Week 8 - Wednesday

COMP 3400

Last time

- What did we talk about last time?
- Internet
- P2P architectures
- Transport layer:
  - UDP details

Questions?

Project 2

Transport Layer

TCP

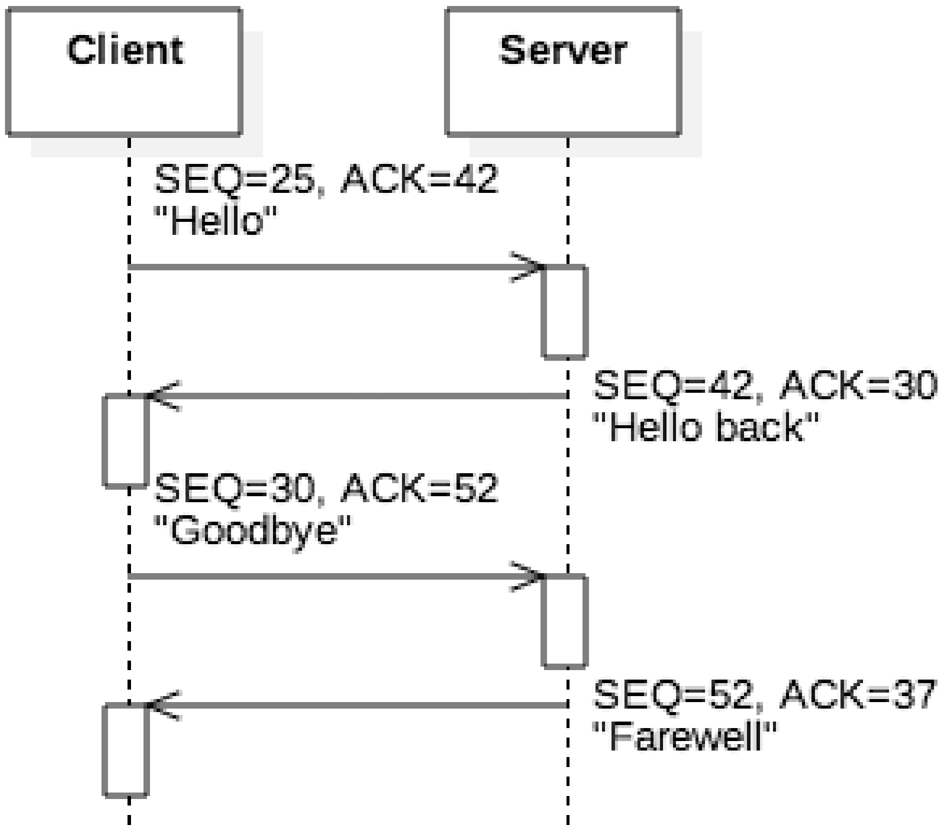
- Reliable transport is often desirable, so **Transmission Control Protocol** (**TCP**) is usually used for that purpose
- Unlike UDP, TCP creates a **session** with multiple messages sent back and forth between the two hosts
- Messages are numbered
- TCP also uses flow control, allowing hosts to avoid sending more data at once than their receivers can handle

TCP segments

- Because they have to do more, TCP segments contain more information:
  - Source port
  - Destination port
  - Sequence number (SEQ)
  - Acknowledgement number (ACK)
  - Flags
  - Receive window
  - Checksum
  - Urgent data pointer
  - Optional fields
  - Payload (actual data)
- Like UDP, most of these fields are 16 bits
  - SEQ and ACK are 32 bits
  - Optional fields vary
  - Payload is however long it needs to be

Numbering

- So that segments aren't lost, hosts send a sequence number (SEQ) with each segment
- The initial value is a random number ***k***
- After sending ***n*** bytes, the next SEQ will be ***n*** + ***k***
- So that the A knows how much B has gotten, B's next response to A contains an acknowledgement number (ACK) which is the last SEQ from A plus the size of that message
- In this way, both sides know how much the other side is sending, what's lost, and what's received
- If nothing is lost and messages are going back and forth, each SEQ will be the last ACK received



Flow control

- Buffers are always finite
- A TCP connection has a buffer that's reading information as it arrives from the other host
- Data is removed from this buffer as the process reads it from the socket
- If too much data is arriving, the buffer fills up, and data will be lost
- Each time a process sends a TCP segment, it also sends a **receive window** value, giving the number of bytes available in the buffer for that connection
- If there's not enough space for the next message, the sender will break its message into parts so that the part it sends will fit into the receive window

Example TCP segment

- The following is a TCP segment for an HTTP GET request

Header	1388 0050 0000 0017 0000 002a 5010 1000 cf33 0000	source port = 5000 (0x1388) destination port = 80 (0x0050) sequence number = 23 (0x17) acknowledgement number = 42 (0x2a) flags receive window = 4096 (0x1000) checksum urgent data ptr
Payload	4745 5420 2f20 4854 5450 2f31 2e31 0d0a 486f 7374 3a20 6578 616d 706c 652e 636f 6d0d 0a43 6f6e 6e65 6374 696f 6e3a 2063 6c6f 7365 0d0a 0d0a	GET / HTTP/1 .1\r\nHost: ex ample.com\r\nC onnection: c lose\r\n\r\n

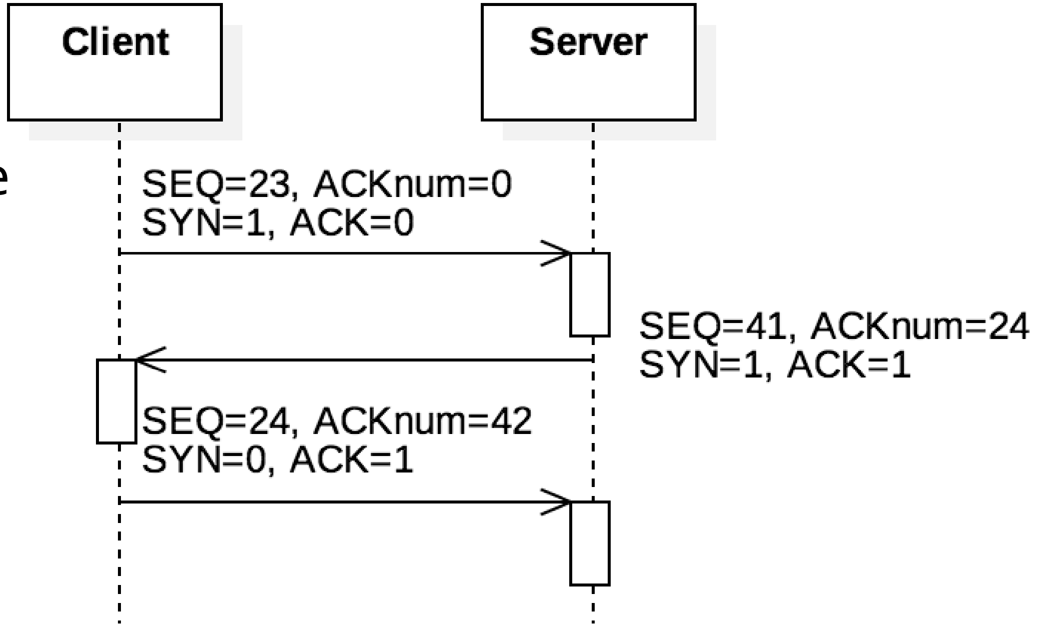
Flags

- The TCP header has a 16-bit flags field that can signal different information about a segment
- The book mentions six of these:
  - URG: Urgent pointer field is significant
  - ACK: Acknowledgment field is significant
  - PSH: Push buffered data to the receiving applications
  - RST: Reset the connection
  - SYN: Synchronize sequence numbers (set only in the first segment)
  - FIN: Last segment from sender

Index	0-3				4-9						10-15					
Meaning	Length of header in 32-bit words				Not explained here						URG	ACK	PSH	RST	SYN	FIN
Value	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
Hex	5				0				0				0			

TCP handshake

- When a TCP connection is being established by a client calling **`connect()`**, three segments are sent:
  - SYN (from the client)
  - SYN-ACK (from the server)
  - ACK (from the client)
- These segments are called the three-way handshake
- They are normal segments except that they have no data
- The SYN bit is set on the SYN and SYN-ACK segments, and the ACK bit is set on the ACK segment
- ACK is set on any segment intended to show that an earlier segment is being acknowledged



TCP handshake example

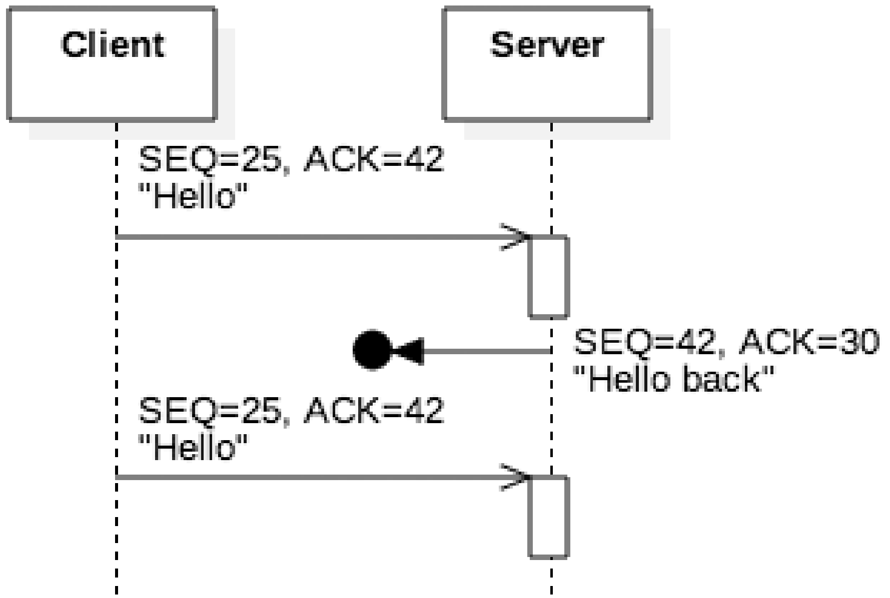
SYN Request (client to server)	`1388`
	`0050`
	`0000 136d`
	`0000 0000`
	`5002`
	`1000`
	`2e67`
	`0000`
		source port = 5000 `(0x1388)`
		destination port = 80 `(0x0050)`
		sequence number = 4973 `(0x136d)`
		acknowledgement number = 0
		flags = SYN
		receive window = 4096 `(0x1000)`
		checksum
		urgent data ptr

SYN-ACK Response (server to client)	`0050`
	`1388`
	`0000 0273`
	`0000 136e`
	`5012`
	`1000`
	`2be3`
	`0000`
		source port = 80 `(0x0050)`
		destination port = 5000 `(0x1388)`
		sequence number = 627 `(0x273)`
		acknowledgement number = 4973 `(0x136d)`
		flags = SYN and ACK
		receive window = 4096 `(0x1000)`
		checksum
		urgent data ptr

ACK Response (client to server)	`1388`
	`0050`
	`0000 136e`
	`0000 0274`
	`5010`
	`1000`
	`2bf6`
	`0000`
		source port = 5000 `(0x1388)`
		destination port = 80 `(0x0050)`
		sequence number = 4974 `(0x136d)`
		acknowledgement number = 628 `(0x274)`
		flags = ACK
		receive window = 4096 `(0x1000)`
		checksum
		urgent data ptr

Packet loss

- Using SEQ and ACK numbers with the checksum allows for error detection
- It's hard to be sure what went wrong, but some conclusions can be drawn:
  - Incorrect ACK: If the ACK is too small, the sender of the ACK missed one or more messages
  - Incorrect SEQ: If the SEQ is larger than expected, the receiver of the SEQ missed one or more messages
  - Incorrect checksum: The segment is corrupted or part is missing
- In all three cases, sending the last segment based on acknowledged data is a request for the other side to resend



Timeouts

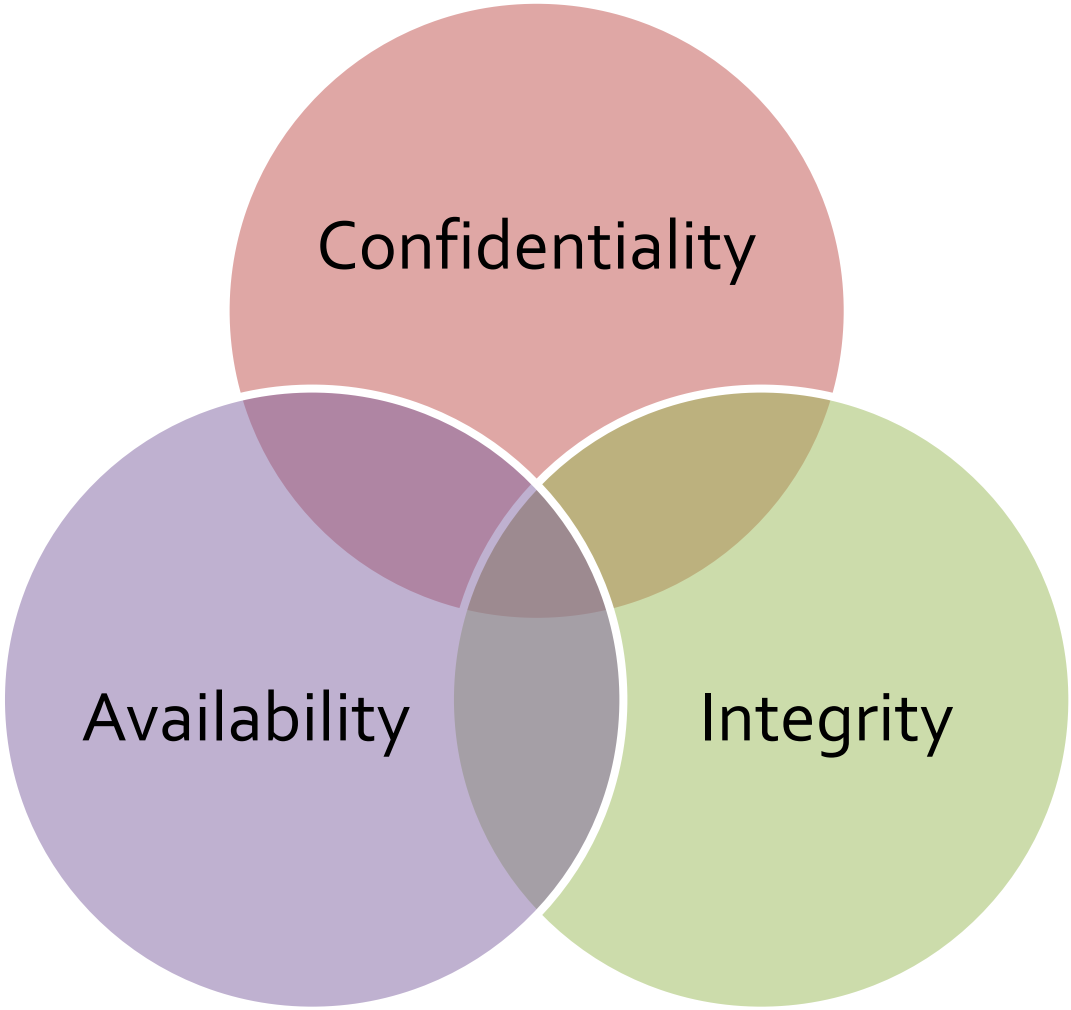
- To make robust guarantees about message delivery, TCP also keeps track of the time it takes for segments to make a trip
- If a segment is missing for long enough, TCP can request it again
- How long should it wait?
- Because the Internet is a large and heterogeneous place, it wouldn't make sense to wait for any particular fixed time
- Instead, the retransmission timeout (RTO) is computed based on previous transmission times and how much they fluctuate

RTT and SRTT

- Round-trip time (RTT) is the amount of time it takes to for a segment to be sent to another host and then receive a reply
  - RTT can change for each segment
- To estimate how long the next RTT is likely to be, TCP uses a smoothed round-trip time (SRTT), which is a weighted average of the old SRTT and the latest RTT
  - The new RTT is often weighted with $\frac{1}{8}$, but other values are possible
  - $SRTT' = \frac{7}{8}SRTT + \frac{1}{8}RTT$
  - Larger values would weight the most recent exchange heavier against history
- Using another weighted average, TCP keeps track of the variance of the RTT
- A final formula uses the expected RTT (the SRTT) and this variance to compute the current RTO

Network Security

CIA



- Network security is built on principles from general computer security:
  - Confidentiality
  - Integrity
  - Availability

Confidentiality

- **You don't want other people to be able to read your stuff**
  - Some of your stuff, anyway
- Cryptography, the art of encoding information so that it is only readable by those knowing a secret (key or password), is a principle tool used here
- Confidentiality is also called **secrecy** or **privacy**

Integrity

- **You don't want people to change your stuff**
- You want to know:
  - That your important data cannot be easily changed
  - That outside data you consider trustworthy cannot be easily changed either
- There are many different ways that data can be messed up, and every application has different priorities

Availability

- **You want to be able to use your stuff**
- Many attacks are based on **denial of service**, simply stopping a system from functioning correctly
  - A SYN flood where attackers try constantly to create TCP connections from spoofed IP addresses is a classic DoS attack
- Availability can mean any of the following:
  - The service is present in usable form
  - There is enough capacity for authorized users
  - The service is making reasonable progress
  - The service completes in an acceptable period of time

Ticket Out the Door

Upcoming

Next time…

- Finish network security
- Internet layer
- Link layer
- Wireless

Reminders

- Finish Project 2
  - **Due Friday by midnight!**
- Read sections 5.6, 5.7, and 5.8