COMP312-09A
Communications and Systems Software

Lecture 5 – TCP I

Matthew Luckie

mluckie@cs.waikato.ac.nz

Overview

- Brief re-cap of COMP202-08B lectures 13 to 16 inclusive
  - Pages 532 to 554
- New material today
  - sliding windows vs. stop-and-wait (pages 211 to 223)
  - bandwidth delay product (page 559)

TCP

- TCP is a reliable, connection oriented, network-friendly, stream-based transport protocol
- Most Internet applications use TCP because a stream-based, reliable transport protocol is useful in many scenarios, e.g.
  - bulk transfer (reliability important, delay tolerant)
    - file transfer
    - email
  - interactive data flows (reliability important, delay intolerant)
    - remote login
- Some Internet apps don't use TCP by choice
  - Reliability in the form of retransmissions is inappropriate
    - Voice
    - Games

TCP header: learned in 202

0 15 16 31

Source Port	Destination Port
Sequence Number	
Acknowledgement Number	
ACK, RST, SYN, FIN	Window Size
Checksum	

Note: portions have intentionally been left clear

TCP: connection-oriented

- must establish a connection before data can be transmitted
  - 3-packet handshake (syn, syn/ack, ack)
- connection defined by
  - source IP address
  - destination IP address
  - source port
  - destination port
- two connections cannot share the same combined set of address/port values because it would be hard to distinguish them

TCP: reliability

- Retransmit timer (RTO)
  - if acknowledgement is not received in time, packet is retransmitted
  - RTO(k) = SRTT(k) + f * $SRTT_{var}$(k)
- Acknowledge data successfully received
- Checksum over TCP header, and data
- Transparently handles segments received out of order
- Transparently handles segments being duplicated
- Flow control to avoid overwhelming end-host and network

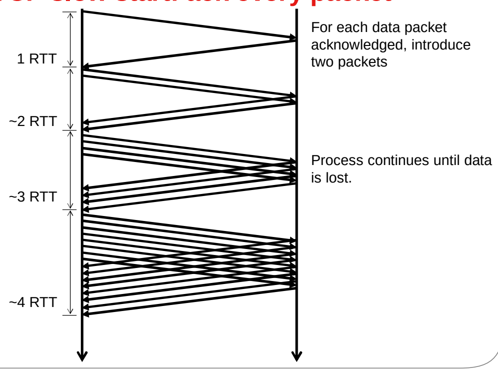
TCP: stream based

- TCP applications do not deal in packets; they deal in bytes
  - No ability for TCP application to see how received data was transmitted, i.e. 80 bytes:
    - could have been transmitted in one 80 byte segment
    - could have been transmitted in two 40 byte segments
  - Sender writes bytes into one end of the connection
  - Receiver reads identical stream of bytes at the other end

TCP: network friendly

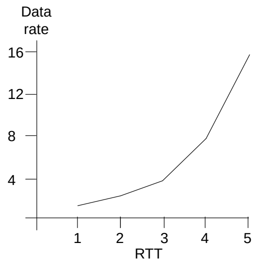
- Slow-start: (rapidly) find the available capacity in the network path
- Congestion avoidance: slowly increase rate of transmission towards threshold established in slow-start

TCP slow start: ack every packet

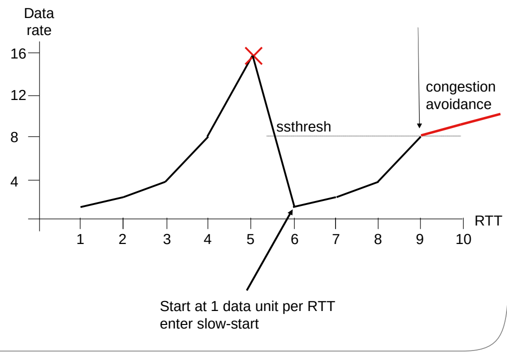


TCP slow start

- Despite its name, TCP slow-start results in an exponential increase in transmission rate
  - 1 RTT: 1 data unit
  - 2 RTT: 2 data units
  - 3 RTT: 4 data units
  - 4 RTT: 8 data units
  - 5 RTT: 16 data units
  - and so on



TCP congestion avoidance



TCP: what we're going to learn

- TCP header options
  - Window scaling
  - Selective acknowledgements
  - Segment size
- TCP for interactive data flows
  - Nagle
  - Delayed acknowledgement
- TCP for bulk transfer
  - Bandwidth delay product
  - Fast Retransmit
  - Fast Recovery
  - Explicit Congestion Notification
- Formalise what we know about TCP's behaviour

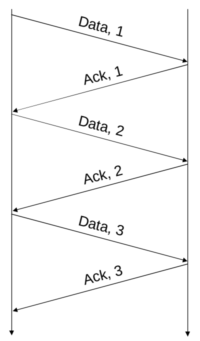
Data transmission methods

- Stop and wait
  - What TFTP uses
- Sliding window
  - What TCP uses

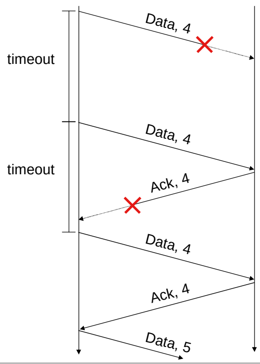
Stop and wait (TFTP)

- Whenever a node transmits a frame
  - it waits to receive an acknowledgement from the destination for a predetermined time.
- Whenever a node receives a frame that passes error checking
  - it responds to the source of that frame with an acknowledgement
- If the predetermined time expires without the source receiving an acknowledgement
  - it retransmits the frame
- Whenever the source receives an acknowledgement
  - it is free to transmit another frame if it has more data to send.

Stop and wait



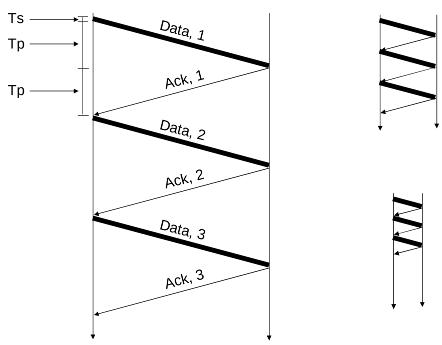
Stop and wait



Stop and wait performance

- The time to transfer a frame is composed of
  - the time to put the packet on the wire (serialisation delay, Ts)
  - the time for it to travel (propagation delay, Tp)
  - and processing time at nodes.
- Usually we neglect the processing time and the serialisation time for the acknowledgement.
  - computers are fast, packets should be processed in a fraction of a millisecond
  - acknowledgements are small frames, serialisation time should be small
- For stop and wait, the time to transmit each frame is one serialisation delay and two propagation delays (Ts + 2Tp).
- The useful work is being done during the serialisation delay.
  - If the propagation delay is large, the efficiency is poor.
  - i.e. good on LANs, poor on the Internet

Stop and wait performance



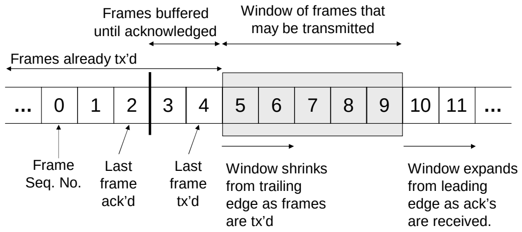
3

Sliding window

- A window size is defined: W frames.
- The source is allowed to transmit up to W frames without acknowledgement.
- Frames contain sequence numbers so that they can be identified (distinguished from each other)
- When the destination receives a frame that passes error check it responds with an acknowledgement.
- When the source receives an acknowledgement for frame number n it may transmit up to frame n + W

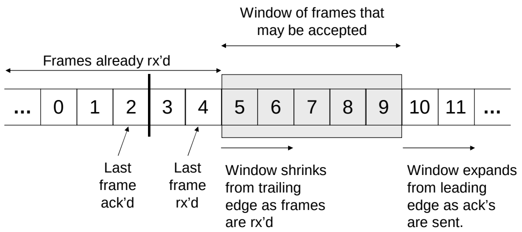
Sliding Window

Sender's perspective



Sliding Window

Receiver's perspective



Sliding window performance

- The transmitter can send W frames before it has to stop to wait for acknowledgement.
- The time to do this is W.Ts
- So long as this is greater than the round trip propagation delay W.Ts > 2 then acknowledgements will have already arrived and the window advanced.
- In this case sliding window throughput can approach 100%.
- This can be restated as the window size must be greater than the **bandwidth delay product**.

Bandwidth delay product

- In order to not limit the connection speed, the receive window needs to be at least this size

$$\text{WindowSize} = \text{BW} * \text{RTT}$$

$$\text{Throughput} = \frac{\text{WindowSize}}{\text{RTT}}$$

BDP example 1

- An Ethernet connected device has a receive buffer of only 500 bytes. What data rate can it receive from a host that is 10ms away round-trip, in bytes per second?
- How much buffer memory would be required to receive at 100 kB/s from a host 50ms away round-trip?

BDP example 1

- An Ethernet connected device has a receive buffer of only 500 bytes. What data rate can it receive from a host that is 10ms away round-trip, in bytes per second?
  - 500 bytes / 0.01 = 50,000 bytes/sec
- How much buffer memory would be required to receive at 100 kB/s from a host 50ms away round-trip?
  - 100 kB/s = 102400 bytes/sec
  - Window = 102400 * 0.05 = 5120 bytes

BDP example 2

- You have a Windows 2000 computer at university with a default receive window of 17520 bytes
  - What is the maximum throughput available to a host 4ms away (say auckland to hamilton)
  - What is the maximum throughput available to a host 150ms away (say google to hamilton)

BDP example 2

- You have a Windows 2000 computer at university with a default receive window of 17520 bytes
  - What is the maximum throughput available to a host 4ms away (say auckland to hamilton)
    - 17520 / 0.004 = 4,380,000 bytes/sec
  - What is the maximum throughput available to a host 150ms away (say google to hamilton)
    - 17520 / 0.150 = 116,800 bytes/sec

BDP examples 3

- You have a FreeBSD or Linux system at home with a default receive window of 262144 bytes
  - What is the maximum throughput available from youtube.com, 240ms away?
  - How does this compare with Windows 2000 w/ 17520 byte window size?

BDP examples 3

- You have a FreeBSD or Linux system at home with a default receive window of 262144 bytes
  - What is the maximum throughput available from youtube.com, 240ms away?
    - 262144 / 0.240 = 1,092,266 bytes / sec
  - How does this compare with Windows 2000 w/ 17520 byte window size?
    - 17520 / 0.240 = 73,000 bytes / sec

Summary

- Stop and wait is a simple protocol, but its performance is only good over very short round-trip-times
- Sliding window is better, but its performance depends on the size of the receiver's window and round-trip-time
- Next lecture:
  - Window scaling issues
  - TCP issues with Long Fat Networks