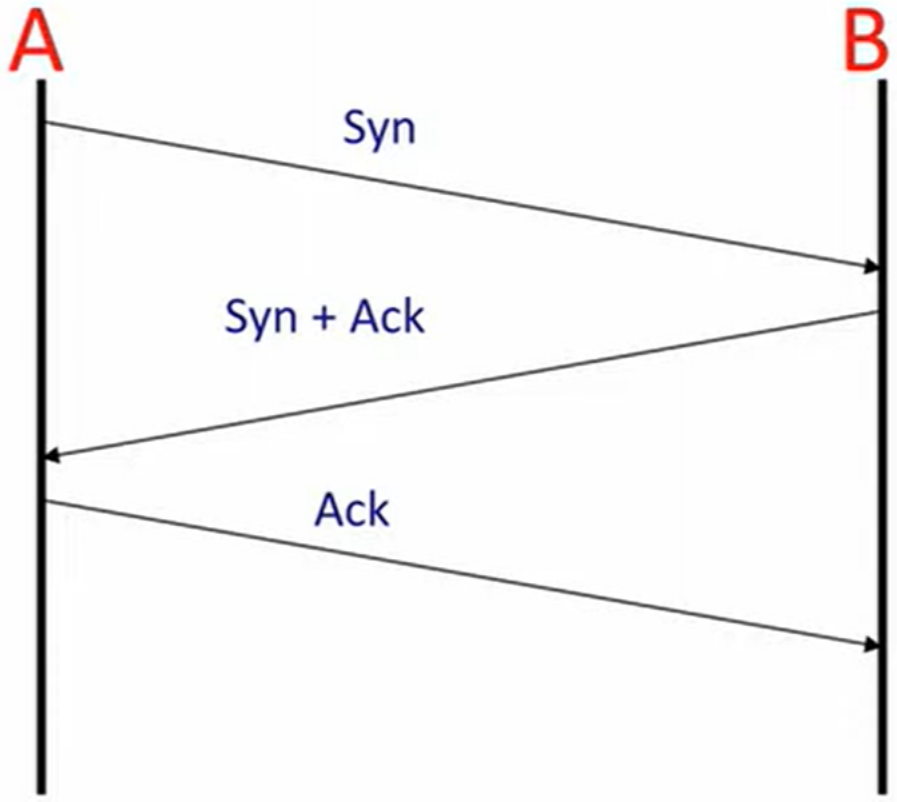
Transport Layer
The TCP Service Model

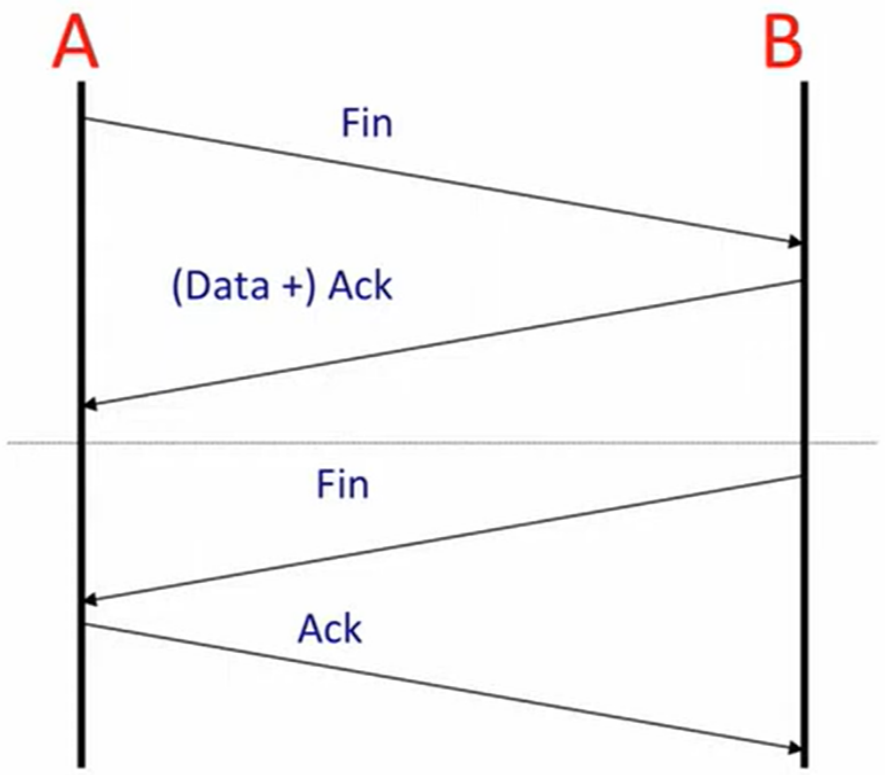
三次握手建立连接

Connection setup
3-way handshake



四次挥手断开连接

Connection teardown



The TCP Service Model

Property	Behavior
Stream of bytes	Reliable byte delivery service.
Reliable delivery	1. Acknowledgments indicate correct delivery. 2. Checksums detect corrupted data. 3. Sequence numbers detect missing data. 4. Flow-control prevents overrunning receiver.
In-sequence	Data delivered to application in sequence transmitted.
(Congestion Control	Controls network congestion.)

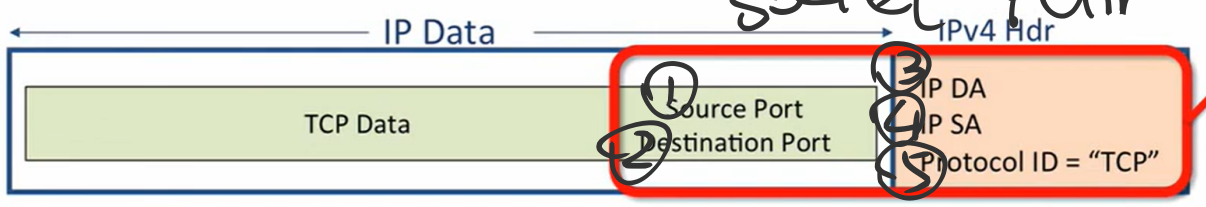
The TCP Segment Format

Bit 0 Bit 31

Source port	Destination port
Sequence # (of first byte)	
Acknowledgment Sequence #	
HLEN / RSVD / Flags (ACK URG PSH RST SYN FIN)	Window Size
Checksum	Urgent Pointer
(TCP Options)	
TCP Data	

The Unique ID of a TCP connection

Socket Pair

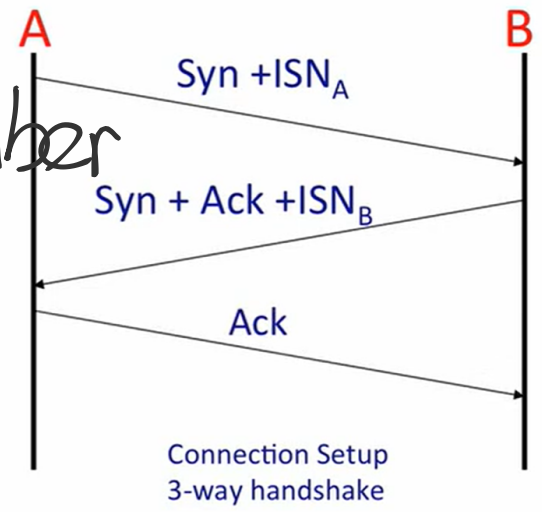


104-bit globally unique ID (Internet-wide)

1. Host A increments source port for every new connection

2. TCP picks ISN to avoid overlap with previous connection with same ID.

ISN means Initial Sequence Number

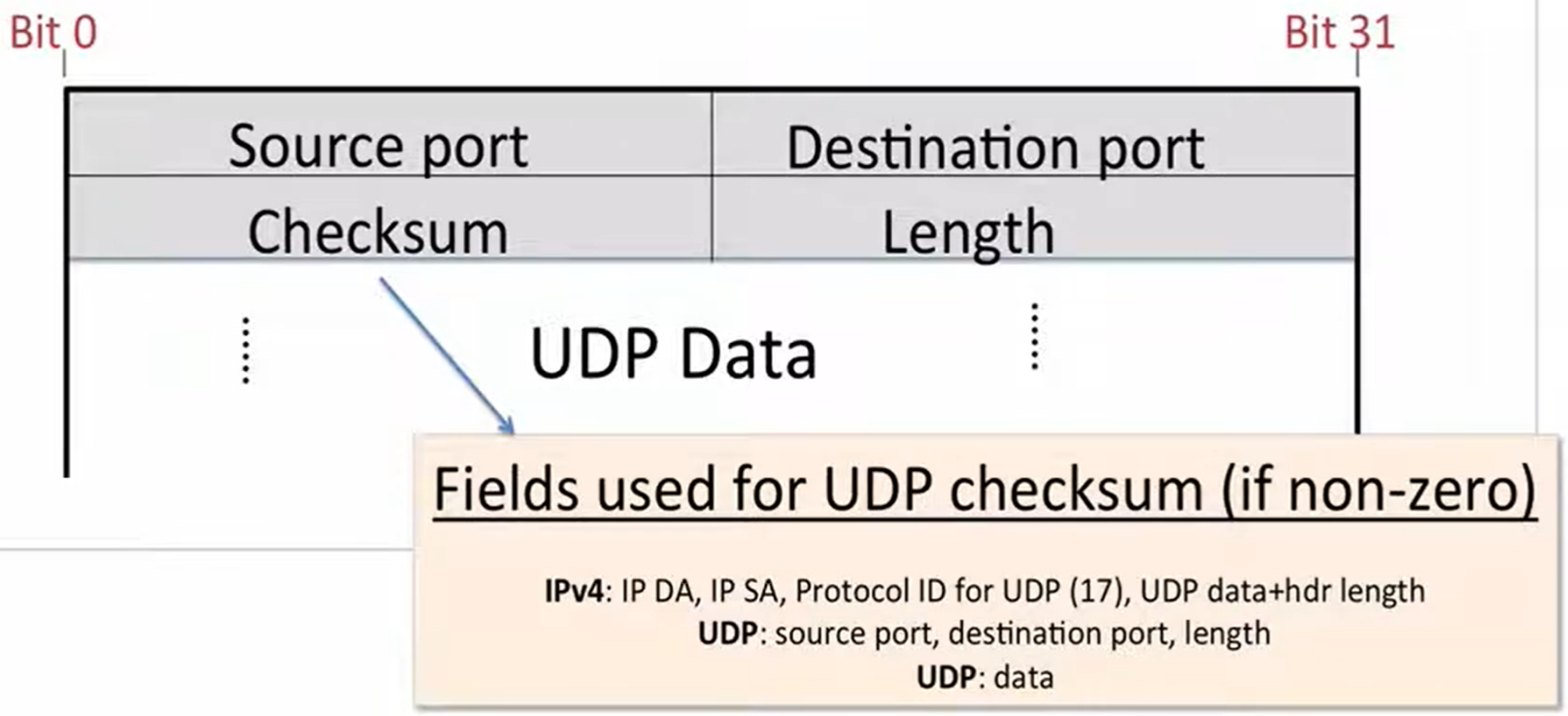


The UDP Service Model

↓

User Datagram Protocol

The UDP Datagram Format



Fields used for UDP checksum (if non-zero)

IPv4: IP DA, IP SA, Protocol ID for UDP (17), UDP data+hdr length
UDP: source port, destination port, length
UDP: data

User Datagram Protocol (UDP)

Property	Behavior
Connectionless Datagram Service	No connection established. Packets may show up in any order.
Self contained datagrams	
Unreliable delivery	1. No acknowledgments. 2. No mechanism to detect missing or mis-sequenced datagrams. 3. No flow control.

The ICMP Service Model

Internet Control Message Protocol

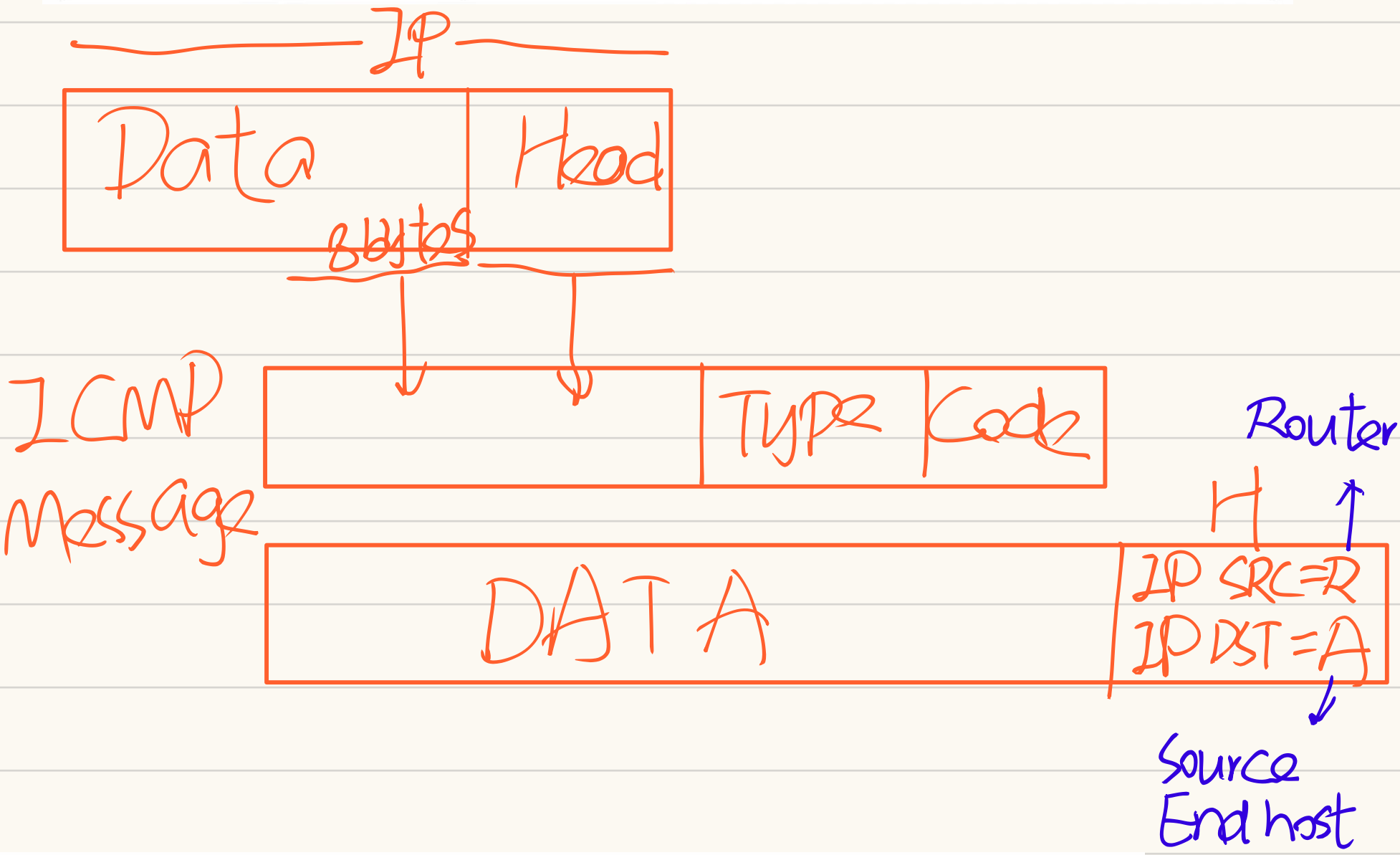
Making the Network Layer Work

三种不同的机制

1. The Internet Protocol (IP)
   - The creation of IP datagrams.
   - Hop-by-hop delivery from end to end.
2. Routing Tables
   - Algorithms to populate router forwarding tables
3. Internet Control Message Protocol (ICMP)
   - Communicates network layer information between end hosts and routers
   - Reports error conditions
   - Helps us diagnose problems

The ICMP Service Model

Property	Behavior
Reporting Message	Self-contained message reporting error.
Unreliable	Simple datagram service – no retries.



(Some) ICMP Message Types

ICMP Type	ICMP Code	Description
0	0	Echo Reply (used by ping)
3	0	Destination Network Unreachable
3	1	Destination Host Unreachable
3	3	Destination Port Unreachable
8	0	Echo Request (used by ping)
11	0	TTL Expired (used by traceroute)

① ICMP provides info about the network layer to end hosts and routers.

② It sits above IP and is therefore strictly a transport layer mechanism.

③ The commonly used tools "Ping" and "traceroute" both rely on ICMP.

The End-to-end principle

The End-To-End Principle

The function in question can completely and correctly be implemented only with the knowledge and help of the application standing at the end points of the communication system. Therefore, providing that questioned function as a feature of the communication system itself is not possible. (Sometimes an incomplete version of the function provided by the communication system may be useful as a performance enhancement.) We call this line of reasoning…"the end-to-end argument"

- Saltzer, Reed, and Clark,
End-to-end Arguments in System Design, 1984

However, this is generally seen as a performance optimization rather than a complete solution

Functions should primarily be implemented at the endpoints of a communication system rather than within the network itself.

"Strong" End to End

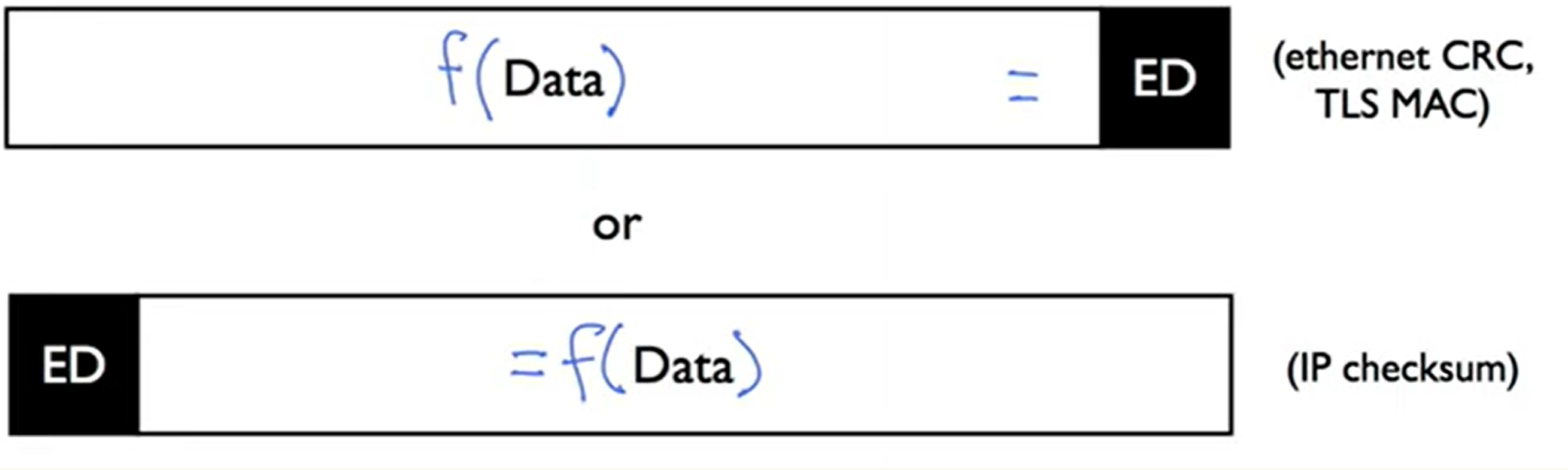
The network's job is to transmit datagrams as efficiently and flexibly as possible. Everything else should be done at the fringes…

– [RFC 1958]

should not be implemented in the middle

(strictly)

Error Detection: 3 schemes
checksum, CRC, MAC



- Checksum adds up values in packet (IP, TCP)
  - Very fast, cheap to compute even in software
  - Not very robust
- Cyclic redundancy code computes remainder of a polynomial (Ethernet)
  - More expensive than checksum (easy today, easy in hardware)
  - Protects against any 2 bit error, any burst $\leq c$ bits long, any odd number of errors
- Message authentication code: cryptographic transformation of data (TLS)
  - Robust to malicious modifications, but not errors
  - If strong, any 2 messages have a 2^{-c} chance of having the same code

IP Checksum

- IP, UDP, and TCP use one's complement checksum algorithm:
  - Set checksum field to 0, sum all 16-bit words in packet
  - Add any carry bits back in: 0x8000 + 0x8000 = 0x0001
  - Flip bits (0xc379 becomes 0x3c86), unless 0xffff, then checksum is 0xffff
  - To check: sum whole packet, including checksum, should be 0xffff
- Benefits: fast, easy to compute and check
  - Motivated by earliest software implementations
- Drawbacks: poor error detection
  - Only guarantees detecting a single bit error
  - Can detect other errors, but actual guarantees are both weak and complex

Cyclic Redundancy Check (CRC)

- Cyclic Redundancy Check (CRC): distill n bits of data into c bits, $c \ll n$
  - Can't detect all errors: 2^{-c} chance another packet's CRC matches
- CRC designed to detect certain forms of errors: stronger than checksum
  - Any message with an odd number of bit errors
  - Any message with 2 bits in error
  - Any message with a single burst of errors $\leq c$ bits long
- Link layers typically use CRCs
  - Fast to compute in hardware (details in a moment)
  - Can be computed incrementally
  - Good error detection for physical layer burst errors

Diversion: CRC Mathematical Basis

- Cyclic Redundancy Check (CRC): distill *n* bits of data into *c* bits, *c* << *n*
- Uses polynomial long division
  - Consider the message M a polynomial with coefficients 0 or 1 (pad with *c* zeroes)
    - E.g., M = 10011101 = $x^7 + x^4 + x^3 + x^2 + 1$
  - Use a generator polynomial G of degree *c* also with coefficients 0 or 1
    - Pad first term (always 1) for frustrating historical reasons
    - E.g. G = 1011 = $x^4 + x^3 + x + 1$
    - USB (CRC-16) = 0x8005 = $x^{16} + x^{15} + x^2 + 1$
  - Divide M by G, the remainder is the CRC: pick G carefully!
- Append CRC to message M: M' = M + CRC
  - Long division of M' with G has a remainder of 0

MAC

- Message Authentication Code (MAC)
  - Not to be confused with Media Access Control (MAC)!
- Uses cryptography to generate c = MAC(M, s), |c| ≪ |M|
  - Using M and secret s, can verify c = MAC(M, s)
  - If you don't have s, very very hard to generate c
  - Very very hard to generate an M whose MAC is c
  - M + c means the other person probably has the secret (or they're replayed!)
- Cryptographically strong MAC means flipping one bit of M causes every bit in the new c to be randomly 1 or 0 (no information)
  - Not as good for error detection as a CRC!
  - But protects against adversaries

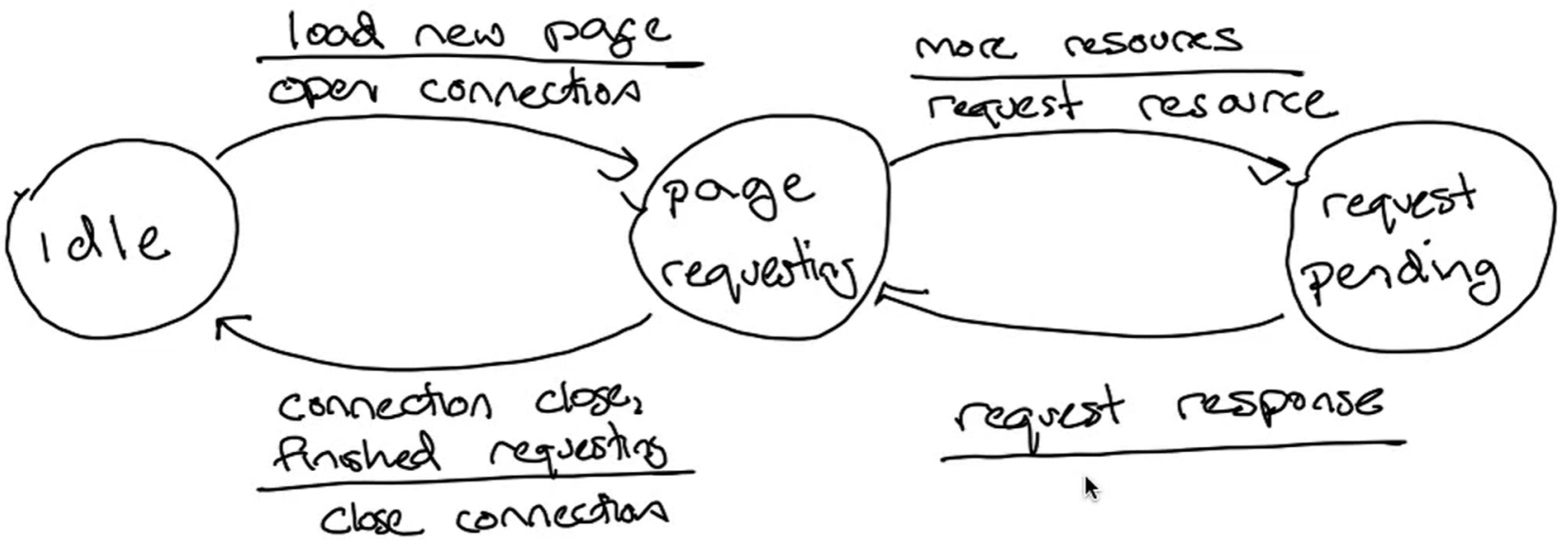
For each error detection algorithm, mark which errors it can guarantee catching, if any. The CRCs use a good generator polynomial and the MAC algorithm is cryptographically strong.

Algorithm	Single bit error	Run of 2 bit errors	Run of 9 bit errors	Two bit errors 100 bits apart
8-bit checksum	Y	N	N	N
16-bit checksum	Y	N	N	N
8-bit CRC	Y	Y	N	N
16-bit CRC	Y	Y	Y	N
32-bit MAC	N	N	N	N

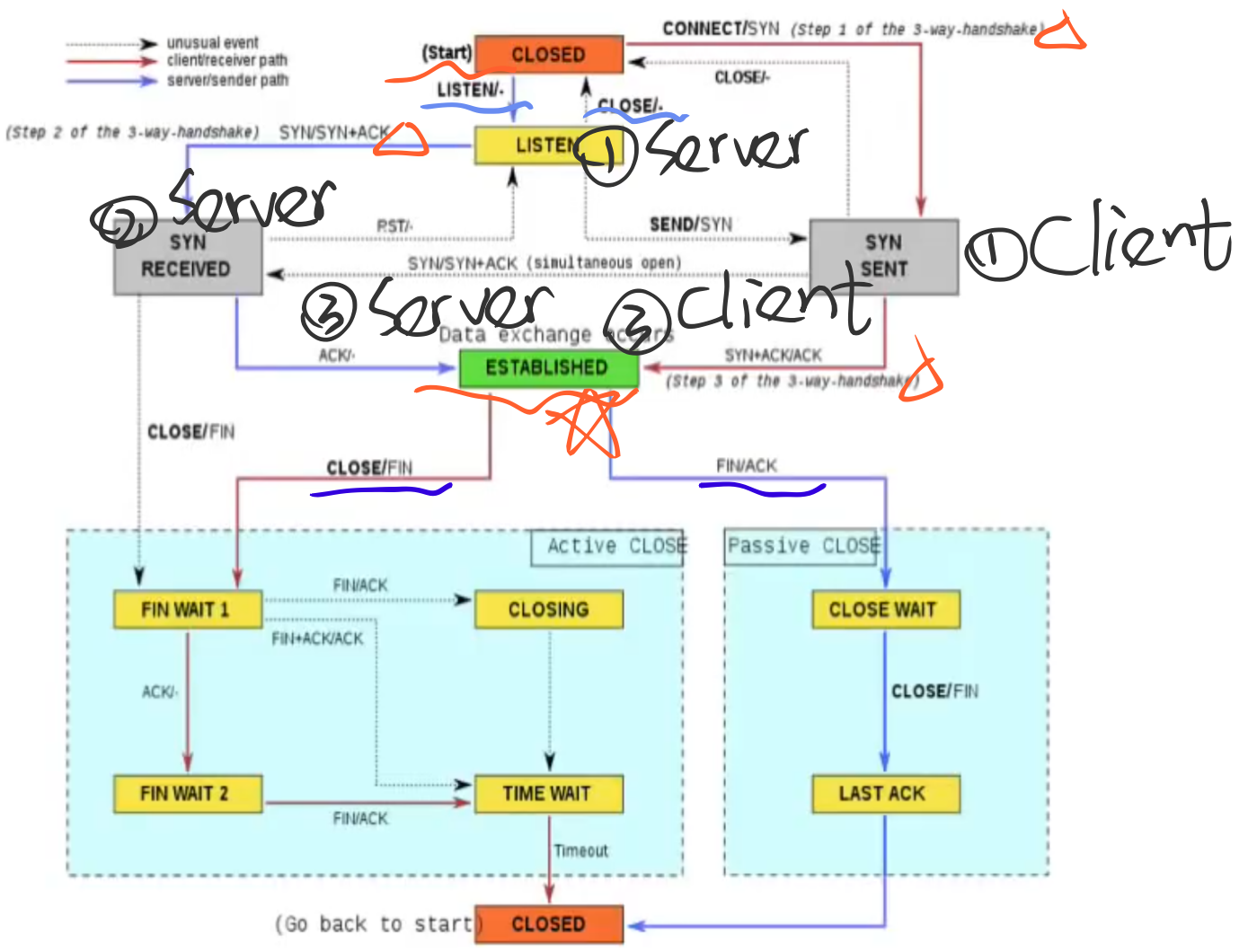
Finite State Machines

Protocol specification

FSM Example: HTTP Request



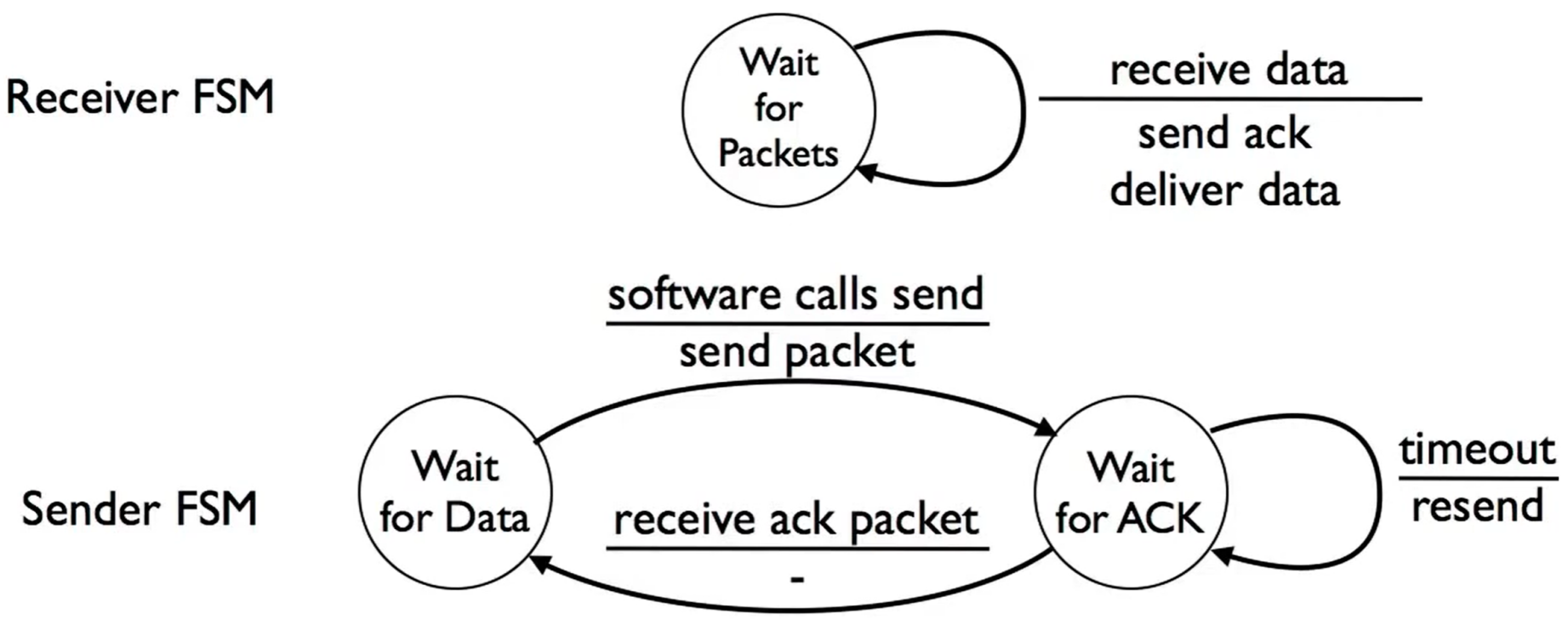
FSM Example: TCP Connection



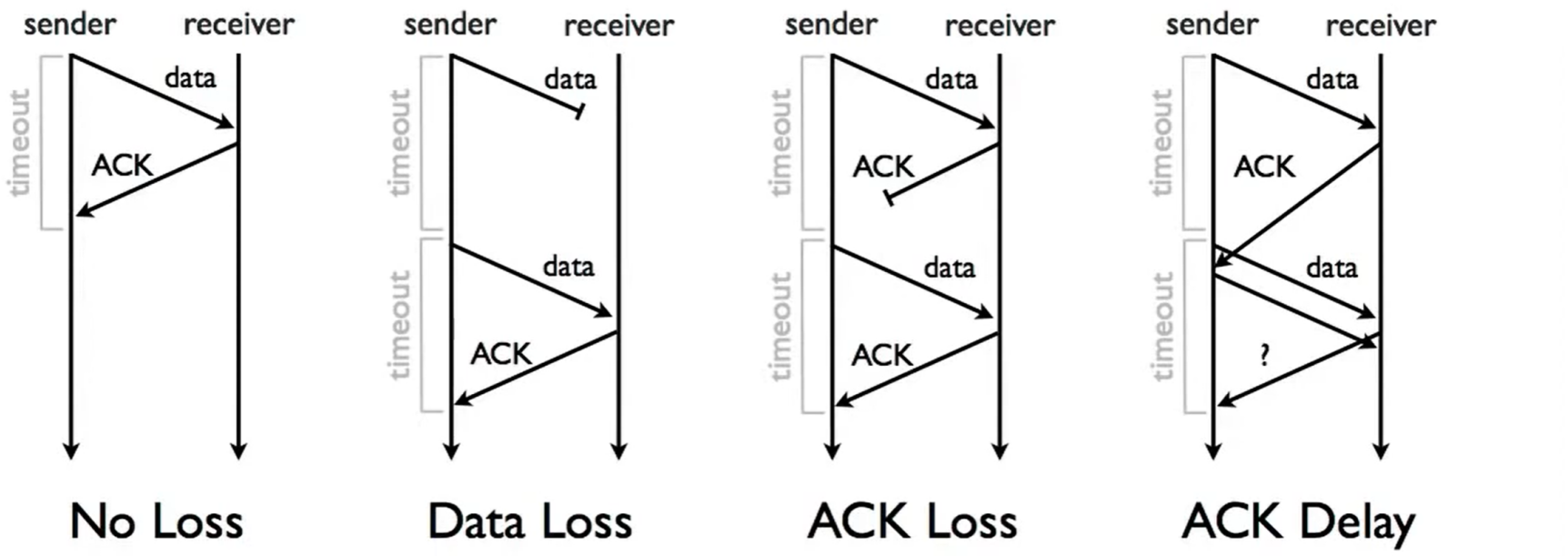
Flow Control

Stop-and-Wait

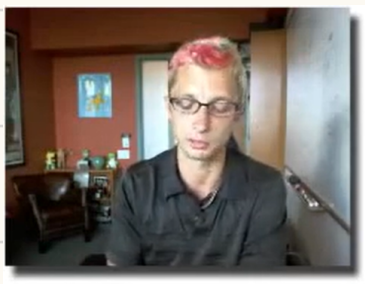
Stop and Wait FSM



Example Executions

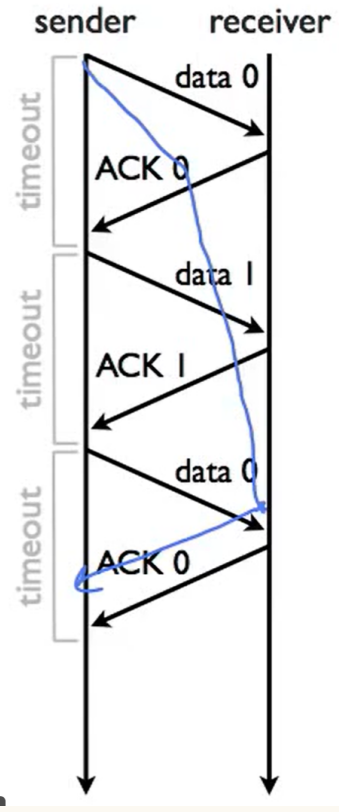


To Solve the ACK Delay



Duplicates

- Use 1-bit counter in data and acknowledgments
  - Receiver can tell if new data or duplicate
- Some simplifying assumptions
  - Network does not duplicate packets
  - Packets not delayed multiple timeouts



Conclusion:

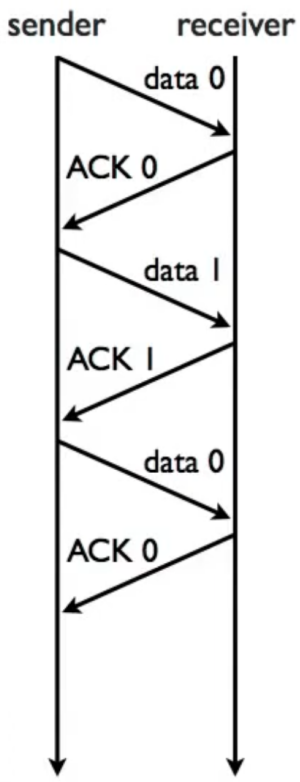
① At most one packet in flight at any time.
② Sender sends one packet
③ Receiver send ACK packet when it receives data.
④ On receiving ACK, sender sends new data
⑤ On timed out, sender sends current data
⑥ Use 1-bit counter to detect duplicates.

Stop and Wait & Sliding Window

generate

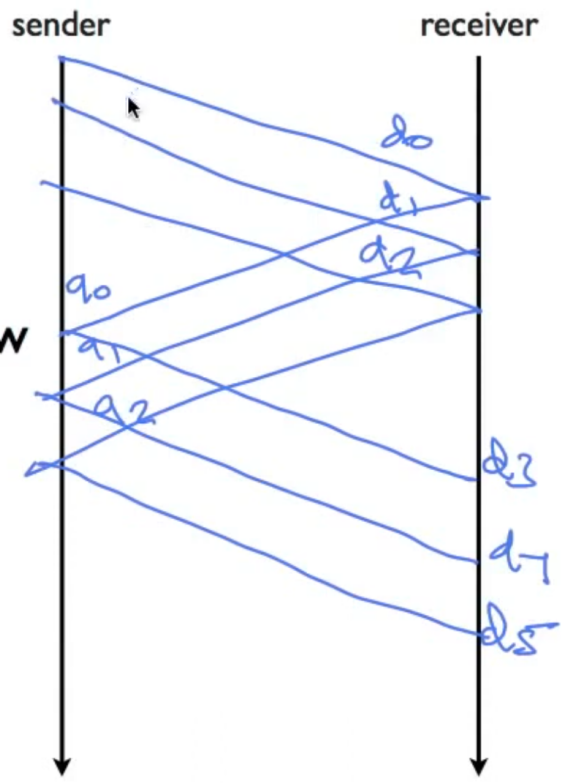
Example Execution

Stop and Wait



Sliding Window

$a = 3$



Sliding Window Sender

- Every segment has a sequence number (SeqNo)
- Maintain 3 variables
  - Send window size (SWS)
  - Last acknowledgment received (LAR)
  - Last segment sent (LSS)
- Maintain invariant: (LSS - LAR) ≤ SWS
- Advance LAR on new acknowledgment
- Buffer up to SWS segments

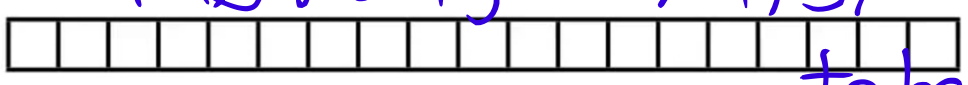
Also means:

$LSS \leq LAR + SWS$

for example

$SWS = 5$ $LAR = 11$

Then only 12, 13, 14, 15, 16 Allow to be sent



Sliding Window Receiver

- Maintain 3 variables
  - Receive window size (RWS)
  - Last acceptable segment (LAS)
  - Last segment received (LSR)
- Maintain invariant: (LAS - LSR) ≤ RWS
- If received packet is < LAS, send acknowledgment
  - Send *cumulative* acks: if received 1, 2, 3, 5, acknowledge 3
  - NOTE: TCP acks are next *expected* data (e.g., ack 4 in above example)

Also means:
LAS ≤ RWS + LSR
For example: if RWS = 5
LSR = 3
Then it can only accept 4, 5, 6, 7, 8

means that I've received 3 and everything before it, not just 3

RWS, SWS, and Sequence Space

- RWS ≥ 1, SWS ≥ 1, RWS ≤ SWS
- If RWS = 1, "go back N" protocol, need SWS+1 sequence numbers
- If RWS = SWS, need 2SWS sequence numbers
- Generally need RWS+SWS sequence numbers
  - RWS packets in unknown state (ACK may/may not be lost)
  - SWS packets in flight must not overflow sequence number space

ensure no buffer area is wasted

RWS = 3
SWS = 3
0, 1, 2, 3, 4, 5, 6, 7, 8
Ack2

Sliding Window Flow Control

1. Allow a "window" of unacknowledged packets in flight
2. When acknowledgment arrives, advance window
3. Required sequence number space size depends on window sizes.

Retransmission Strategies

Go Back N vs Selective Repeat

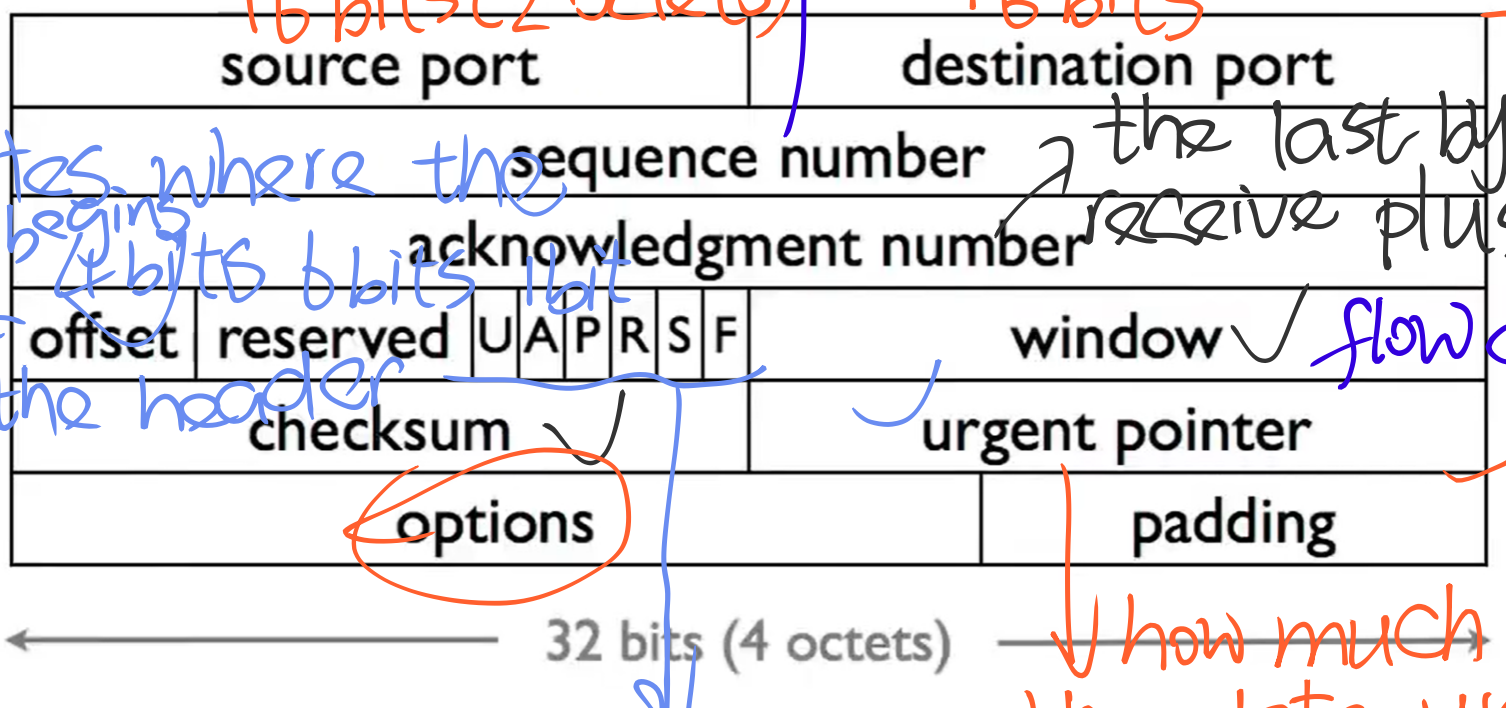
Pessimistic

Optimistic

Sender has a window of N
Receiver has a window of 1

Sender and Receiver both have a window of N

TCP Header



is of the first byte of the data region which follows the segment header

16 bits (2 octets)

16 bits

indicates where the data begins

the last byte receive plus 1

flow control

4 bits 6 bits flag after the header

how much of the data urgent

control bits:

(whether urgent) U → urgent, P → push

(whether buffer)

A → ACK, R → RST, S → SYN, F → FIN

The TCP header can have a varible length due to options field, that is the reason why "offset" is needed.

The smallest TCP header (without any options) is 20 bytes (5 words of 4 bytes each)

TCP setup and teardown

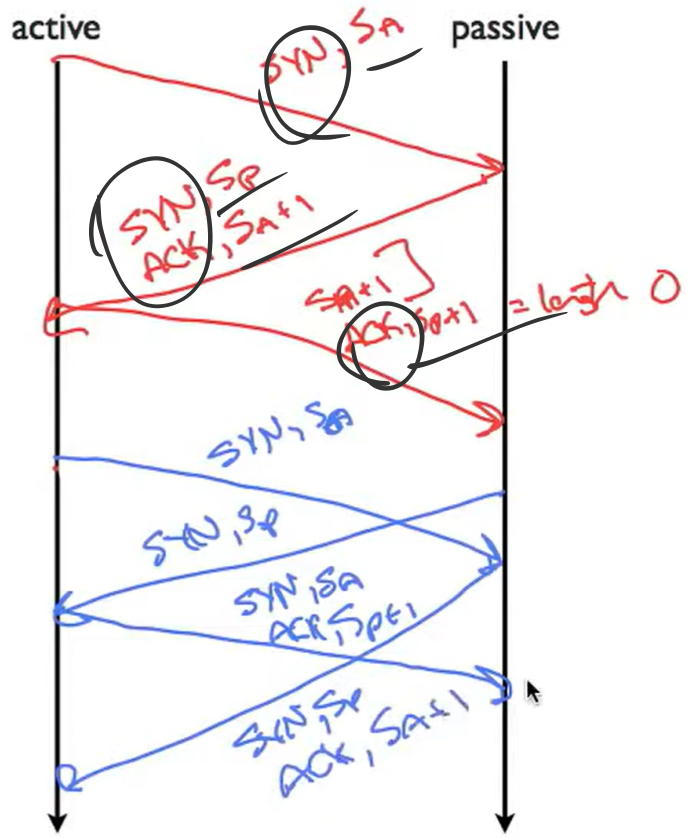
Connection Setup

source port								destination port
sequence number								
acknowledgment number								
offset	reserved	U	A	P	R	S	F	window
checksum								urgent pointer
options								padding

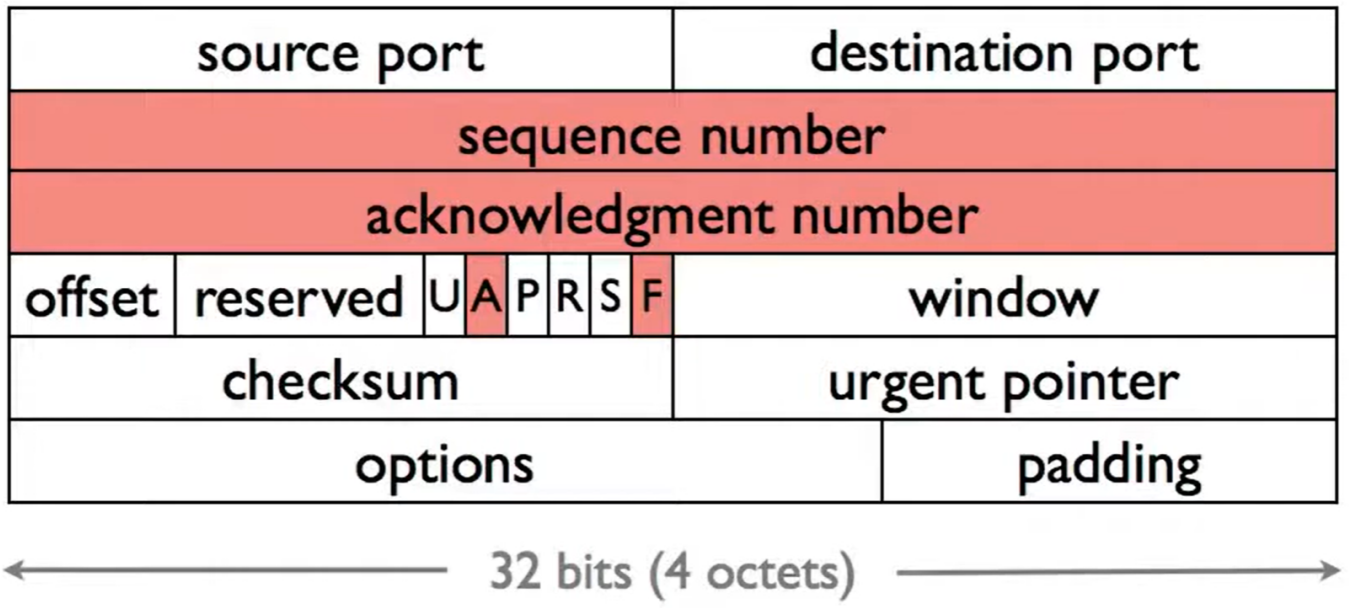
32 bits (4 octets)

3-way Handshake

- Active opener sends first packet
  - SYN with sequence number
- Passive opener responds
  - SYN with sequence number
  - ACKs active opener's SYN packet
- Active opener responds
  - ACKs passive openers SYN packet
- Also support "simultaneous open"
  - Two SYNs pass each other
  - Each side ACKs the other



Connection Teardown



Connection Teardown

- FIN bit says no more data to send
  - Caused by close() or shutdown() on other end
- Both sides must send FIN to terminate a connection
- Typical teardown exchange:
  - A → B: FIN, seq S_A, ack S_B
  - B → A: ack $S_A + 1$
  - B → A: FIN, seq S_B, ack $S_A + 1$
  - A → B: ack $S_B + 1$
- Can also have simultaneous close
- Can A and B forget about closed socket after final message?

bi-direction connection