# *Computer Networks and Distributed Systems*

## *Transport Layer*

Dr Fidelis Perkonigg

March 6, 2018

# Overview

- End-to-end communication using addresses and ports
- Transport protocols - UDP and TCP
- Connection control and establishment
- Congestion control

# Host-to-Host Communications

- Datagrams transferred between hosts
- Routed through networks based on IP addresses
  - Source address of sending machine
  - Destination address of recipient machine
- But:
  - No identification of which applications send or receive
  - Only unreliable connection-less datagram service

# Ports

- Use *ports* to define end points (services) within a host
  - 16 bit unsigned integer: 0 - 65535
  - Abstract addressing
- Applications use pairs of IP addresses and ports to communicate with each other
  - Use operating system calls to bind to port
  - Packets have source and destination ports

# Well Known Service Ports

- Other systems need to know which ports to connect to
  - Standard services usually run on well known ports
- List of well known services (RFC 1060)
  - Ports 0 - 255: Internet Assigned Numbers Authority (IANA)
  - Ports 0 - 1023: Privileged UNIX standard services

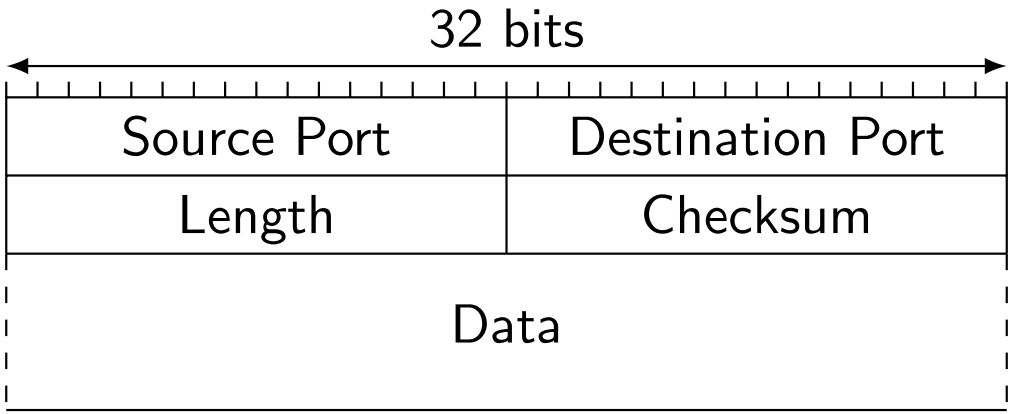
Examples:

| Port | Protocol | Use |
|---|---|---|
| 22 | SSH | secure remote login |
| 25 | SMTP | exchanging e-mails |
| 80 | HTTP | access to web pages |
| 110 | POP3 | remote email access |
| 115 | SFTP | secure file transfer |
| 123 | NTP | synchronising time |
| 143 | IMAP | remote email access |
| 443 | HTTPS | secure access to web pages |

- Static file with service/port mappings
  - Unix: /etc/services
  - Windows: windows/system32/drivers/etc/services

# User Datagram Protocol (UDP)

## Header Format

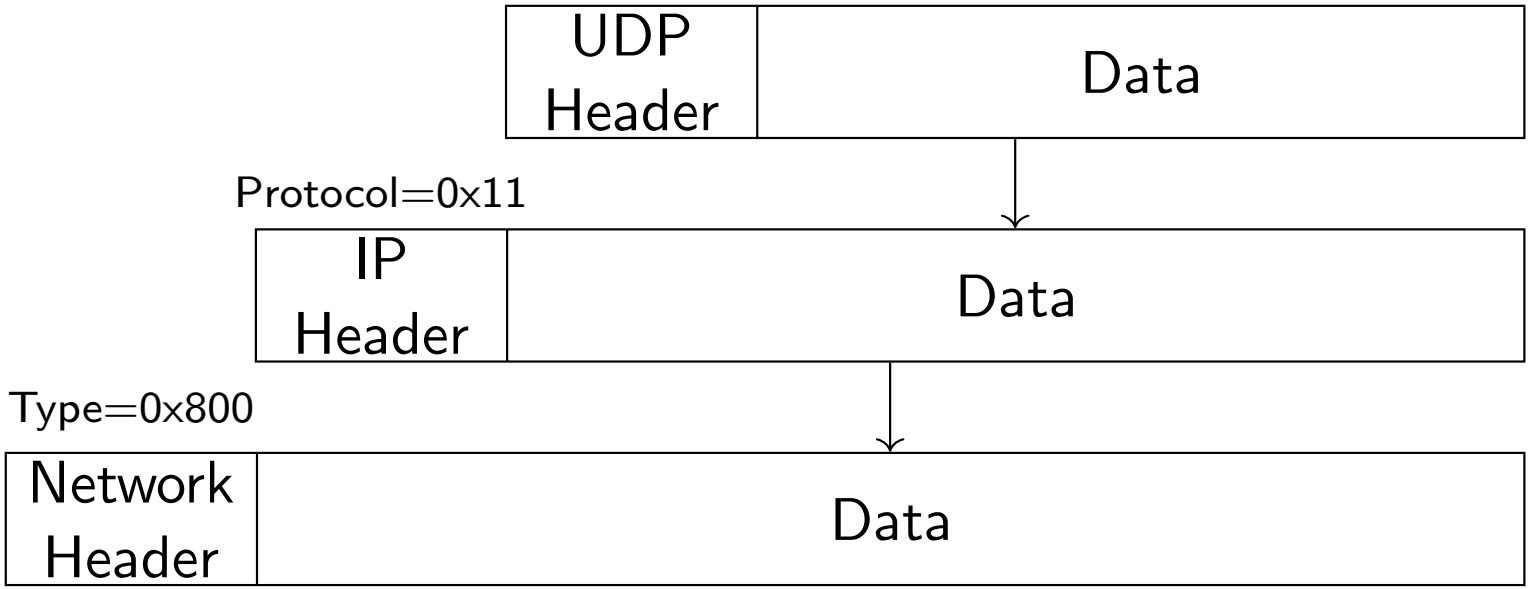



- Source port
  - Optional, 0 if not used, reply-to port if used
- Destination port
- Length (in bytes)
  - Includes header and data (min. 8 for no data)
  - Maximum data length depends on network layer protocol (e.g. IPv4)
- Checksum of header and data (optional)

# UDP Checksum

- Checksum (16 bit using 1s complement sum)
- Calculated over IP pseudo header + UDP header + data
- Pseudo header mimics IP header and includes
  - Source IP address
  - Destination IP address
  - Protocol (0x11)
  - UDP length
- Inclusion of pseudo header
  - Detection of changed IP headers by gateways/routers (incorrect package delivery?)
  - Violates the protocol hierarchy (IP addresses belong to the network layer)

# UDP

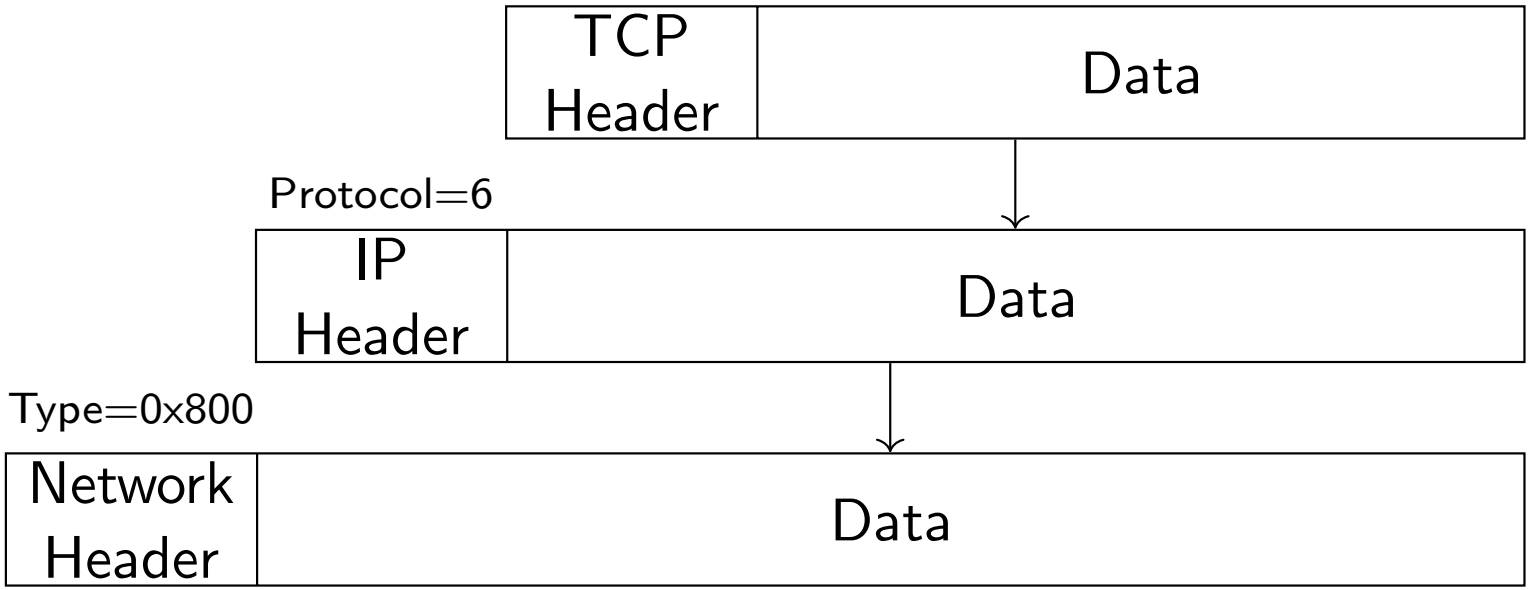



- UDP provides plain, IP-like service
- Connection-less datagrams, unreliable delivery, no sequence control
- Good for fast transfer with resilience to packet loss

# Reliable Service

- UDP is unreliable
- Could build reliable service over UDP
  - Needs to keep track of successfully transmitted packets
  - Add error-correcting and retransmission mechanisms

# Transmission Control Protocol (TCP)




- Provides connection-oriented service on top of connectionless IP
- Complexity and overhead

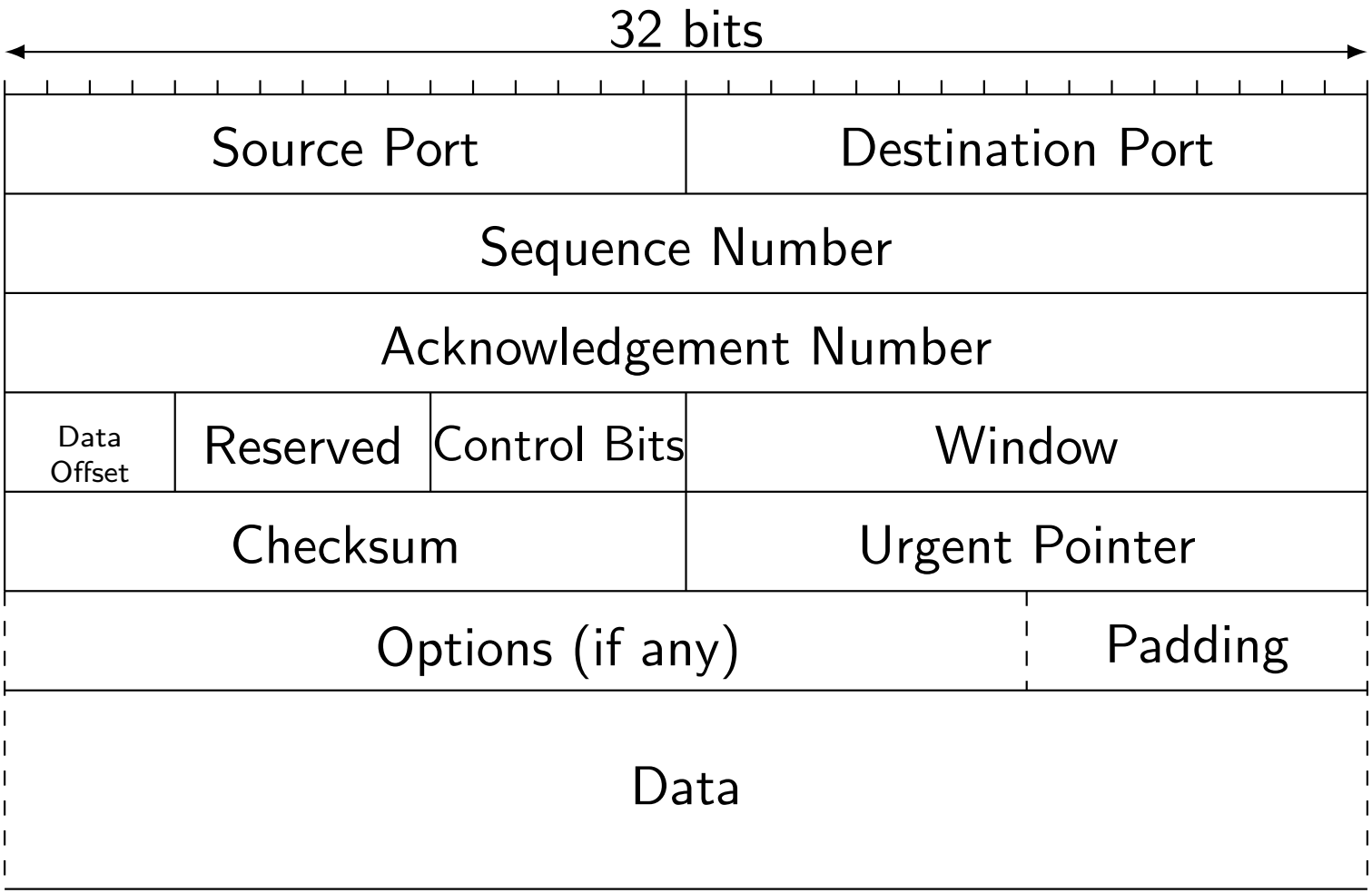
# TCP Features

- Streams
  - TCP data is stream of bytes
  - Underlying datagrams concealed
- Reliable delivery
  - Detects lost data and arranges retransmission
  - Stream delayed during retransmission to maintain byte sequence
- Flow control
  - Manages data buffers and coordinates traffic (between sender and receiver) to prevent overflows
  - Fast senders have to pause for slow receivers
- Congestion control
  - Monitors and learns delay characteristics of the network
  - Adjusts operation to maximise throughput without overloading network

# Connections

- TCP uses connections as its basic abstraction
- Endpoints are tuples: IP address and TCP port
  - Source: 146.169.15.121:1069
  - Destination: 140.247.60.24:25
  - Connections are identified by 5-tuple: protocol (TCP) + source IP + source port + destination IP + destination port
- Full-duplex (traffic in both direction at the same time)
- No multicasting or broadcasting supported

# TCP Segment Format



- Sequence Number
  - Indicates position in byte stream
  - Controls packet order, detects loss, and duplicates
- Acknowledgement Number
  - Acknowledge the receipt of data
  - Number of the next sequence number that is expected
  - Exploit full-duplex connection to send acknowledgments

# TCP Fields

- Data offset
  - Size of the TCP header and also offset to data
  - Needed because Options can be of variable length
- Window size used for flow control
  - Sender should not send more than window size
  - 0 means "no more data now, please"
- Checksum
  - Same as for UDP with pseudo header
- Urgent pointer
  - Pointer to high priority data in stream (e.g. error conditions)
- No data length
  - Can be calculated (IP datagram length)

- Control bits (flags)
  - **SYN**: synchronise sequence numbers
  - **ACK**: ACK number valid
  - **FIN**: sender has reached end of byte stream
  - **RST**: reset connection
  - **URG**: urgent pointer valid
  - **PSH**: push received segments promptly to application

# Open Connections

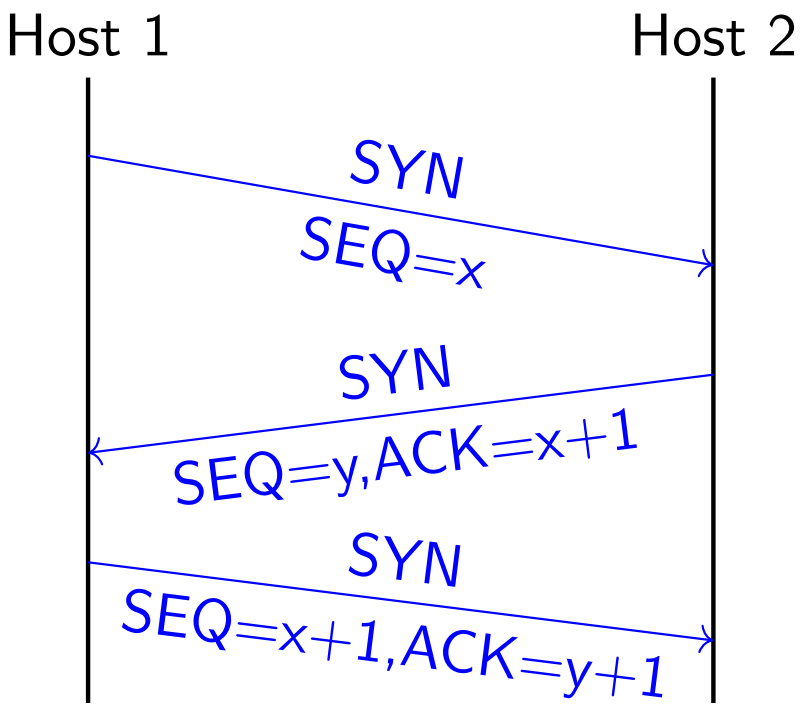
Passive open: Waits for incoming requests (servers)

Active open: Initiates communication (clients)

# Connection Control

- Sequence numbers to controls packet order, detects loss, and duplicates
- Two hosts must synchronise sequence numbers
- Initial sequence number (ISN) chosen randomly
  - Starting at 0 bad idea because of old packets
  - Need to be unique over life-time of connection
- Use SYN to establish connection
  - Establish initial sequence number
  - Stream positions are offsets from ISN
  - 1st data byte of stream = ISN + 1

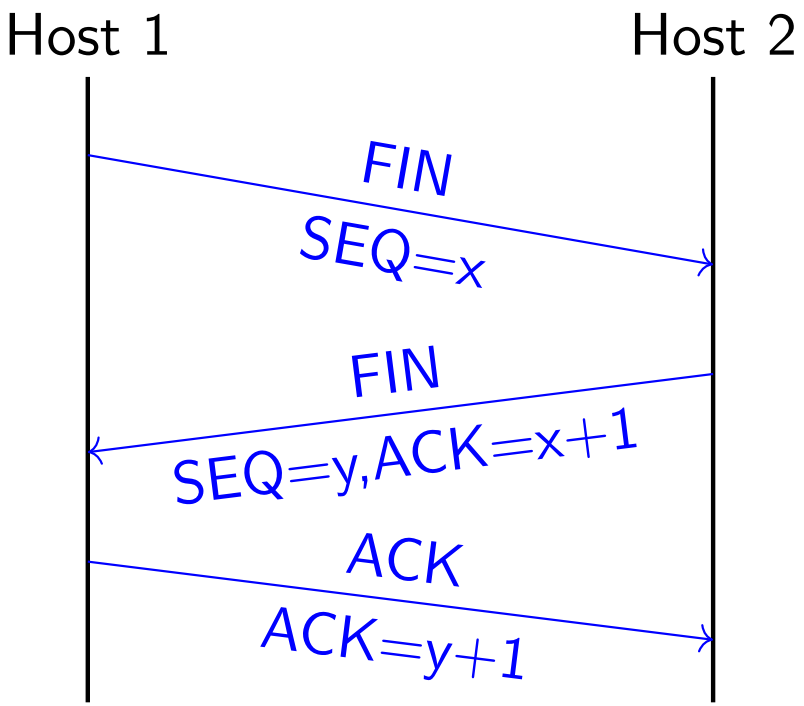
# Connection Establishment




Three way handshake:

- Sender and receiver agree on ISN
- Works when two hosts establish connection concurrently (only one connection is kept)
- SYN flooding: attacker sends many SYN packets

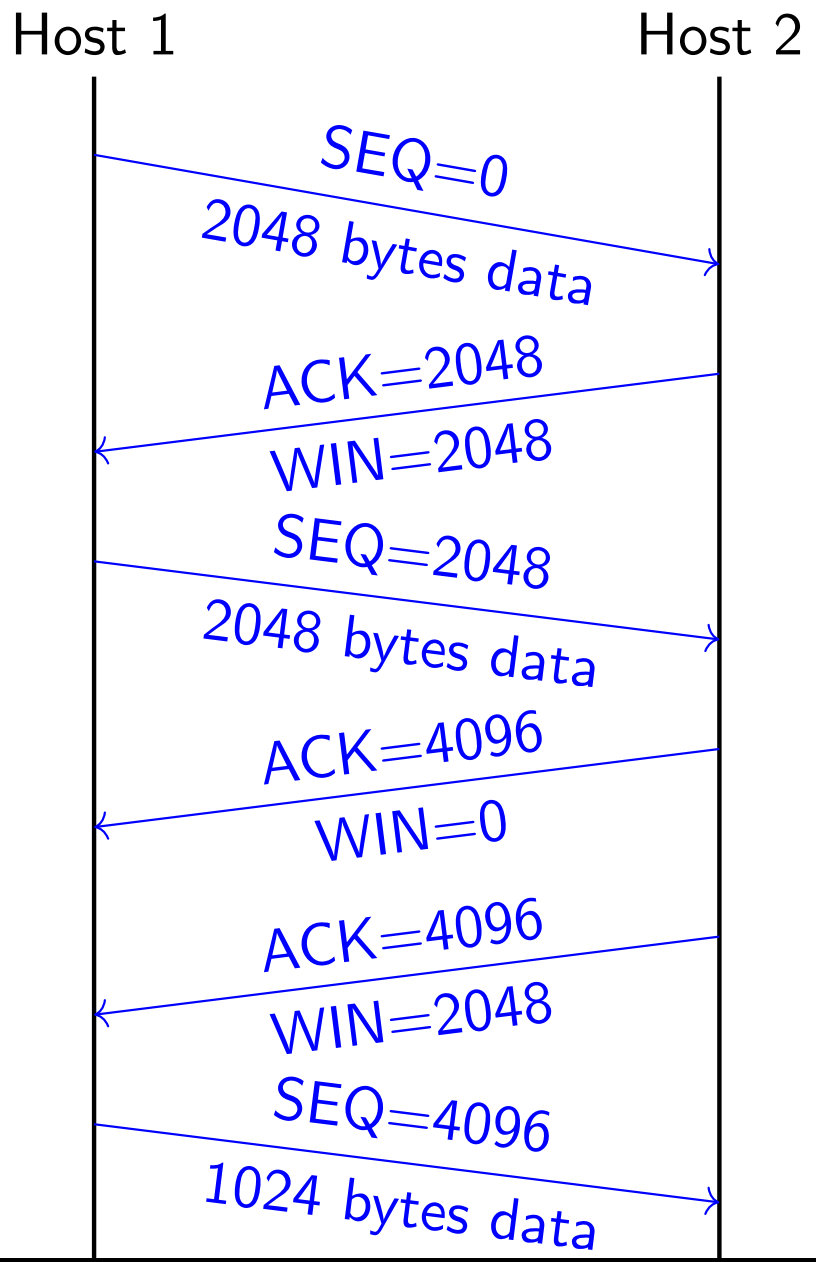
# Connection Release



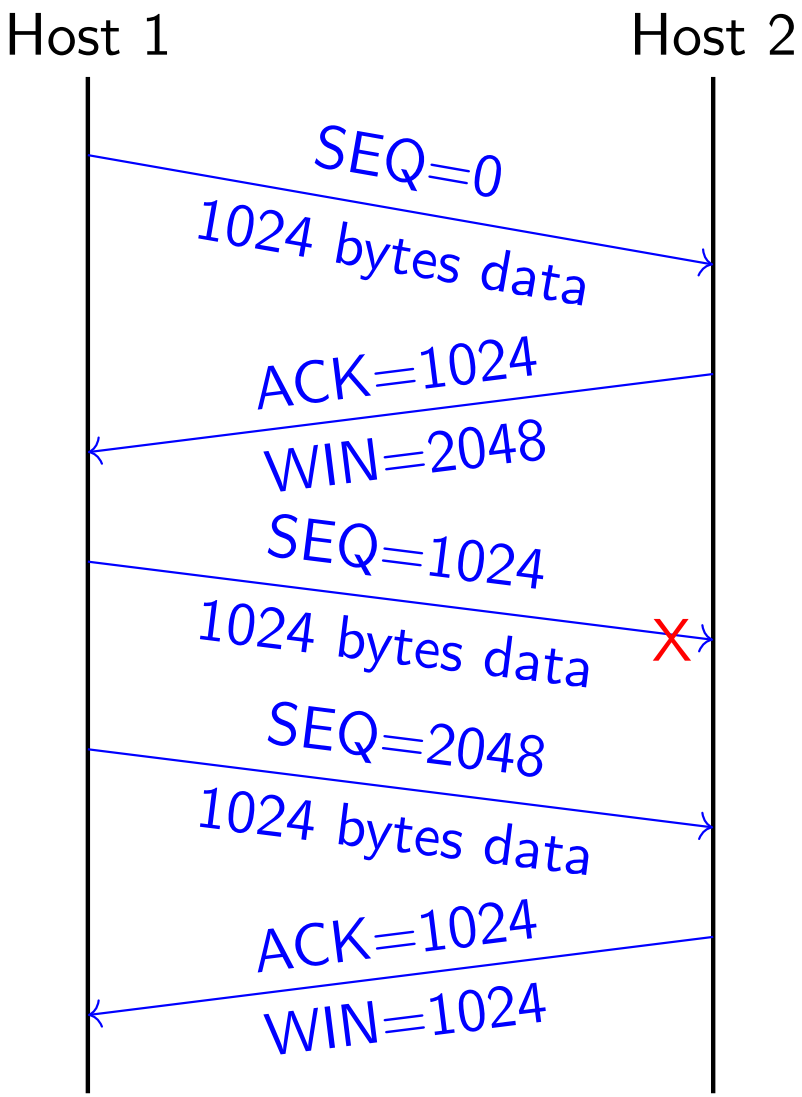

- Treat connection as two unidirectional connections
- Every endpoint releases connection
- Possible to release connection in only one direction (asymmetric). Data can still flow in other direction.
- ACK and FIN can be set in the same segment
- Hosts agree on end sequence number

# TCP Transfer and Flow Control

## Example




- Host 2 has buffer of 4096 bytes
- Sent data controlled by WINdow field
- Sender does not have to fill receiver's buffer with each segment
- SEQuence and ACKnowledgement indicate what has been sent/received
- Acknowledgements do not have to be sent immediately




- Retransmission triggered by:
- Timeouts
  - Sender resends data if timer[a] has timed out
  - Timer is reset whenever ACK has been received
- Duplicate ACK
  - Generated when received packets are out-of-order
  - Notifies the sender of expected data
  - Sender resends data if several duplicate ACK are received (fast retransmission)

[a] Retransmission TimeOut (RTO)

# Congestion

- Occurs when a network node (e.g. router) deals with more traffic than it can handle
- Routers use store-forward
  - Process each packet before sending
  - If buffer becomes full, drops packets
- Multiple incoming links can saturate single outgoing link
- Slower outgoing link can be saturated by one incoming link
- Goal is to send as fast as the slowest link permits (ACK clock)

# TCP Congestion Control

- Packet loss mostly due to congestion and not error
  - Detect congestion by considering packet loss
  - Change transmission rate to adapt to congestion
- TCP sender maintains 2 windows
  - Receiver Window (flow control) and congestion window
  - Uses whichever currently smaller

# TCP Congestion Window

- Congestion window based on network conditions
  - Windows grows and shrinks based on packet loss
  - Different algorithms for finding optimal size, e.g. additive increase, multiplicative decrease (AIMD)
- Requires efficient timeouts to detect loss
  - TCP measures RTT and adjusts timeouts

UDP:

- No need for reliability and error detection
  - Message exchanges without transactional behaviour, e.g. DNS, DHCP
  - Real-time applications, e.g. sensor monitoring, video live-streaming, VoIP
- Good for short communications
- Efficient for fast networks

TCP:

- Need for reliability, error correction, flow and congestion control, or better security
  - Terminal sessions, e.g. SSH, Telnet
  - Large data transfer, e.g. web, SFTP, e-mail
- Efficient for long-lived connections
- Requires more CPU time and bandwidth than UDP