5 Computer Networks Communications Architecture and Protocols

Part 11

The Transport Layer

Why Transport layer ?

- The Network layer
  - Can be unreliable as it spans LAN’s and WAN’s and users have no control over this subnet.
  - Some networks have poor service, bad routers (maybe they crash quite often), and
  - Poor error handling capabilities at the data link layer can vary from carrier to carrier, which in turn affects the quality of service provided by the network layer.

The transport layer functions

- shields the session layer from the vagaries of the underlying network mechanisms.
- provides a reliable and transparent data transfer mechanism
- provides a cost-effective service to users which is under their control

The *primary* function of the transport layer can also be considered to be that of enhancing the **quality of service (QoS)** provided by the network layer.

QOS Parameters (1)

- *Throughput* - number of bytes of user data transferred per second
- *Transit delay* - transit time between two transport entities
- *Connection establishment delay*
- *Connection establishment failure probability*
- *Residual error rate* - ratio of improper or lost data units to total transmitted

QOS Parameters (2)

- *Protection* - against third parties reading or modifying the data
- *Resilience* - probability of the transport layer itself terminating connection due internal problems
- *Priority* – a way for transport users to indicate the importance of some connections over others

Issues in a Reliable Transport Protocol

- Addressing
- Multiplexing
- Reliable data transfer
- Flow control
- Connection Management

Addressing

- Target user specified by:
  - User identification
    - Usually host, port number pair
      - Pair is called a *socket* in TCP
      - Port number is also called Transport service access point (TSAP)
    - Port represents a particular transport service (TS) user
  - Transport entity identification
    - Generally only one per host
    - If more than one, then usually one of each type
      - Specify transport protocol (TCP, UDP)
  - Host address
    - Internet-wide IP address

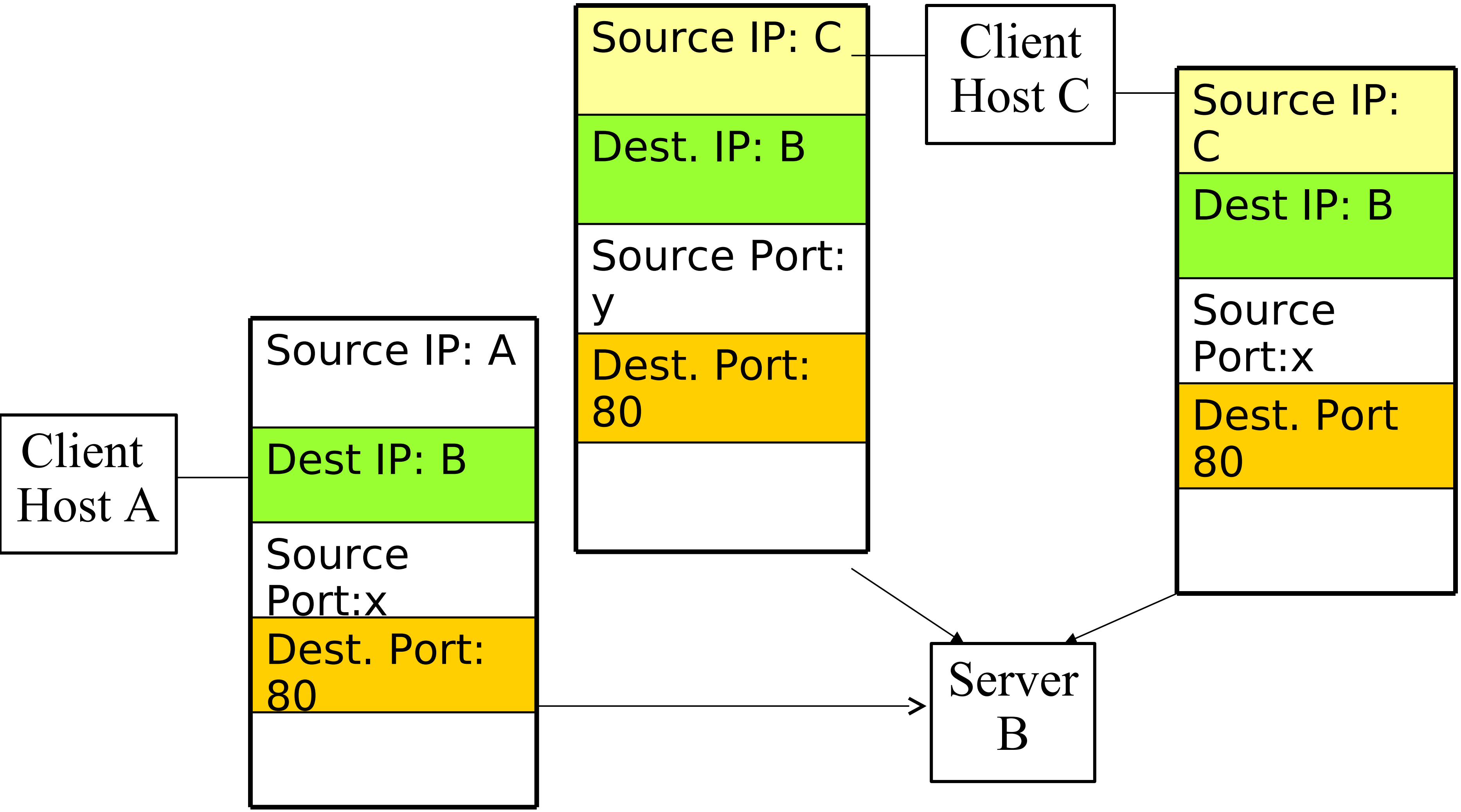
Finding Addresses

- Four methods
  - Know address ahead of time
    - e.g. collection of network device stats
  - Well known addresses
  - Name server
  - Sending process request to well known address

Multiplexing

- Multiple users employ same transport protocol
- User identified by port number or TSAP
- May also multiplex with respect to network services used
  - Single IP process serves more than one transport layer process.
- A transport layer segment (or datagram in case of UDP) always has a source port and destination port
  - This allows **multiplexing** and **Demultiplexing**
  - Example: well-known port numbers: FTP=21,

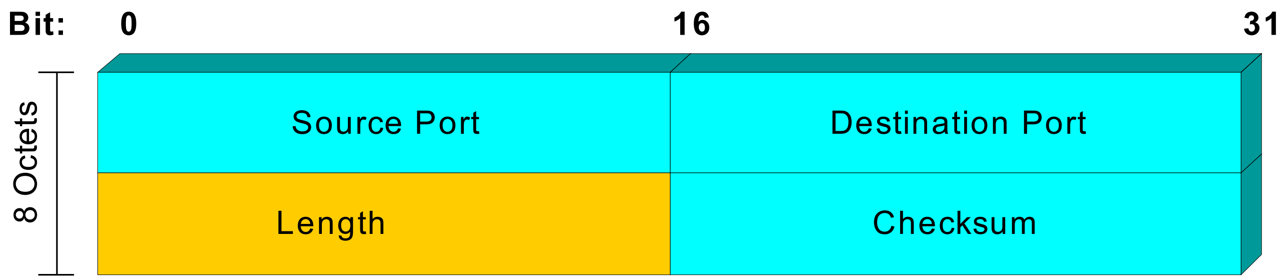
Multiplexing Example



User Datagram Protocol (UDP)

- Connectionless, unreliable transport protocol
- Provides no guarantee
- Datagrams may arrive out of order
- No flow control
- Best-effort forwarding
- Reduced overhead
- Fast and efficient

UDP Header



- Source and destination port numbers
  - The source and destination processes
- Length = length of header + data
- Checksum covers header and data
  - Optional in UDP but mandatory in TCP

UDP Uses

- Inward and Outward data collection/dissemination
  - **SNMP for network management**
  - **RIP routing table updates**
  - **NFS remote file server**
- Request-Response
  - Eg. DNS uses UDP for name translation
- Real time application
  - Streaming multimedia and internet telephony
  - Video conferencing

TCP Attributes(1)

- Provides a full-duplex bidirectional connection-oriented data transfer
- As seen by the user, data is transmitted as a byte stream (not in blocks) across a TCP connection.
- User data transmitted in *segments*
  - Implementation-dependent: 1500, 536, and 512 bytes typical

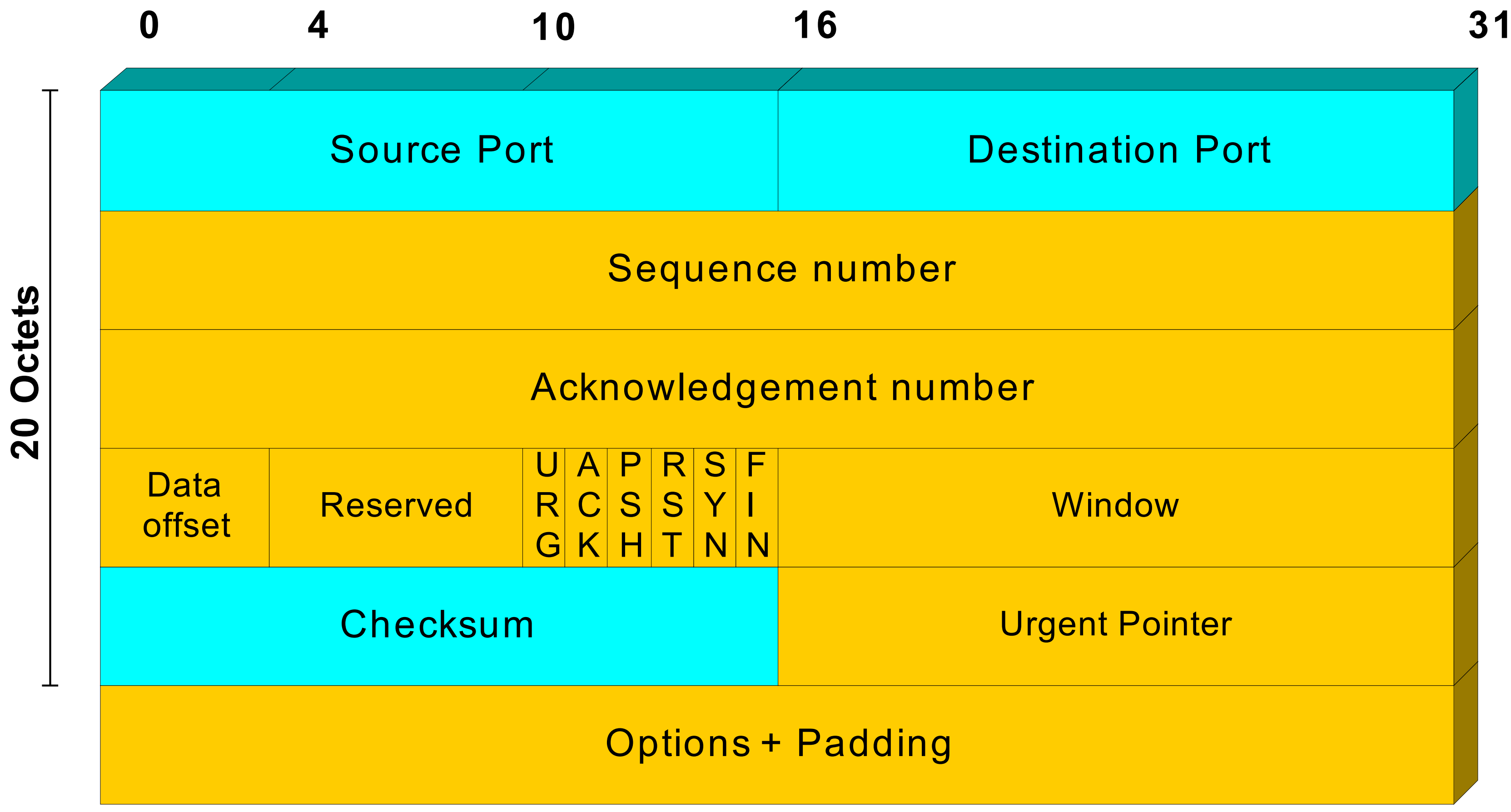
TCP Attributes(2)

- TCP endeavors to transfer data associated with exchanges over a connection
  - error-free,
  - with no losses or duplicates and,
  - in the same order as submitted

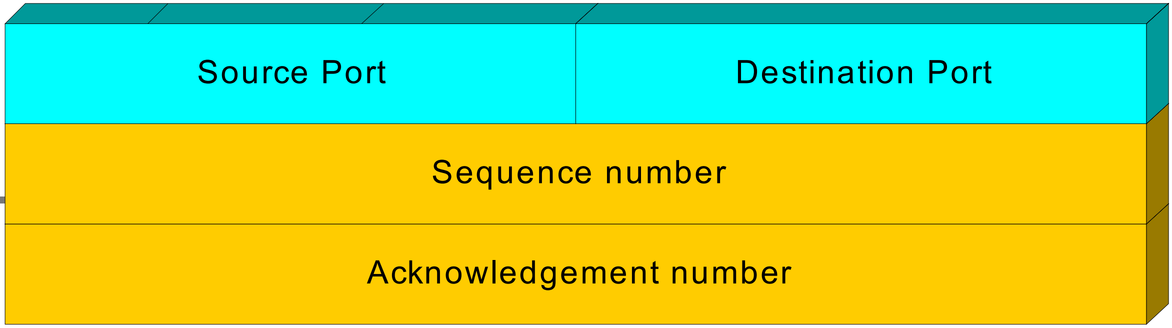
TCP Attributes (3)

- Reliable data transmission using:
  - Sequence numbers
  - Checksums
  - Acknowledgements with segment retransmissions after acknowledgements timeout
- Sliding window principle for flow control and efficiency
- Urgent data and Push functions
- Graceful connection shutdown

TCP Header

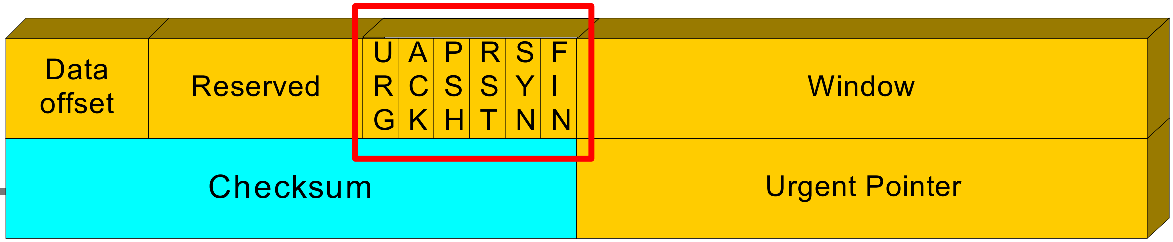

0
4
10
16
31
20 Octets
Source Port
Destination Port
Sequence number
Acknowledgement number
Data offset
Reserved
URG
ACK
PSH
RST
SYN
FIN
Window
Checksum
Urgent Pointer
Options + Padding

TCP Header (1)



- Source and destination port numbers – sending and receiving applications
- Sequence number: Byte number in the stream of data from the sender to receiver that the first byte of this segment represents
- Acknowledgement number: Next sequence number that the sender of the acknowledgement expects to receiver
  - Only valid if ACK flag is on
  - Costs nothing since its part of the header

TCP Header (2)



- Data offset: length of header in 32-bit words
  - Required since options field is variable
- Flag fields
- Window size is used for flow control
- Checksum: covers header and data and is mandatory
- Urgent pointer – points to urgent data in the byte stream
  - Only valid if URG is set

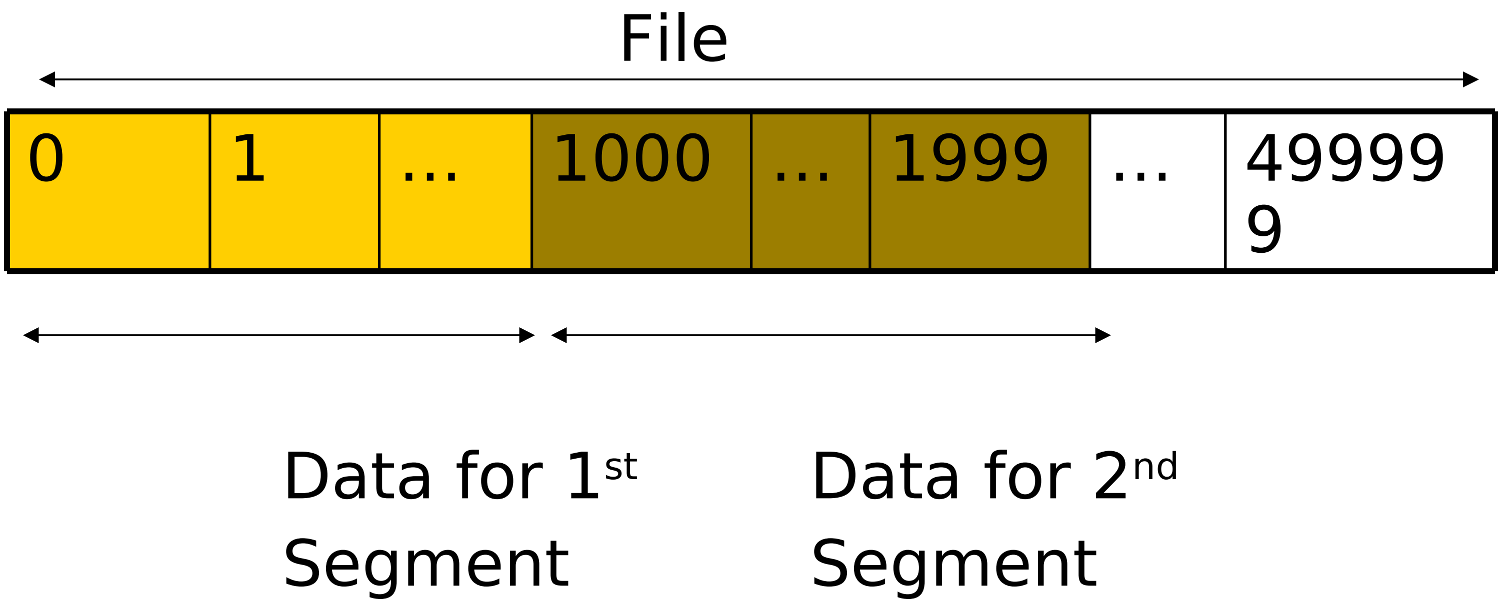
TCP Header (3)

U	A	P	R	S	F
R	C	S	S	Y	I
G	K	H	T	N	N

- Flags Field – 6 bits
  - URG – there is urgent data in this segment – inform upper layer entity and use urgent pointer
  - ACK bit shows value in acknowledgement field is valid
  - PSH – receiver should pass data to upper layer immediately
  - RST, SYN and FIN used for connection setup and shutdown – see later
- Options & Padding
  - Mainly used to specify maximum segment size (MSS)

TCP: Reliable data transfer (1)

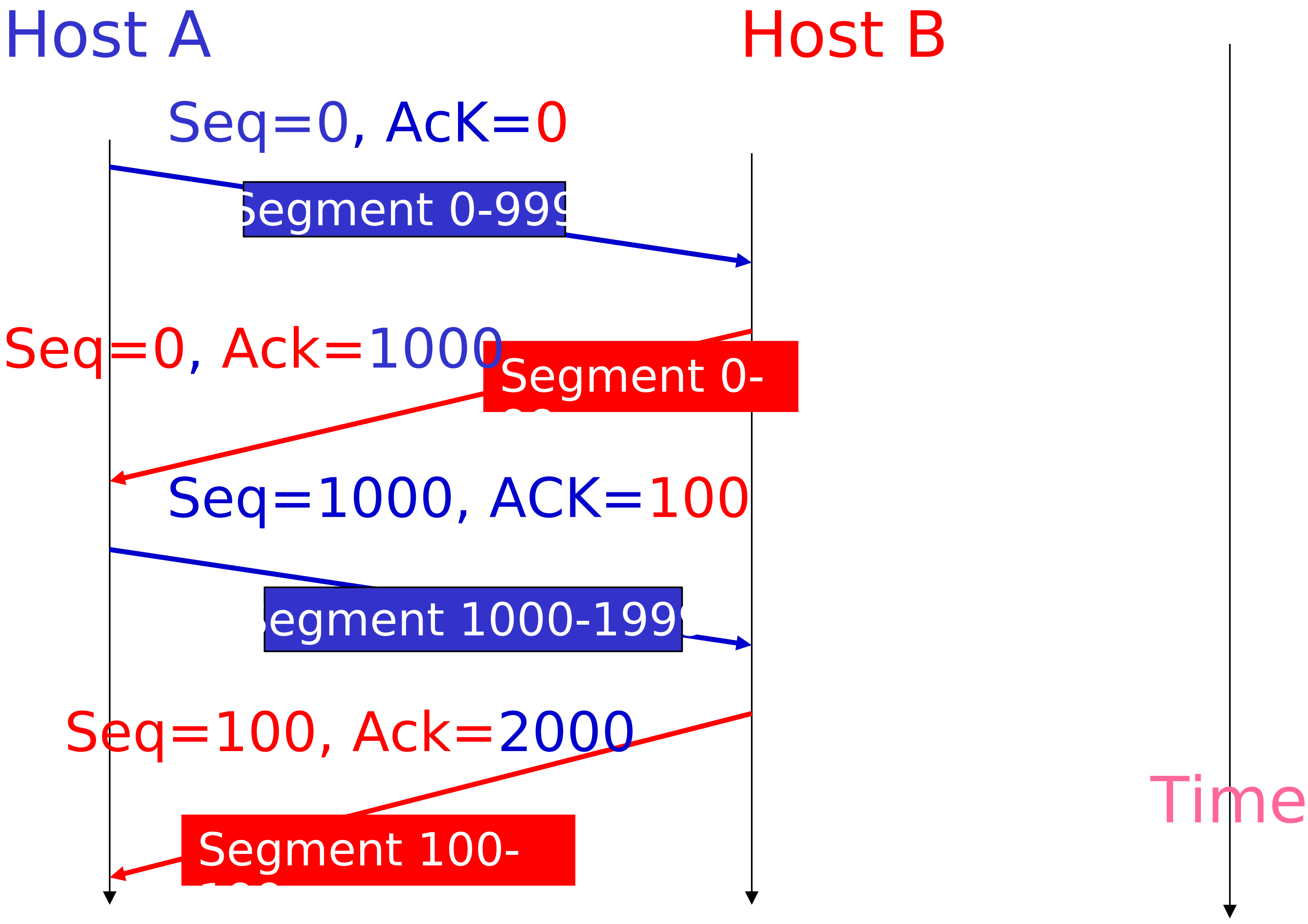
TCP Segments



Reliable Data Transfer (2)

- Data transfer
  - Logical stream of octets
  - Octets numbered sequentially modulo 2^{32}
  - Segments are numbered by the first octet number in the segment
  - Send Sequence Number
    - First byte number in current segment (on payload)
  - Acknowledgement Sequence number
    - ***Next*** byte number that receiver expects
- Need orderly data delivery
- Flow control via window size (see later)
- Data buffered at transmitter and receiver

Reliable Data Transfer (3)



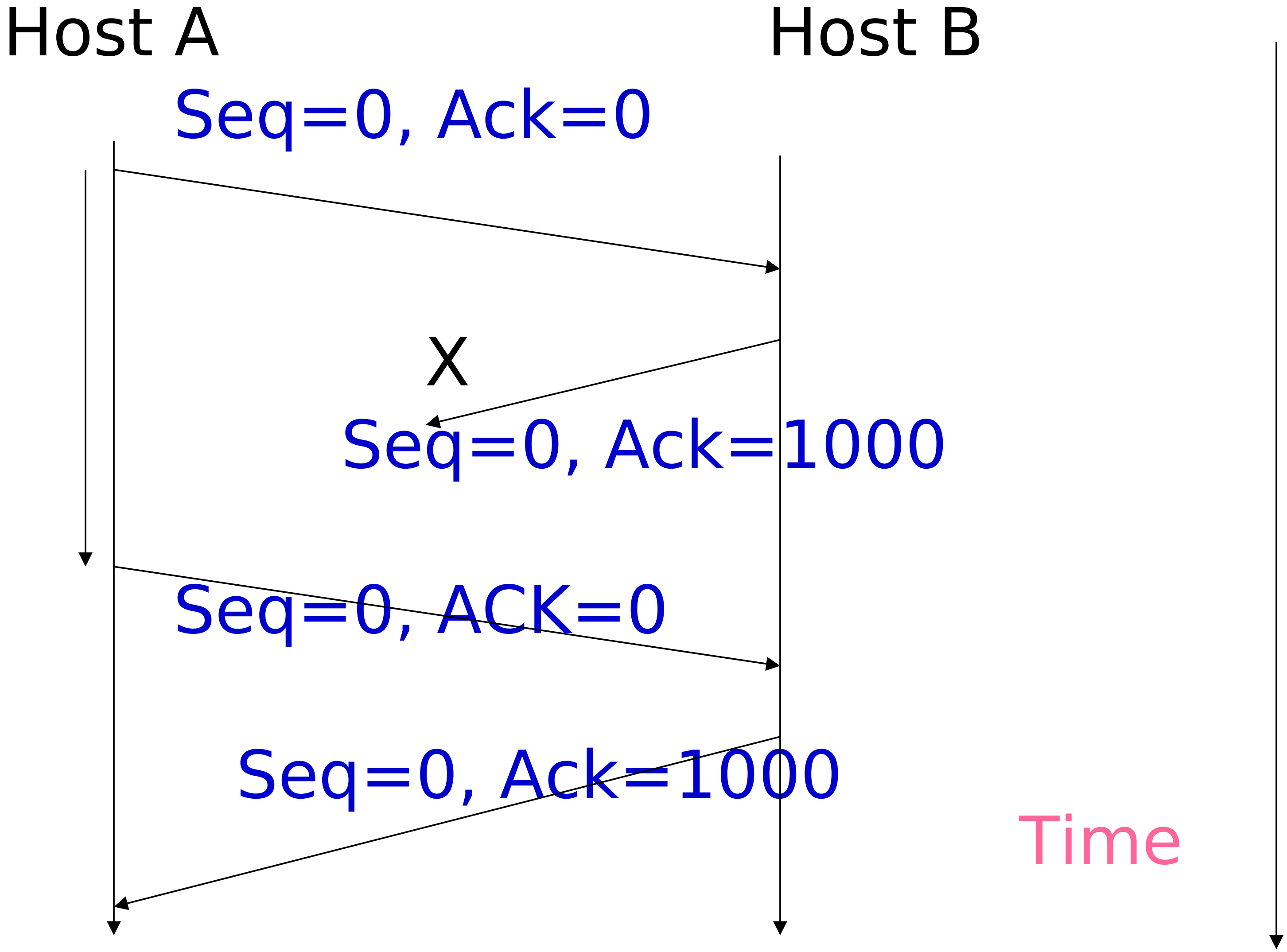
Data transfer problems

- Segments may arrive out of order
- Segments may
  - get lost (and also ACK’s)
  - Get damaged on its way
- Duplicate ACK’s
- Cumulative ACK’s - TCP maintains queue of segments transmitted but not acknowledged
- ACK Timeout

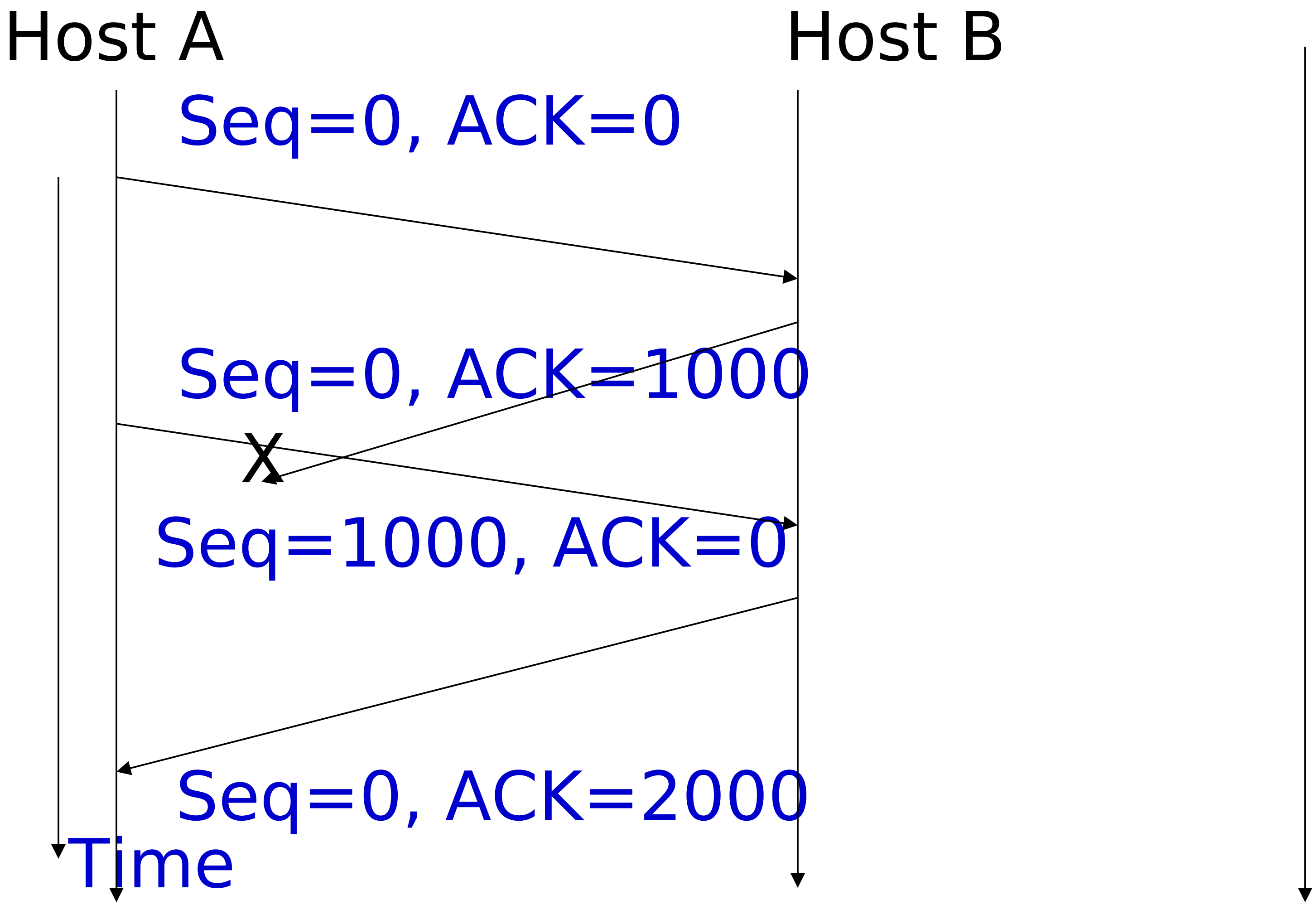
Orderly delivery

- Segments out of order
  - Discard out-of-order segments (Go-back-N)
  - Buffer and request for retransmission of missing segments – Selective repeat
- Time out waiting for ACK triggers re-transmission
- Damaged segments discarded and retransmission requested

Lost ACK


Host A
Host B
Seq=0, Ack=0
X
Seq=0, Ack=1000
Seq=0, ACK=0
Seq=0, Ack=1000
Time

Cumulative ACK's



Cumulative ACK's avoid retransmission of 1st Segment

Duplication Detection

- If ACK lost, segment is re-transmitted
- Receiver must recognize duplicates
- Duplicate received prior to closing connection
  - Receiver assumes ACK lost and ACKs duplicate
  - Sender must not get confused with multiple ACKs
  - Sequence number space large enough to not cycle within maximum life of segment
- Duplicate received after closing connection

Flow Control

- Sender maintains receive window
- Initial window sizes exchanged at connection setup
- Window size updated by each ACK
- Windows size of zero means sender cannot send more segments
- Sender must ensure that number of unacknowledged bytes is always less than window size
- Window size is dynamic (sliding window)

Connection Management

- Establishment and Termination
  - Allow each end to know the other exists
  - Negotiation of optional parameters
  - Triggers allocation of transport entity resources
- By mutual agreement

Connection Establishment (1)

- Two way handshake
  - A send SYN, B replies with SYN
  - Lost SYN handled by re-transmission
    - Can lead to duplicate SYNs
  - Ignore duplicate SYNs once connected

Connection Establishment (2)

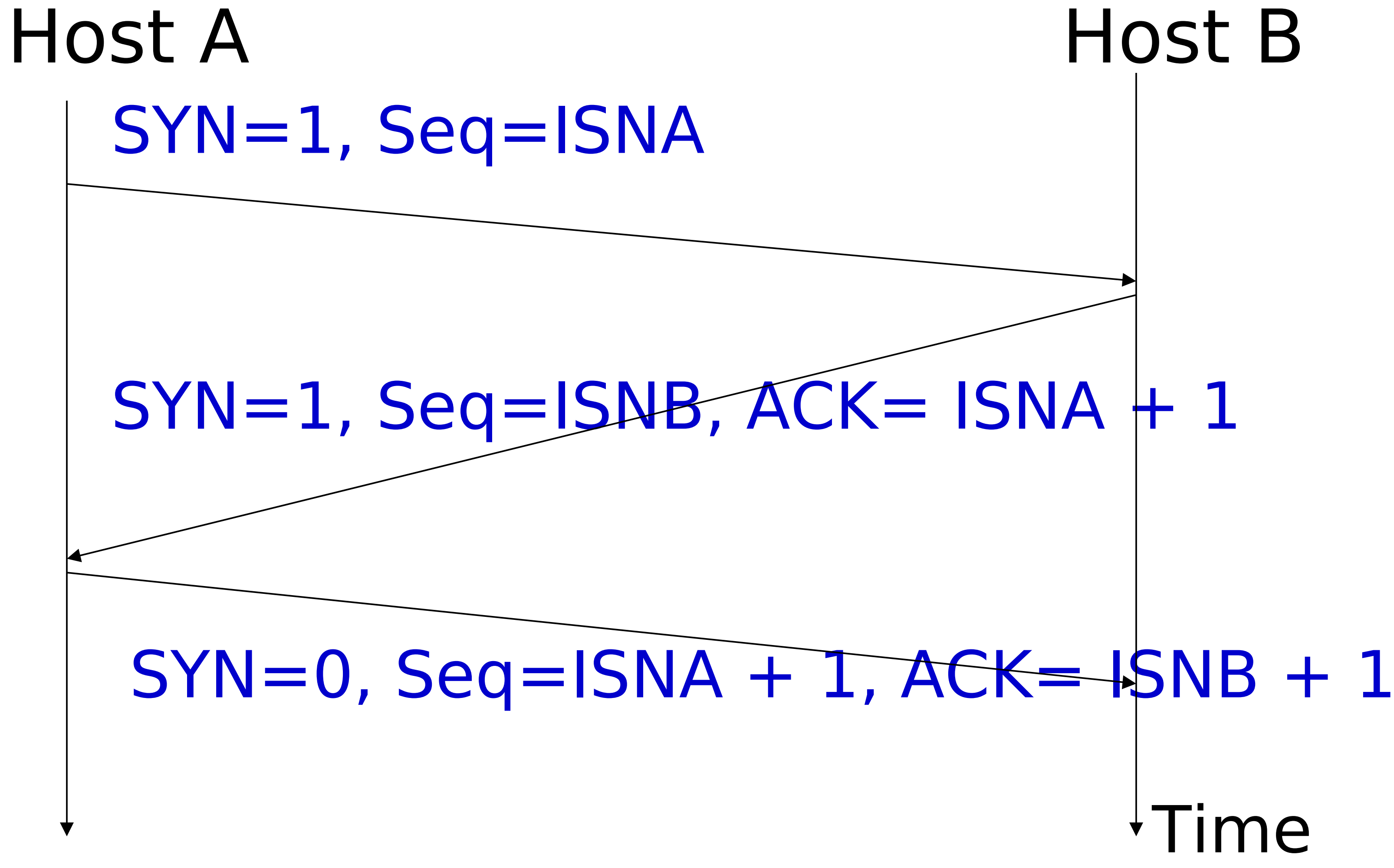
- Lost or delayed data segments can cause connection problems
  - Segment from old connections
  - Start segment numbers far removed from previous connection
    - Use SYN i
    - Need ACK to include i
    - Three Way Handshake

Connection Establishment (3)

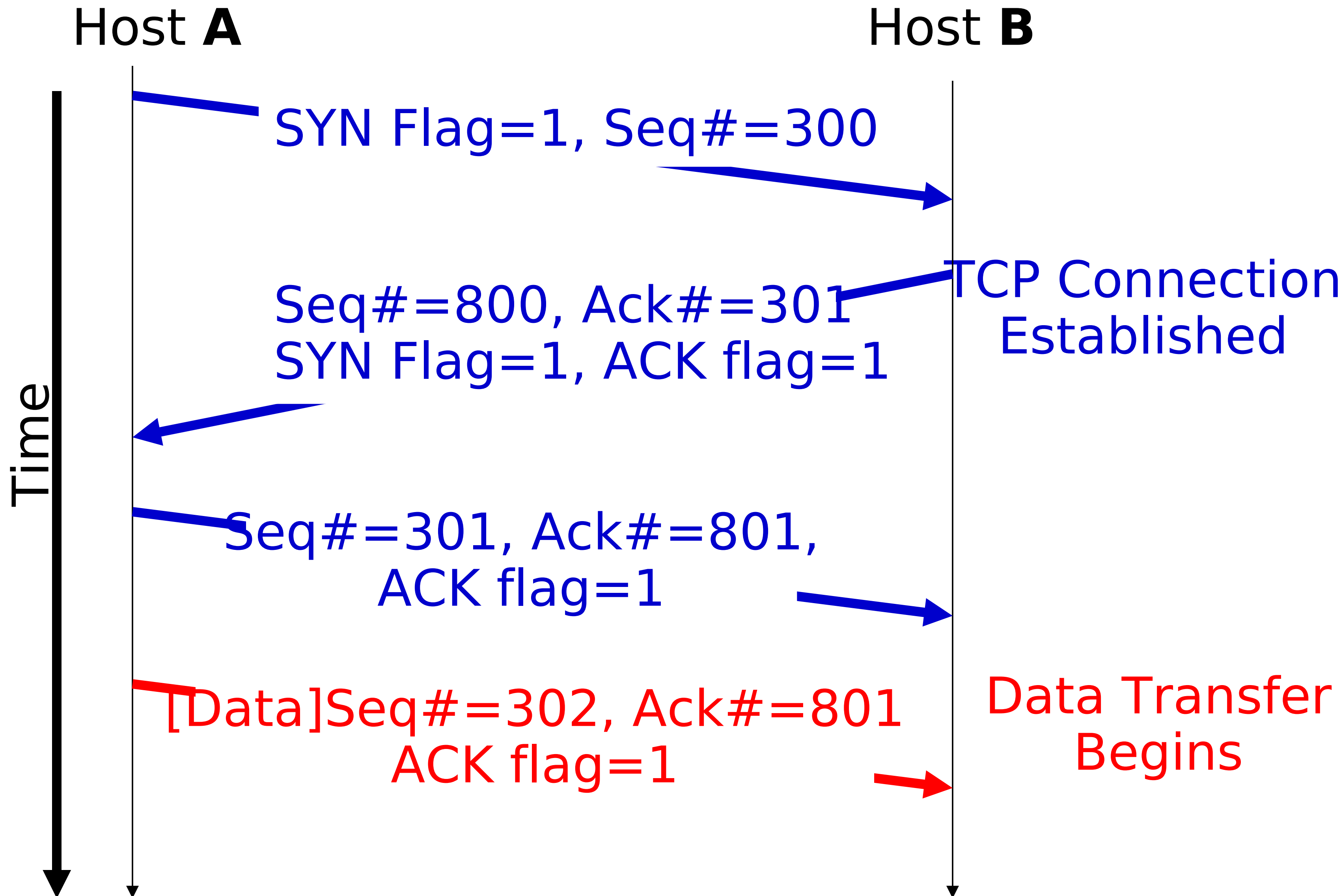
- Three way handshake
- Between pairs of ports
- One port can connect to multiple destinations

Connection Establishment (4)

TCP three-way handshake



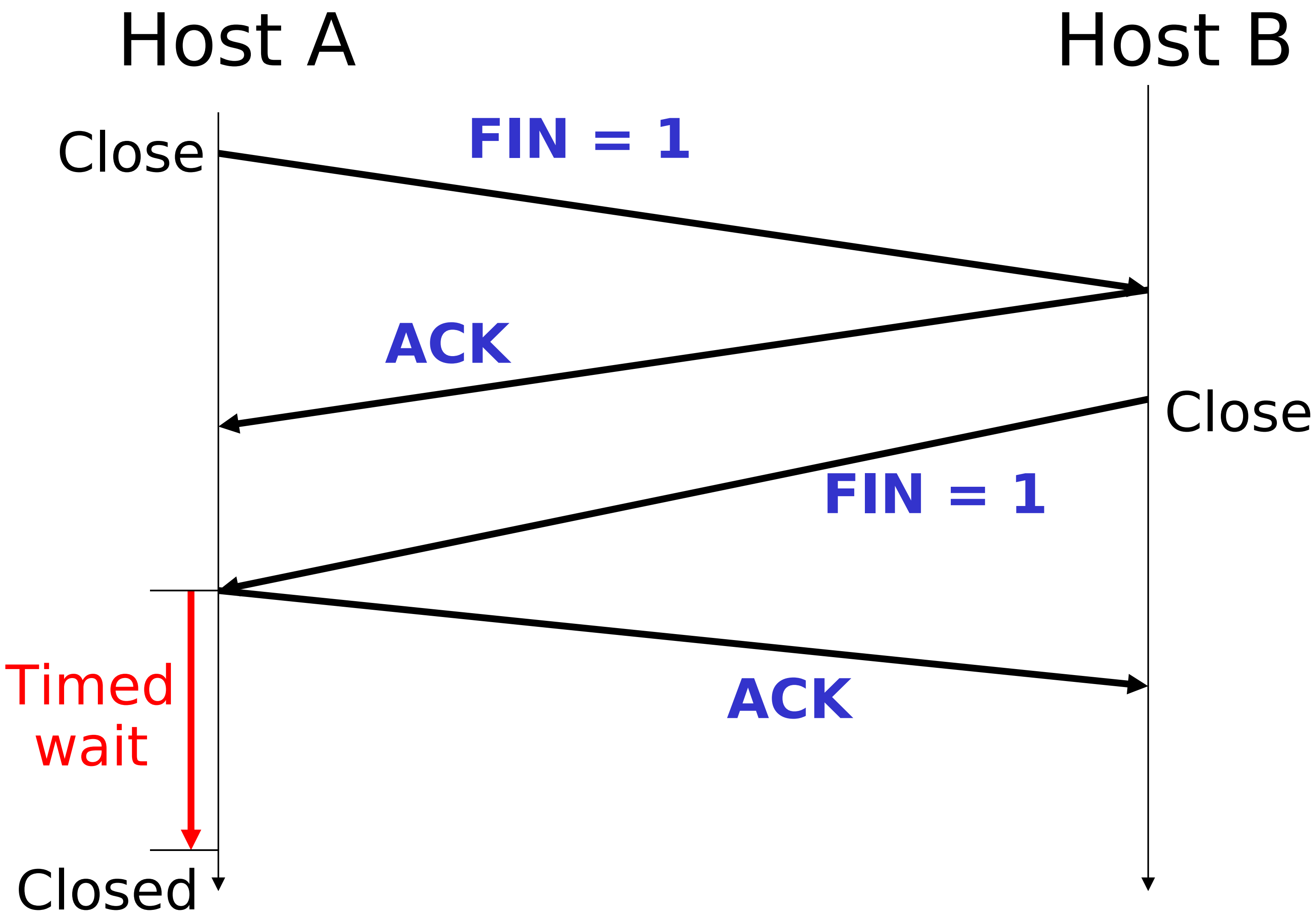
Connection Establishment (5)



Termination

- Either of the two processes in TCP connection can end the connection
- By mutual agreement
- Graceful termination
  - TCP user issues CLOSE primitive
  - Transport entity sets FIN flag on last segment sent
  - Close wait state must accept incoming data until FIN received
- Abrupt termination
  - TCP user issues ABORT primitive
  - Entity abandons all attempts to send or receive data
  - RST (reset the connection) segment transmitted

Closing a TCP Connection



Side Initiating Termination

- TS user issues Close request
- Transport entity sends FIN, requesting termination
- Connection placed in CLOSE WAIT state
  - Continue to accept data and deliver data to user
  - Not send any more data
- When FIN received, inform user and close connection

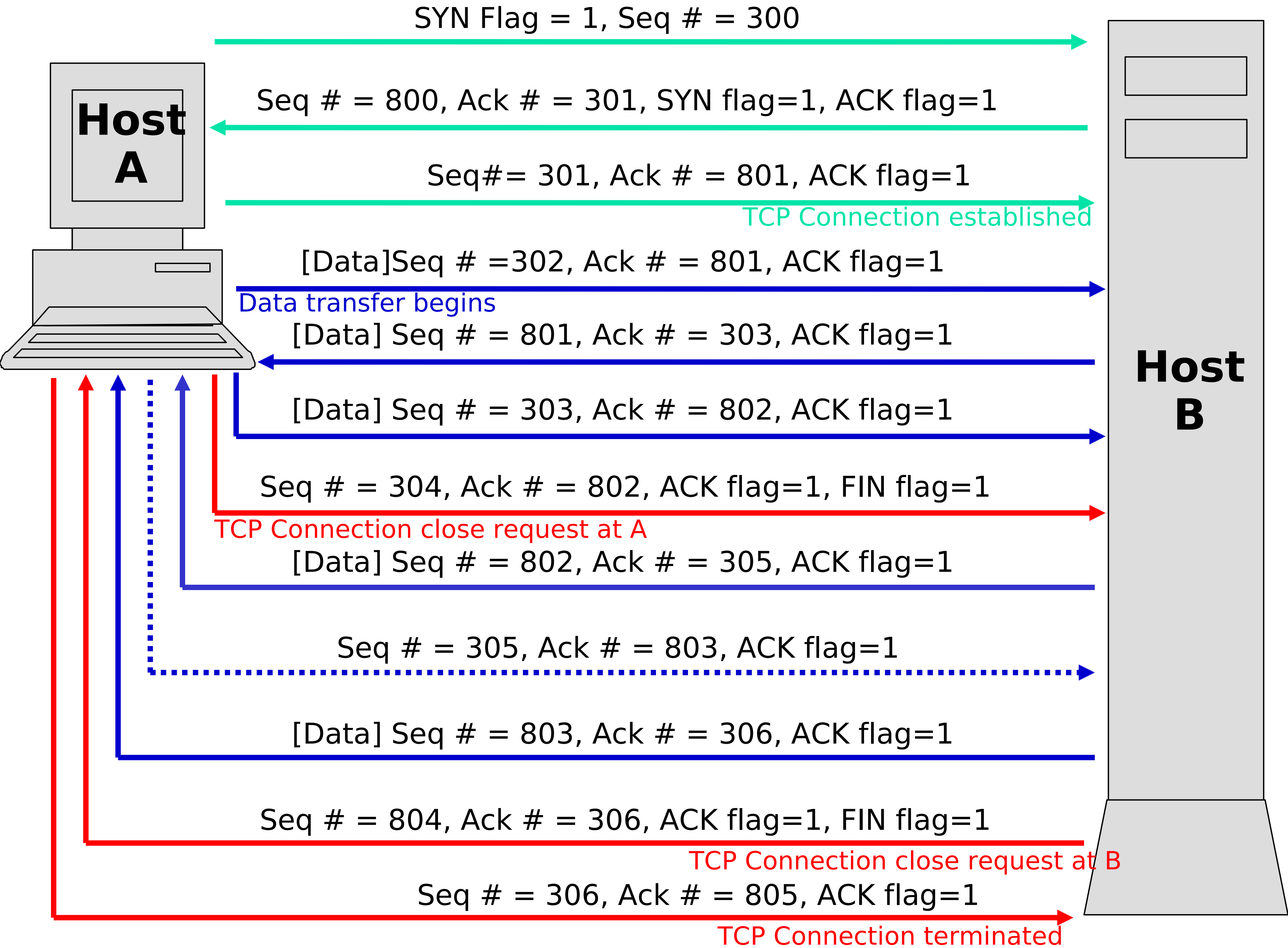
Side Not Initiating Termination

- FIN received
- Inform TS user Place connection in CLOSE WAIT state
  - Continue to accept data from TS user and transmit it
- TS user issues CLOSE primitive
- Transport entity sends FIN
- Connection closed

- All outstanding data is transmitted from both sides
- Both sides agree to terminate

Connection Termination

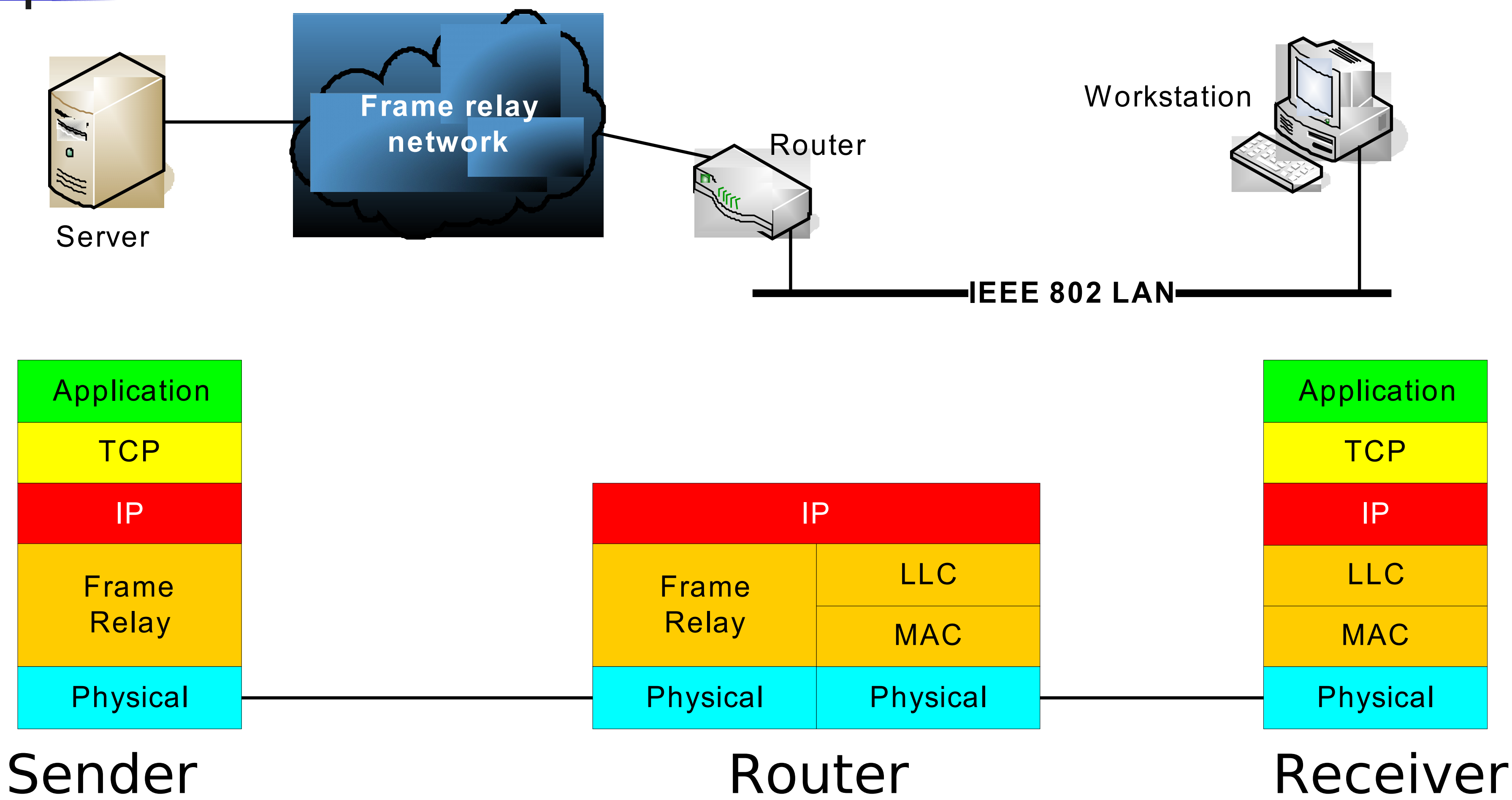
- Entity in CLOSE WAIT state sends last data segment, followed by FIN
- FIN arrives before last data segment
- Receiver accepts FIN
  - Closes connection
  - Loses last data segment
- Associate sequence number with FIN
- Receiver waits for all segments before FIN sequence number
- Then issues a FIN segment
  - Must explicitly ACK FIN


SYN Flag = 1, Seq # = 300
Host A
Seq # = 800, Ack # = 301, SYN flag=1, ACK flag=1
Seq#= 301, Ack # = 801, ACK flag=1
TCP Connection established
[Data]Seq # =302, Ack # = 801, ACK flag=1
Data transfer begins
[Data] Seq # = 801, Ack # = 303, ACK flag=1
Host B
[Data] Seq # = 303, Ack # = 802, ACK flag=1
Seq # = 304, Ack # = 802, ACK flag=1, FIN flag=1
TCP Connection close request at A
[Data] Seq # = 802, Ack # = 305, ACK flag=1
Seq # = 305, Ack # = 803, ACK flag=1
[Data] Seq # = 803, Ack # = 306, ACK flag=1
Seq # = 804, Ack # = 306, ACK flag=1, FIN flag=1
TCP Connection close request at B
Seq # = 306, Ack # = 805, ACK flag=1
TCP Connection terminated

Crash Recovery

- After restart all state info is lost
- Connection is half open
  - Side that did not crash still thinks it is connected
- Close connection using persistence timer
  - Wait for ACK for (time out) * (number of retries)
  - When expired, close connection and inform user
- Send RST i in response to any i segment arriving
- User must decide whether to reconnect
  - Problems with lost or duplicate data

Internetworking Example (TCP/IP)
Configuration for TCP/IP Example



Modified after Stallings & van Slyke, 1998, Business Data Communication

Operation of TCP/IP: Action at Sender

Modified after Stallings & van Slyke, 1998, Business Data Communication

1. Preparing the data. The application protocol prepares a block of data for transmission: for example an E-mail message (SMTP), a file (FTP), or a block of user input (TELNET)

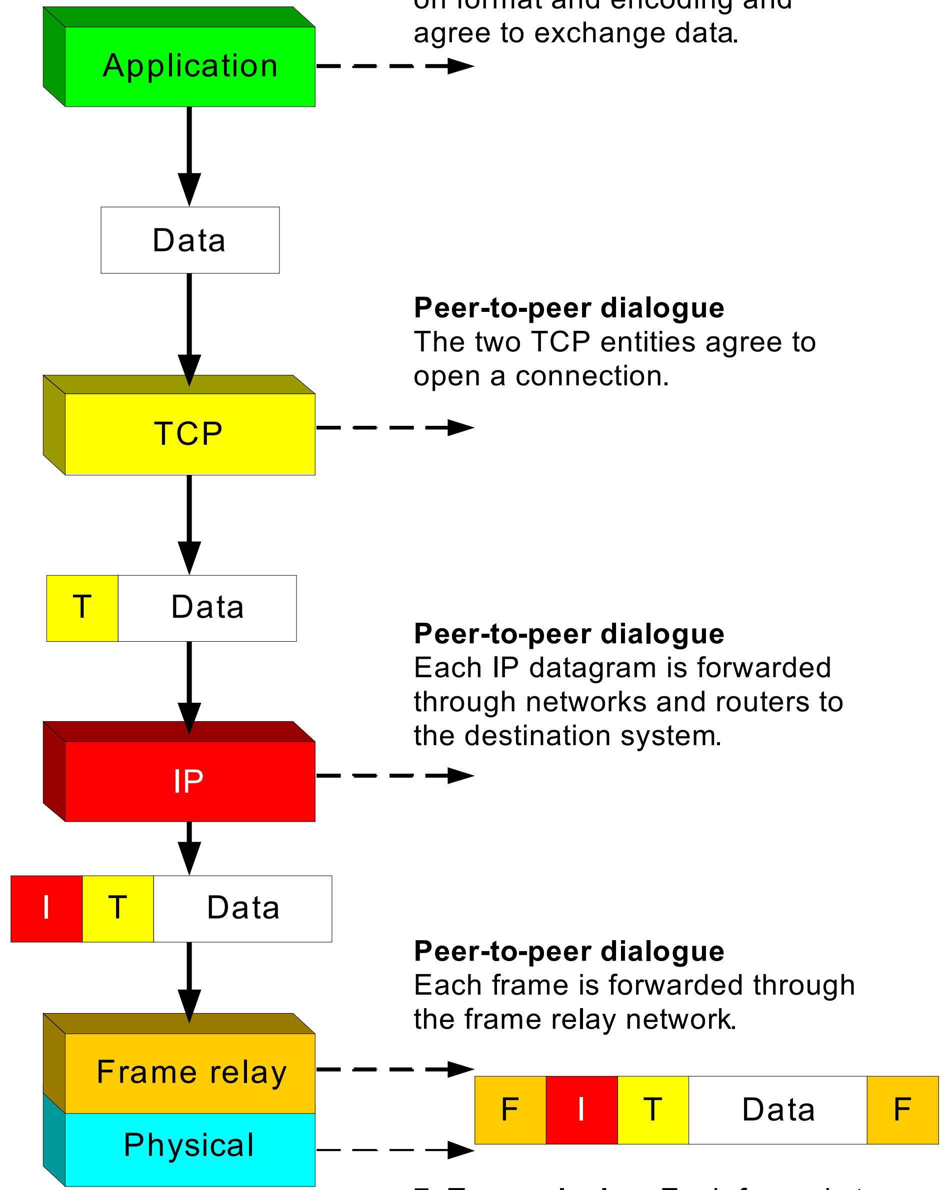
2. Using a common syntax. If necessary, the data are converted to a form expected b the destination. This may include a different character code, the use of encryption, and/ or compression.

3. Segmenting the data. TCP may break the data block into a number of segments, keeping track of their sequence. Each TCP segment includes a header containing a sequence number and a frame check sequence to detect errors.

4. Duplicating segments. A copy is made of each TCP segment, in case the loss or damage of a segment necessitates retransmission. When an acknowledgement is received from the other TCP entity, a segment is erased.

5. Fragmenting the segments. IP may break a TCP segment into a number of datagrams to meet size requirements of the intervening networks. Each datagram includes a header containing a destination address, a frame check sequence, and other control information.

6. Framing. A frame relay header and trailer is added to each IP datagram. The header contains a connection identifier and the trailer contains frame check sequence.



Peer-to-peer dialogue
Before data are sent, the sending and receiving applications agree on format and encoding and agree to exchange data.

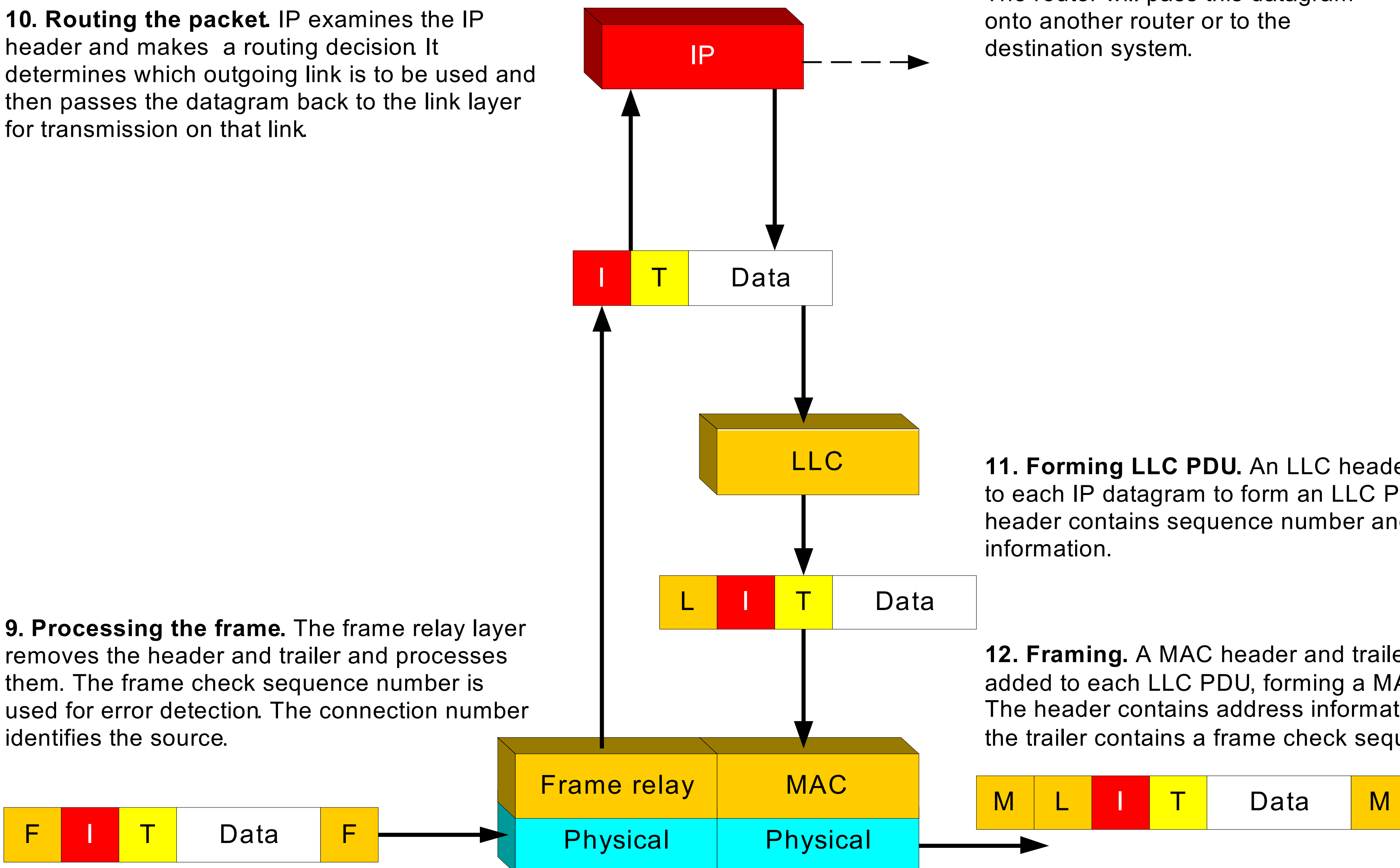
Peer-to-peer dialogue
The two TCP entities agree to open a connection.

Peer-to-peer dialogue
Each IP datagram is forwarded through networks and routers to the destination system.

Peer-to-peer dialogue
Each frame is forwarded through the frame relay network.

7. Transmission. Each frame is transmitted over the medium as a sequence of bits.

Operation of TCP/IP: Action at Router



8. Arriving at router. The incoming signal is received over the transmission medium and interpreted as a frame of bits.

9. Processing the frame. The frame relay layer removes the header and trailer and processes them. The frame check sequence number is used for error detection. The connection number identifies the source.

10. Routing the packet. IP examines the IP header and makes a routing decision. It determines which outgoing link is to be used and then passes the datagram back to the link layer for transmission on that link.

Peer-to-peer dialogue
The router will pass this datagram onto another router or to the destination system.

11. Forming LLC PDU. An LLC header is added to each IP datagram to form an LLC PDU. The header contains sequence number and address information.

12. Framing. A MAC header and trailer are added to each LLC PDU, forming a MAC frame. The header contains address information and the trailer contains a frame check sequence.

13. Transmission. Each frame is transmitted over the medium as a sequence of bits.

Operation of TCP/IP: Action at Receiver

20. Delivering the data. The application performs any needed transformations, including decompression and decryption, and directs the data to the appropriate file or other destination.

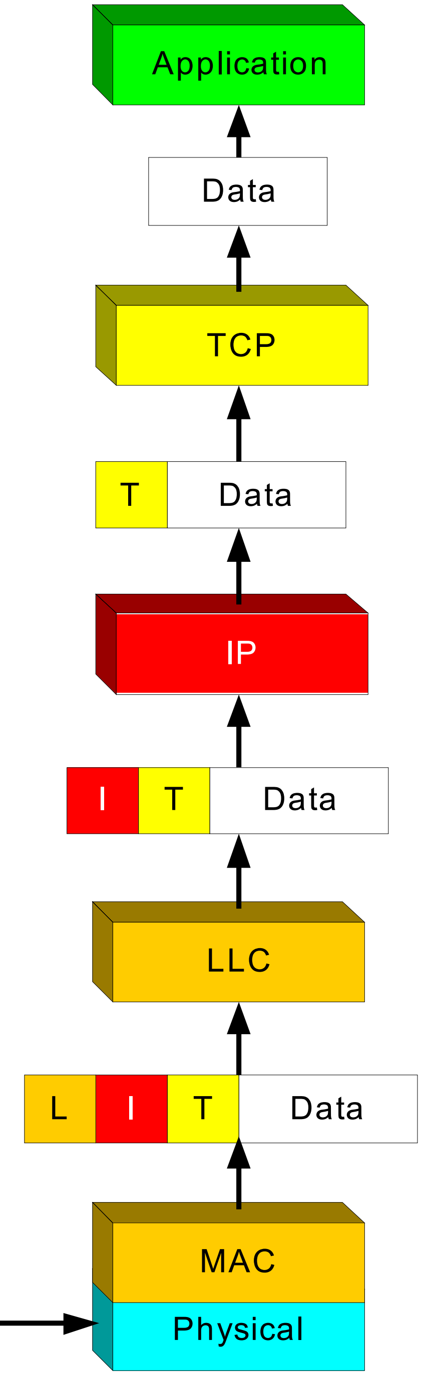
19. Reassembling user data. If TCP has broken the user data into multiple segments, these are reassembled and the block is passed up to the application.

18. Processing the TCP segment. TCP removes the header. It checks the frame check sequence and acknowledges if there is a match and discards for mismatch. Flow control is also performed.

17. Processing the IP datagram. IP removes the header. The frame check sequence and other control information are processed.

16. Processing the LLC PDU. The LLC layer removes the header and processes it. The sequence number is used for flow and error control.

15. Processing the frame. The MAC layer removes the header and trailer and processes them. The frame check sequence number is used for error detection.



14. Arriving at destination. The incoming signal is received over the transmission medium and interpreted as a frame of bits.