NSCOM01

Reliable Data Delivery

3rd Term– AY2022 – 2023

Instructor: Dr. Marnel Peradilla

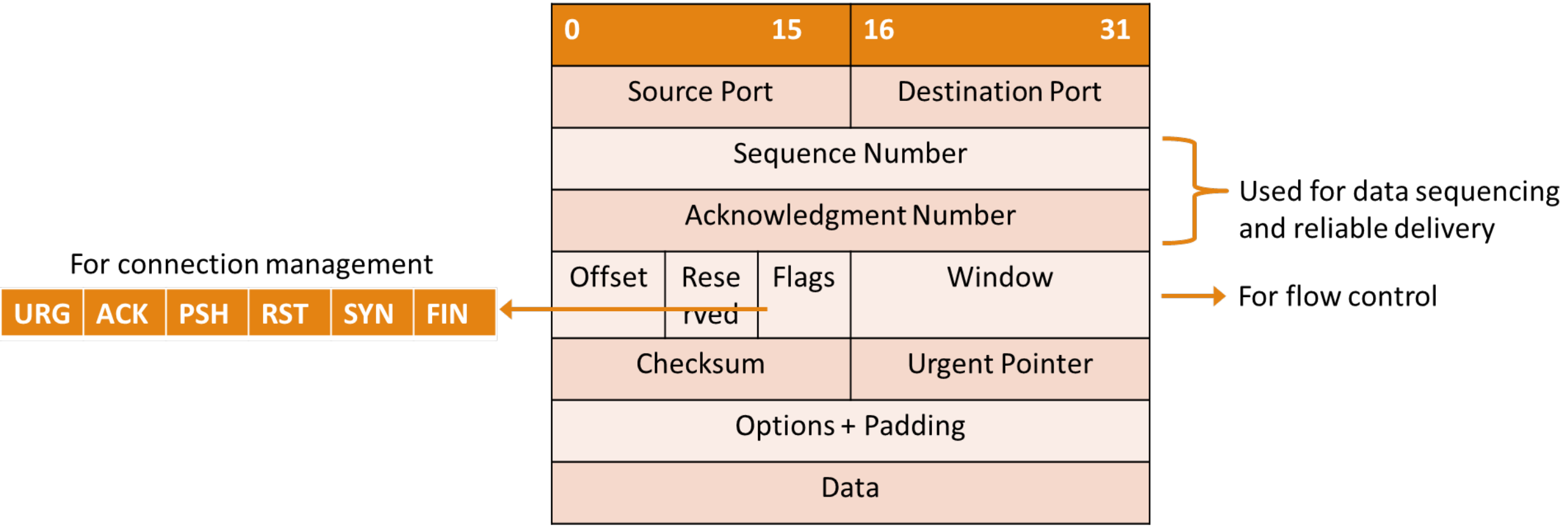
SPIRAL REVIEW: TRANSPORT SERVICES

- **Network applications depend heavily on the transport layer to provide end-to-end transfer of data between hosts**
- **Transport layer protocols provide data transfer through connection-oriented or connectionless services**
  - Connection-oriented service – requires establishment, maintenance and termination of logical connections between users to provide reliable data transport
  - Connectionless service – provides best-effort delivery service for data. Transports without need to establish any connection

TRANSMISSION CONTROL PROTOCOL

- **The Transmission Control Protocol (TCP) is a connection-oriented transport protocol used in TCP/IP networks**
- **Provides reliable communication between pairs of processes (TCP users) across a variety of reliable and unreliable networks**
- **Features:**
  - Stream-oriented – Data is sent in segments but handled as streams
  - Connection Oriented – Includes mechanisms to establish, track state and terminate a connection between 2 hosts
  - Guaranteed delivery – packets are acknowledged by receiving hosts
  - Flow control- Data transmission adapts to network conditions and host capability
  - Ordered delivery – Segments may arrive out of-order but are reassembled in the correct sequence

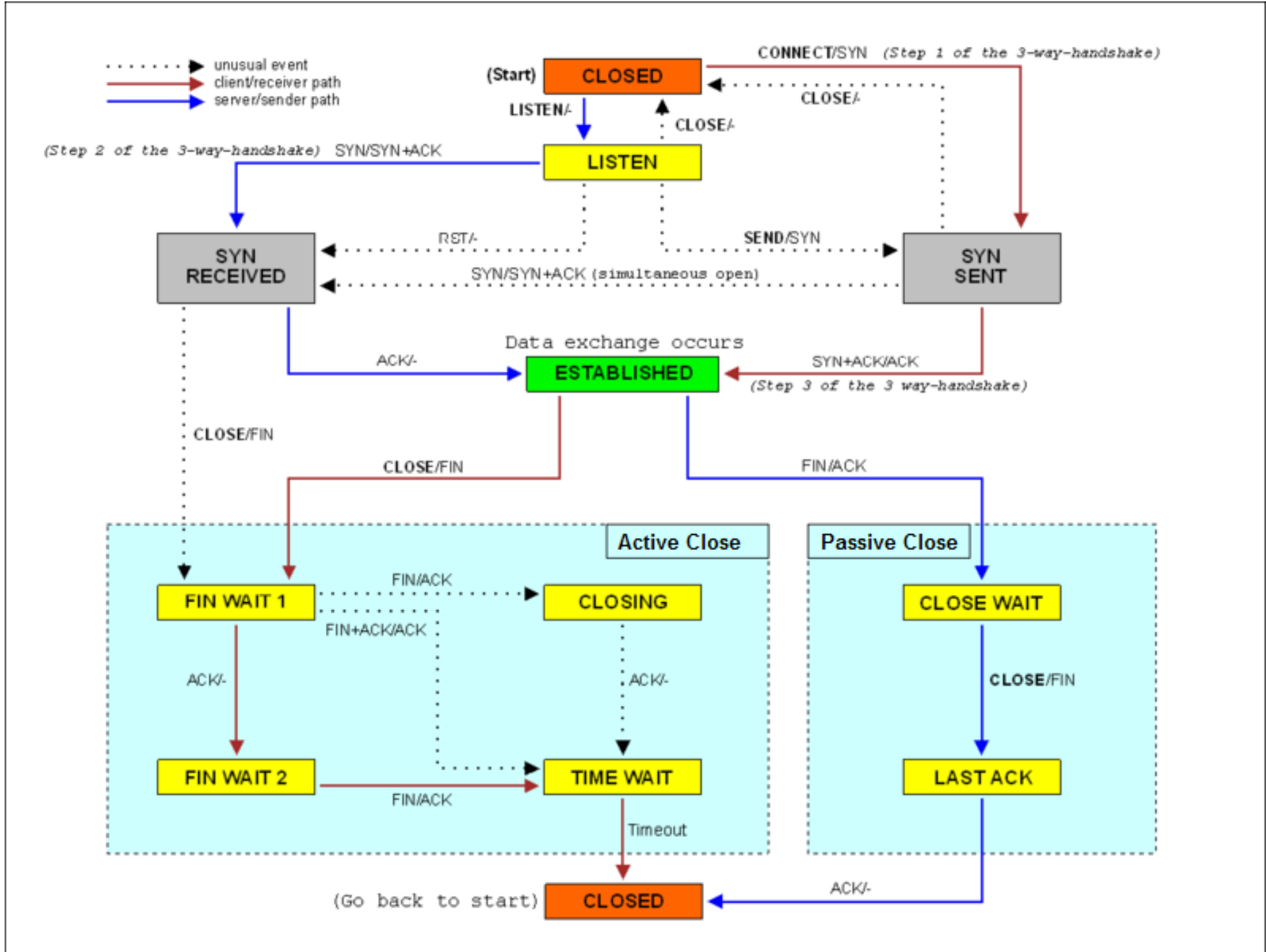
TCP HEADER


0
15
16
31
Source Port
Destination Port
Sequence Number
Acknowledgment Number
Offset
Rese
rved
Flags
Window
Checksum
Urgent Pointer
Options + Padding
Data
Used for data sequencing and reliable delivery
For flow control
For connection management
URG
ACK
PSH
RST
SYN
FIN

TCP CONNECTION ESTABLISHMENT AND TERMINATION

- **In a reliable transport protocol, connection establishment serves 3 purposes:**
  - It allows each end to assure that the other exists.
  - It allows negotiation of optional parameters (e.g., maximum segment size, maximum window size, quality of service).
  - It triggers allocation of transport entity resources (e.g., buffer space, entry in connection table)
- **TCP hosts begin in the CLOSED state and may begin a connection using:**
  - PASSIVE Open – host opens port and listens for incoming connections
  - ACTIVE Open – Host opens port and attempts a connection to a listening port

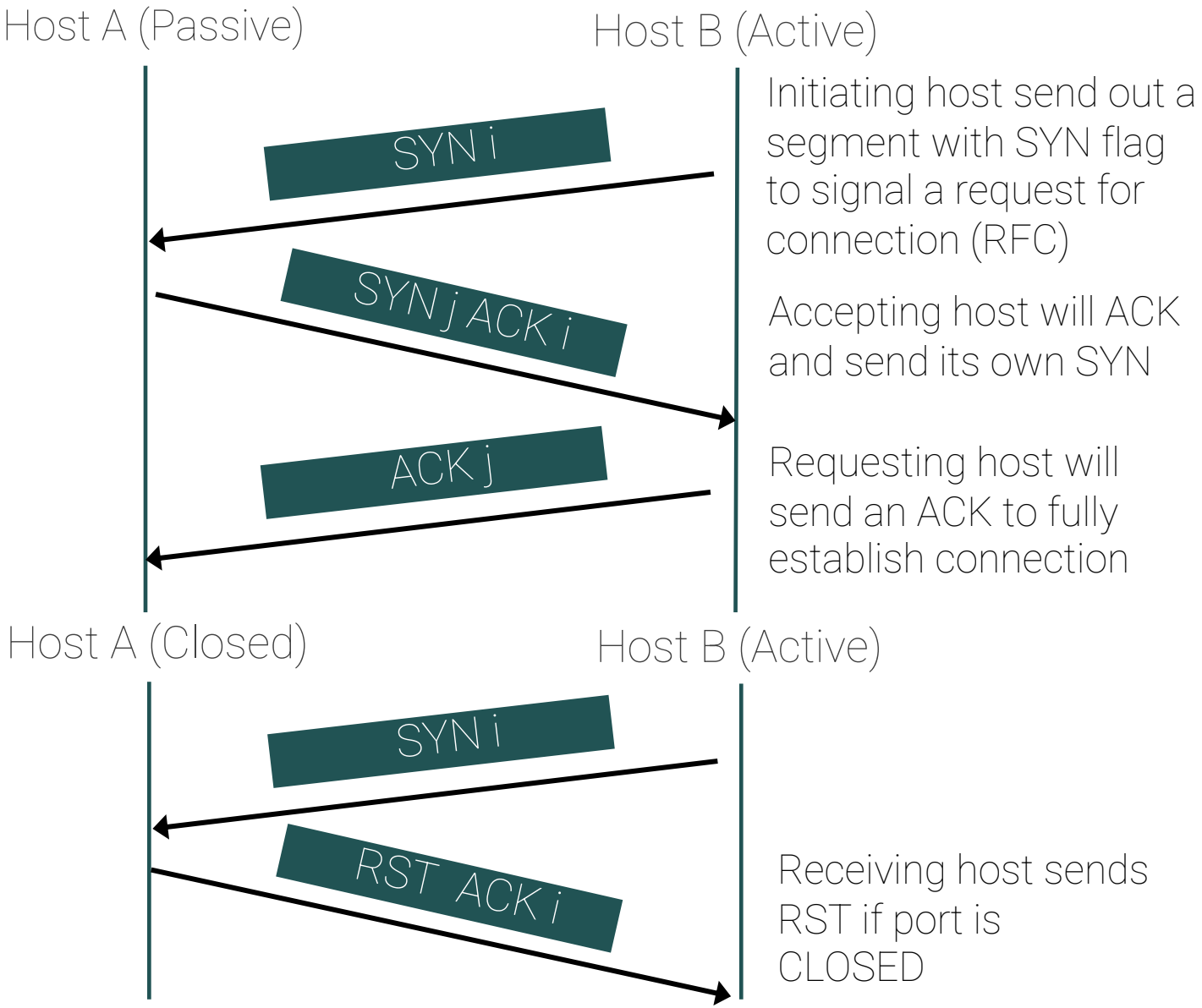
TCP STATE DIAGRAM



22 June 2023

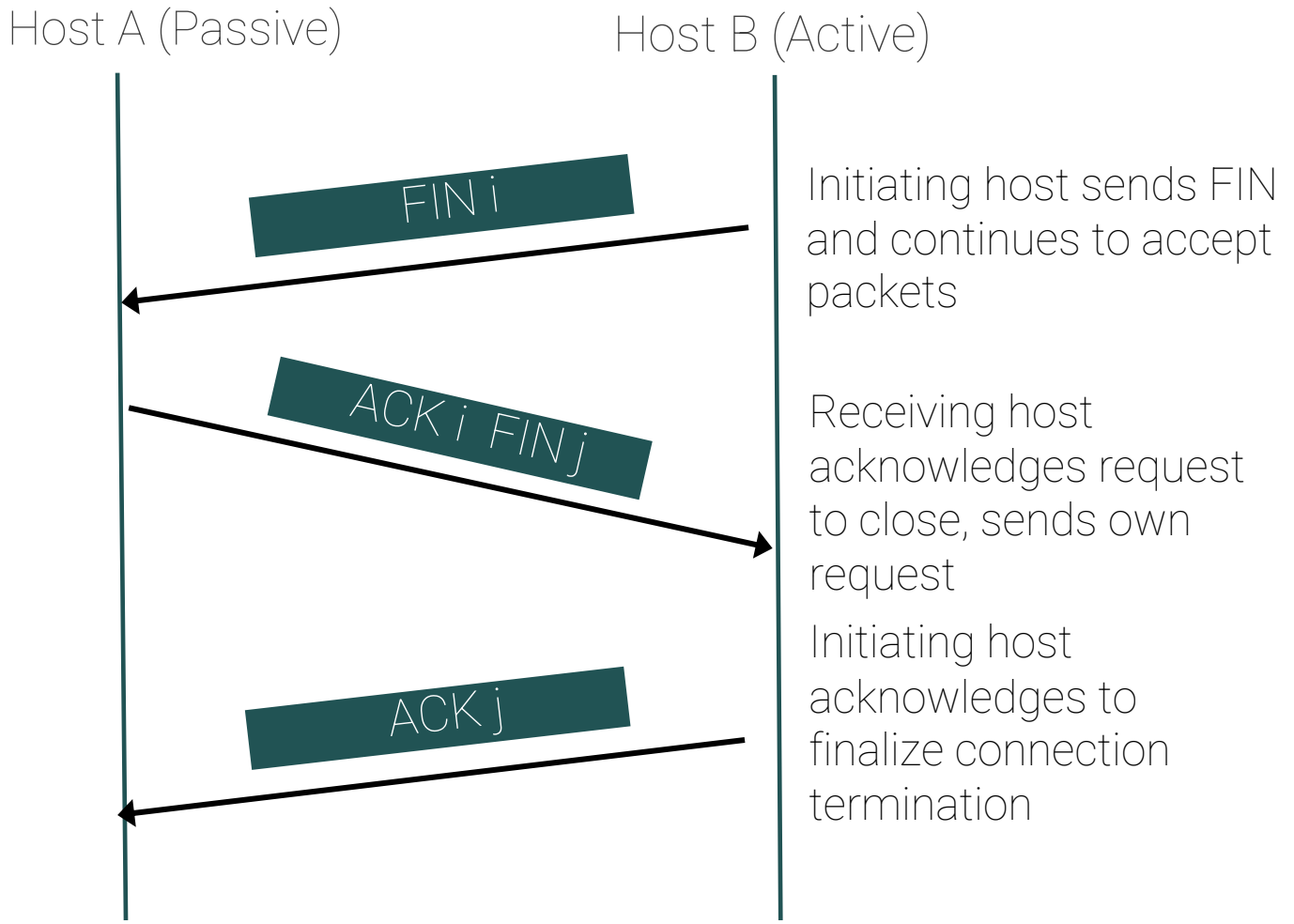
3-Way Handshake

- TCP uses the three-way handshake to establish a connection between hosts
  - A connection is uniquely identified by the source and destination ports.
  - Only 1 TCP connection at a time can be established between a unique pair of ports
- An attempt to request for a connection to a CLOSED port will receive an RST



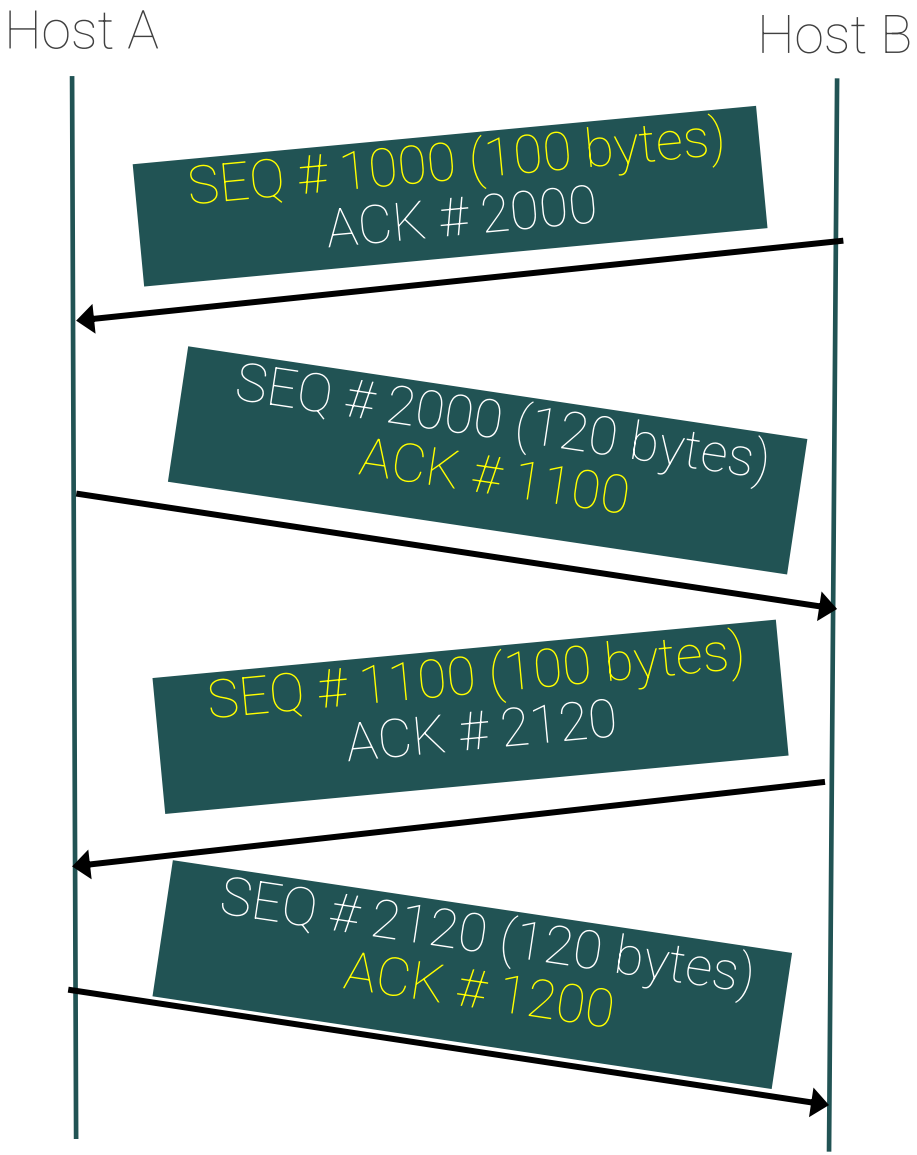
Connection Closing

- Connection termination is accomplished using mutual agreement of communicating hosts
  - ACTIVE Close – host initiates the termination of connection
  - PASSIVE Close – host receives and agrees to the request to terminate
- TCP allows for either abrupt or graceful termination.
- To achieve a graceful connection termination, both sides of the of the connection must send the FIN flag which must be ACKed by the opposite side



Reliable and Ordered Delivery

- TCP uses sequence numbers to indicate segment order and acknowledgment numbers to confirm receipt of data
- Communicating hosts each begin with a random starting sequence number
- Sequence Numbers (SEQ#)
  - Represents byte stream index of first byte in segment's data relative to the starting sequence number
  - "I am sending you a data chunk starting from byte # ____"
- Acknowledgements (ACK#)
  - Sequence number of next byte expected from other opposite host
  - "Got it. Send me the next data chunk starting from byte # ___"
- When data segments arrive over a TCP connection, TCP places the data in a receive buffer for delivery to the user



RELIABLE AND ORDERED DELIVERY

- **Depending on implementation, TCP may use either of 2 options for its data acknowledgment policy:**
  - Immediate
    - When data are accepted, immediately send an empty segment containing the appropriate acknowledgment number
    - Simple but more data to transmit since extra empty segments may need to be sent just for ACKs
  - Cumulative
    - When data segments are accepted, record the need for acknowledgment, but wait for an outbound segment with data to piggyback the acknowledgment
    - Fewer unnecessary transmissions but more complex processing to track pending ACKs

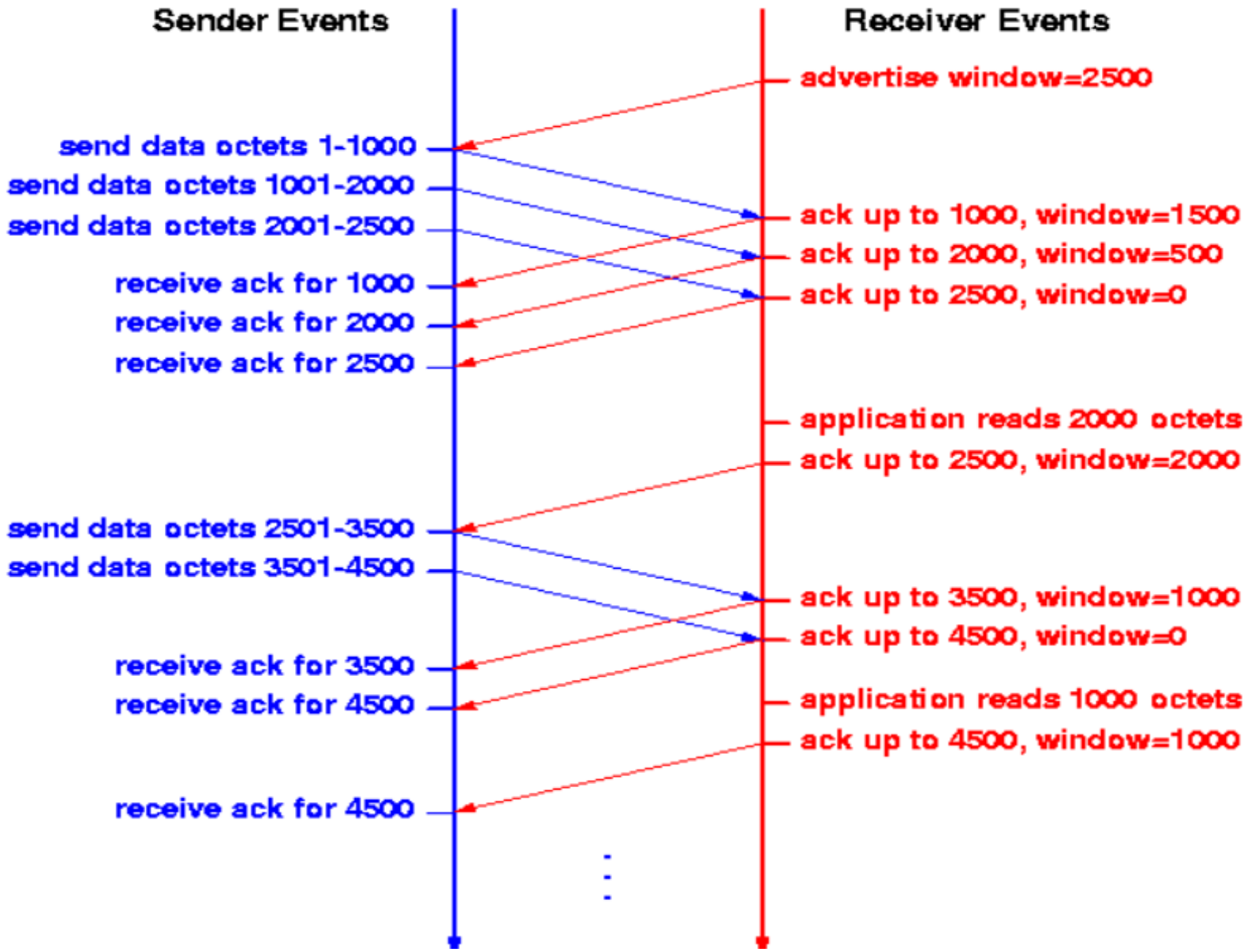
RELIABLE AND ORDERED DELIVERY

- **TCP hosts use timers to wait for data to be acknowledged. If timer expires, data needs to be retransmitted. The following options may be used for its retransmit policy:**
  - First-only
    - 1 retransmission timer for the entire segment queue.
    - If the timer expires, retransmit the segment at the front of the queue then reset timer
  - Batch
    - 1 retransmission timer for the entire queue
    - If the timer expires, retransmit all segments in the queue then reset timer
  - Individual
    - 1 timer for each segment in the queue
    - If any timer expires, retransmit the corresponding segment then reset the timer

FLOW CONTROL

- **The transport layer may need to do flow control due to the receiving user or host not being able to keep up with processing of incoming segments, leading to the receiving buffer filling up**
- **TCP uses a sliding window with credit allocation scheme to control the amount of data that a sending host may transmit to a receiver at any time**
  - Acknowledgment number is used to signal receipt of all data with stream index up to ACK# - 1
  - Window size is used to signal to the sender how much more data a receiver can currently accept
- **As a host continues to receive and acknowledge segments it can**
  - Decrease window size to control flow and prevent overflowing its buffer while waiting for received data to be processed
  - Increase window size to increase flow and accept more data in the buffer once space frees up
- **TCP implementations commonly use a “slow start” technique – Window size starts small and is gradually increased as more data is successfully received**

FLOW CONTROL



EXPEDITED DELIVERY

- **Some data submitted to the transport service may supersede data that was previously submitted**
- **TCP offers expedited delivery services through its data stream push and urgent data signaling capability**
- **Data Stream Push**
  - Provides a means to transmit all outstanding data up to and including those labeled with a PSH
    flag without further delay
  - On the receiving end, TCP will deliver these data to the application layer immediately without waiting for the receive buffer to fill
- **Urgent data signaling**
  - Provides a means of informing the destination device that significant or "urgent" data is in the upcoming data stream for it to take the appropriate handling action
  - Uses the URG flag and indicates the location of urgent data in the payload using the urgent pointer field

CONSIDERATIONS

- **As a stream and connection-oriented protocol, the following need to be considered when using TCP:**
  - More overhead – Additional bandwidth consumption and latency due to the additional mechanisms to support reliability and connection handling
  - Unicast only – TCP supports unicast transmission only because of the need to establish connections. If data needs to be sent to multiple hosts, then 1 copy of the segment must be sent to each destination
  - Buffering – Data is treated as a continuous stream. Original segment boundaries are not preserved when retrieving data from a buffer

TCP-BASED NETWORK APPLICATION PROTOCOLS

- TELNET and Secure Shell (SSH)
- HTTP
- SMTP
- POP
- FTP
- and many more…

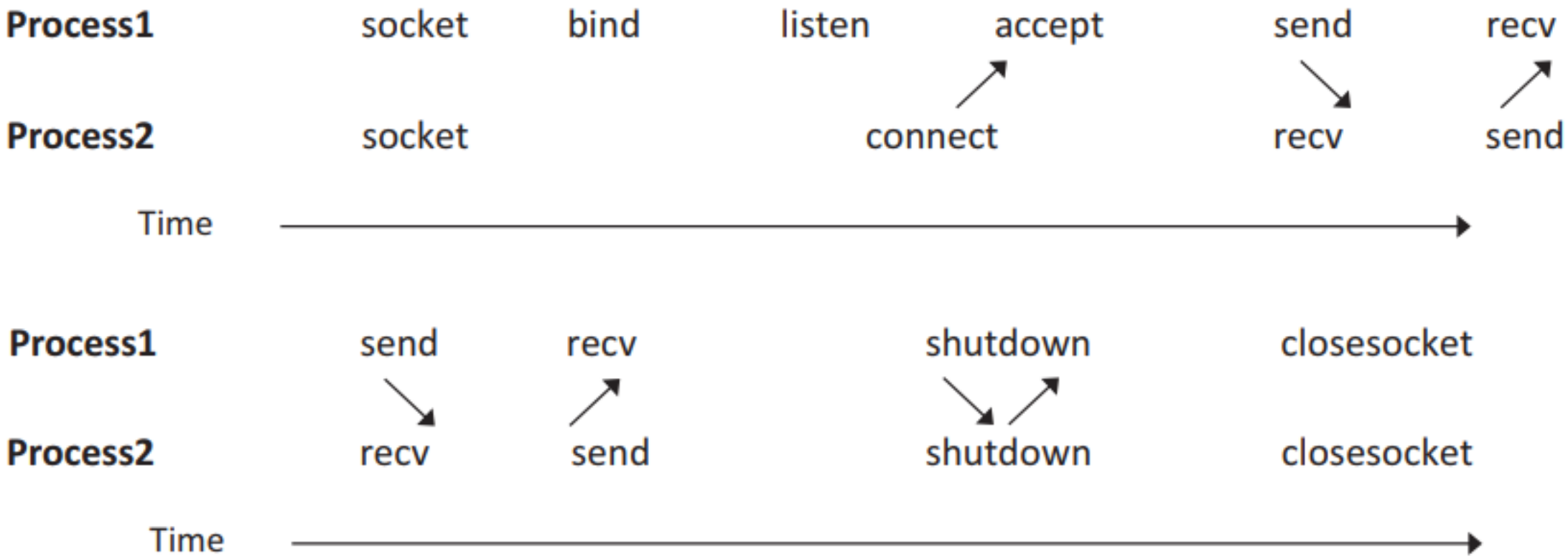
TCP SOCKETS PRIMITIVES

1. **Socket** primitive is used to create a socket which may be set to blocking or non-blocking IO mode
2. **Bind** primitive is used to map a process to a port
3. **Listen** primitive is used to perform a passive open in order for the port to be receptive to incoming connections
4. **Connect** primitive is used to perform an active open by initiating a connection request to a listening host
5. **Accept** primitive is used to link a connection with a handling application to create a dedicated socket and free up the listening socket for succeeding connection requests.

TCP SOCKETS PRIMITIVES

6. **Send** primitive is used to send data to another process.
7. **Recv** primitive is used to retrieve data from the receive buffer.
8. **Shutdown** primitive is used to initiate connection termination
9. **Closesocket** primitive is used to close the socket.

TCP SOCKETS PRIMITIVES



MESSAGE FROM DPO

"The information and data contained in the online learning modules, such as the content, audio/visual materials or artwork are considered the intellectual property of the author and shall be treated in accordance with the IP Policies of DLSU. They are considered confidential information and intended only for the person/s or entities to which they are addressed. They are not allowed to be disclosed, distributed, lifted, or in any way reproduced without the written consent of the author/owner of the intellectual property."