COMP312-09A
Communications and Systems Software

Lecture 6 – TCP II

Matthew Luckie
mluckie@cs.waikato.ac.nz

Overview

- Last lecture: sliding window protocols (TCP)
  - bandwidth delay product
- This lecture:
  - Delayed acknowledgements
  - Receive window issues
  - TCP options
  - window scaling
  - Maximum segment size

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

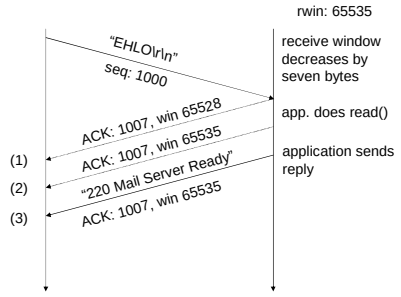
Window Size

- Specifies how much space remains in the receiver's receive buffer
- 16 bits in TCP header
  - i.e. can hold values 0 to 65535
- Each time application reads from its socket, the data comes out of the receive buffer
  - space available in the receive buffer increases by the number of bytes read
  - however, in some circumstances advertising an updated receive window immediately is a bad idea
  - in fact, we often want to wait a little time before sending an acknowledgement as well

Delaying TCP acknowledgements

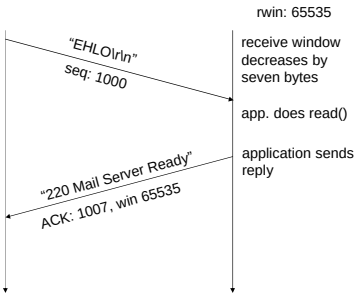
- In general, it is useful to wait a small length of time before sending an acknowledgement back to the sender
- Doing so allows us to combine an acknowledgement for received data with a window update, or a response from the application layer
- Delaying an acknowledgement has the effect of increasing the RTT on the path

Delayed acknowledgements



first ack redundant when second transmitted
first and second acks redundant when third transmitted

Delayed acknowledgements



Window size as limitation

- Recall that the window size specifies how much data the receiver can cope with sender transmitting
- TCP sliding window requires the window size to be the bandwidth delay product so that the window size is not limiting TCP performance
- Window size is 16 bits: 0 to 65535
  - 64KB receive window

100Mbps networks

- 100Mbps LANs are common place
- Can purchase 100Mbps of capacity from carrier network to link two offices
- How far away, in RTT, can these two offices be before a window size of 65535 bytes is insufficient?
  - ignore ethernet framing overhead
  - 8 bits in a byte.
  - 100Mbps = 100,000,000 bits per second

Bandwidth Delay Product

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

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

Bandwidth Delay Product

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

100,000,000 bits per second = 12,500,000 bytes per second

65535/12500000 = 0.0052428

5.24ms

Round trips:
Hamilton to Auckland: 4ms
Hamilton to Wellington: 17ms
Hamilton to ChCh: 19ms

1Gbps, 10Gbps networks

- 1Gbps network interfaces are in every laptop and desktop computer produced in the last 3 or 4 years
- Can purchase 1Gbps of capacity from carrier network
- KAREN (NZ Research and Education high-speed network) does this
  - KAREN is network used between NZ universities, and from NZ universities to international universities
  - up to 10Gbps capacity nationally
  - 622Mbps internationally

KAREN

- Kiwi Advanced Research and Education Network, owned by REANNZ
- Goal to allow researchers access large datasets and bandwidth intensive applications to carry out science
  - E.g. Data from telescopes in north america
  - Physics datasets
  - Packet header traces
- Reannz's 2008 annual report shows five of its 11.5 full-time equivalent staff received salaries of more than $120,000
- As of August 2008, only between 1 per cent and 4 per cent of that capacity was being used.
- http://www.stuff.co.nz/technology/176285

620Mbps

$$RTT = \frac{WindowSize}{BW}$$

620,000,000 bits per second = 77, 500,000 bytes per second

65535 / 77500000 = 0.0008456129

0.8ms

Possible remedies

- Open multiple TCP connections in parallel
  - Makes protocol implementation much more complicated than it should be
  - Some argument as to whether this is good for Internet health
  - This is basically the approach taken with bit torrent
    - Peers open multiple TCP connections to obtain the various portions of a file in parallel
- Window scaling option
  - Scale the window by powers of two
  - Receive window can be increased up to 1 gigabyte in size
  - Limitation: scaling factor has to be fixed, and decided at connection establishment
    - i.e. cannot apply window scale factor to existing TCP connection when window is found to be insufficient

Source Port												Destination Port
Sequence Number												
Acknowledgement Number												
HLEN						ACK		RST	SYN	FIN		Window Size
Checksum												
TCP options												

TCP options

- HLEN 4-bit field specifies the length, in 4 byte words, of the TCP header including any options
  - Minimum value of 5 (5*4 = 20, size of TCP header w/ no opts)
  - Maximum value 2^4 = 16, 16*4 = 60 bytes.
- Options have to begin on a 4 byte boundary

	Type	Len	Value
no-op	1		
max seg size	2	4	Maximum Segment size
window scale	3	3	scale factor

TLVs

- Type Length Value
- Protocol extension mechanism used in many network protocols
  - TCP
  - BGP
- If option is not understood, length value allows implementation to skip over that option to the next one it does understand

TCP window scale option

- Window scale value can only be advised in the SYN packet
  - Scale factor of 1 means window value can go up to 65535 x 2^1
    - 131070
  - Scale factor of 2 means 65535 x 2^2
    - 262140
- Receiver's window value held in 32 bit value
  - Receiver shifts this value right by the scale factor and transmits that 16 bit value.
  - i.e. window value of 240000 = 111010100110000000
  - Shift right by two: 1110101001100000

Maximum Window Scale Factor

- Maximum scale factor is 14
  - 65535 x 2^14 = 1GB
- This is because the sequence value wraps at 4GB.
- Having a larger scale factor runs the risk that if a packet is delayed a long time, retransmitted and then acknowledged, the sequence value will wrap so that when the original packet finally arrives, it could be used incorrectly

TCP Window Scale Option

- Java application specifies receive buffer as follows

```
Socket s;
s.setReceiveBufferSize(262140)
s.connect( ... )

ServerSocket ss;
ss.setReceiveBufferSize(262140)
ss.accept()
```


Encoding window scale factor 2

Source Port										Destination Port	
Sequence Number											
Acknowledgement Number											
HLEN 6					ACK		RST	SYN	FIN	Window Size	
Checksum											
Type (win scale) 3			Length (3 bytes) 3							Value (s/f 2) 2	no-op 1

Maximum Segment Size

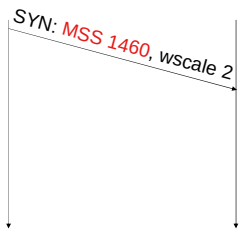
- By default, TCP assumes the receiver can receive data segments up to 536 bytes.
  - IPv4 implementations must be able to reassemble packets up to 576 bytes in size
  - 576 – 20 bytes (IP header) – 20 bytes (TCP header) is 536.
- Ethernet interfaces have a maximum transmission unit (MTU) of 1500 bytes. Can also receive frames of at least 1500 bytes.
- Bigger packets are more efficient to transmit than smaller ones
  - Cost of headers amortized over a larger amount of data
  - More efficient provided network can carry them without fragmenting them

MTU

```
[mluckie@sorcerer mjl]$ ifconfig
rl0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST>
        metric 0 mtu 1500
      options=8<VLAN_MTU>
      ether 00:05:1c:11:be:ff
      inet 130.217.250.39 netmask 0xffff0000
          broadcast 130.217.255.255
      media: Ethernet autoselect
          (100baseTX <full-duplex>)
      status: active
```

TCP MSS option

- Included in TCP SYN packet.
- Tells sender it can receive segments up to 1460 bytes



Summary

- Need for window scale becoming greater with high-delay, high-bandwidth paths
  - TCP provides an option to scale receive window up to 1GB in size
  - TCP options are a backwards-compatible way of improving TCP's performance
  - Other options: MSS option, SACK option (covered later)
- Next lectures
  - Fast Retransmit, Fast Recovery
  - More issues with long-fat-networks
  - Explicit congestion notification
  - Nagle