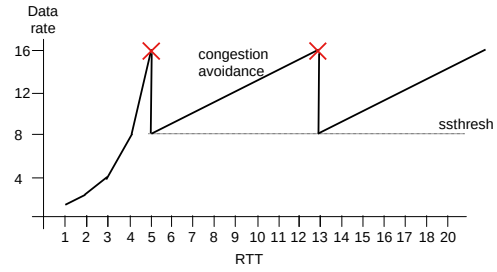


new segment is transmitted

Fast Retransmit + Fast Recovery

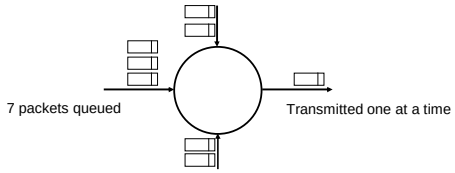
- Step three
  - When an acknowledgement that is asking for new data arrives, set cwnd to ssthresh (the value stored when we entered fast-retransmit).
  - Enter congestion avoidance

Fast Retransmit + Fast Recovery



Loss as indication of congestion

- TCP assumes all packet loss is due to congestion
  - Congestion occurs when the outgoing interface cannot transmit at the same rate as packets are arriving and the queue holding packets for transmission overflows
- However, loss is not always due to congestion



Loss as indication of congestion

- Main example of non-congestion loss in today's Internet is loss over wireless Ethernet
  - Wireless as a medium is much less reliable, as interference is difficult to insulate against
  - Wireless tends to be found at the edges of the network
  - Other wireless devices using same frequency band (2.4Ghz)
    - baby monitors
    - cordless phones
    - microwave ovens
  - Packet loss on wireless is normally not due to congestion, and is a local artefact
- However, no good way to overcome this problem and still have a TCP that is fair in other (non-wireless) scenarios

TCP implementation quirks

- Cumulative acknowledgements
- Nagle
- Zero advertised window

TCP implementation quirks

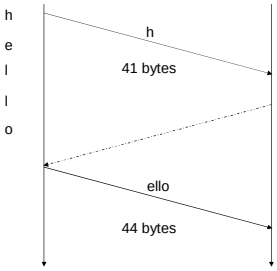
- Recall that TCP acknowledges data received in order
- TCP acknowledgements are cumulative
  - i.e. they acknowledge all data received up to the specified point
- No requirement for TCP receiver to keep data received out of order, as sender is required to keep it until it is acknowledged
  - Makes implementation easy
  - Small, embedded devices may discard data received out of order
  - In practice, most other receivers do buffer data received out of order

Nagle: TCP for interactive data flows

- Consider remote login systems, such as telnet and ssh
- Clients write() typed characters into socket as they are typed
- Inefficient to transmit each character independent of other characters.
  - A one byte character with 20 bytes of TCP and 20 bytes of IP would require 41 bytes to transmit
  - Routers typically are bound by routing decisions, not serialisation rate

Nagle: TCP for interactive data flows

- Nagle's algorithm
- Send first byte immediately. Subsequent writes are buffered until first byte is acknowledged



Nagle: TCP for Interactive flows

- Not all interactive data flows have the same requirements
- Algorithm is disabled for "highly interactive" data flows (e.g. X-windows) where the mouse (for example) is moved frequently

```
Socket s;
s.setTcpNoDelay(true);
```


Zero-advertised window

- Sometimes, we might fill up the receiver's window, to the point we receive a zero-advertised window
- At this point, we rely on the receiver to issue an acknowledgement when space in the window comes free
- What if that acknowledgement is lost?

Zero advertised window

- Sender allowed to send a one byte segment into network to solicit a new acknowledgement
  - and thus be told if space in the receive window has come free

Summary

- Fast Retransmit and Fast Recovery use duplicate acknowledgements to indicate a packet was lost, and begin recovery faster than we otherwise would
- Brief coverage of TCP implementation quirks
  - Cumulative acknowledgements
  - Nagle
  - Zero-advertised window
- Next Lecture:
  - Queues, RED, ECN
  - LFNs