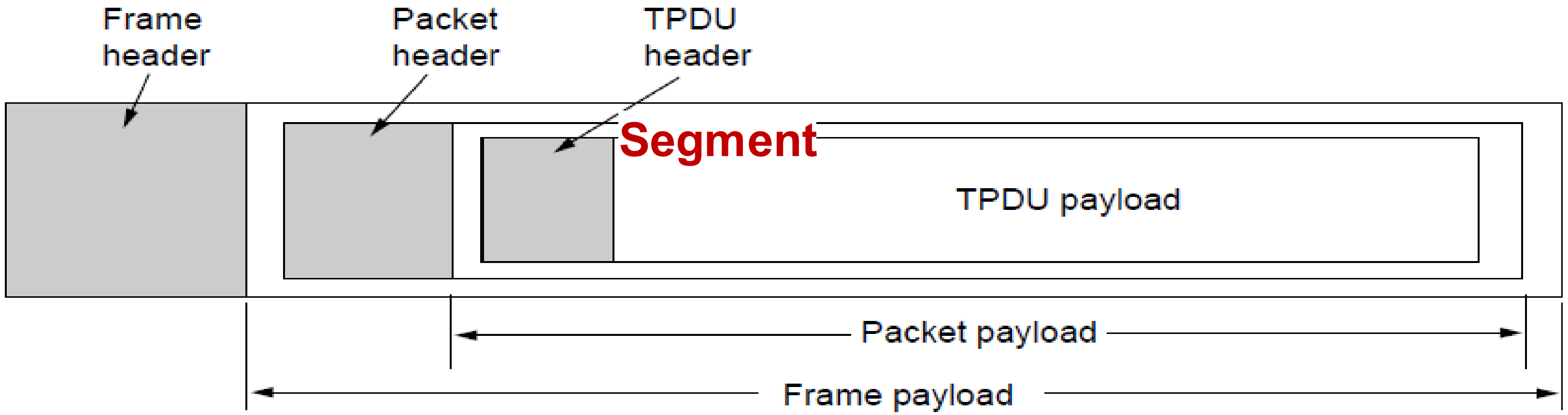
Transport Layer Contd

Internet Technologies
COMP90007

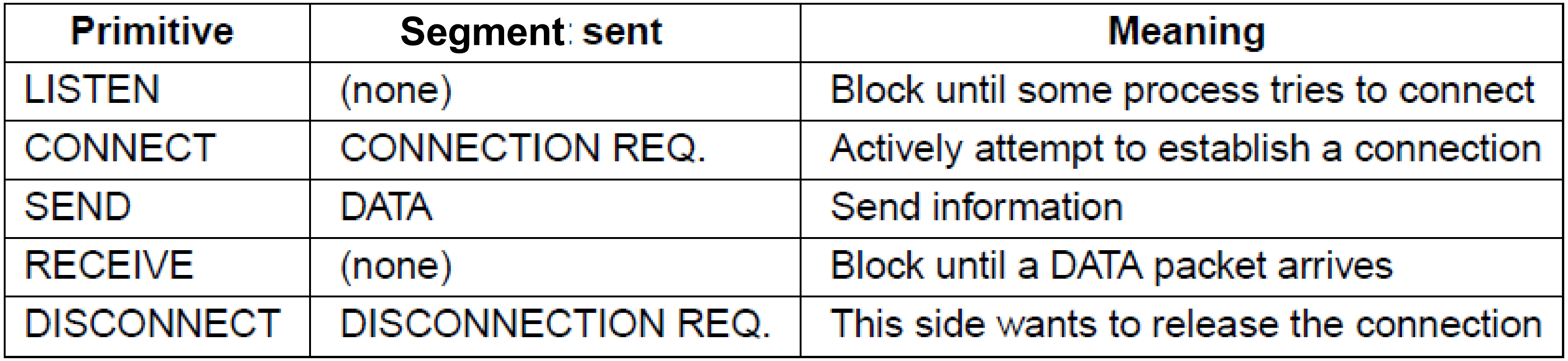
Transport Layer Messages

- Abstract representation of messages sent to and from transport entities
  - Transport Protocol Data Unit (TPDU): Segment
- Encapsulation of transport layer units to network layer units (to frames in datalink layer units)



Transport Primitives with Segments

- Primitives that applications might call at transport layer for a simple connection-oriented service:
  - Server executes LISTEN
  - Client executes CONNECT
    - Sends CONNECTION REQUEST TPDU to Server
    - Receives CONNECTION ACCEPTED TPDU at Client
  - Data exchanged using SEND and RECEIVE
  - Either party executes DISCONNECT



Primitive	Segment sent	Meaning
LISTEN	(none)	Block until some process tries to connect
CONNECT	CONNECTION REQ.	Actively attempt to establish a connection
SEND	DATA	Send information
RECEIVE	(none)	Block until a DATA packet arrives
DISCONNECT	DISCONNECTION REQ.	This side wants to release the connection

Your First Network (Pseudo) Code

```
Socket A_Socket = createSocket("TCP");

connect(A_Socket, 128.255.16.0, 80);

send(A_socket, “My first message!”);

disconnect(A_socket);
```

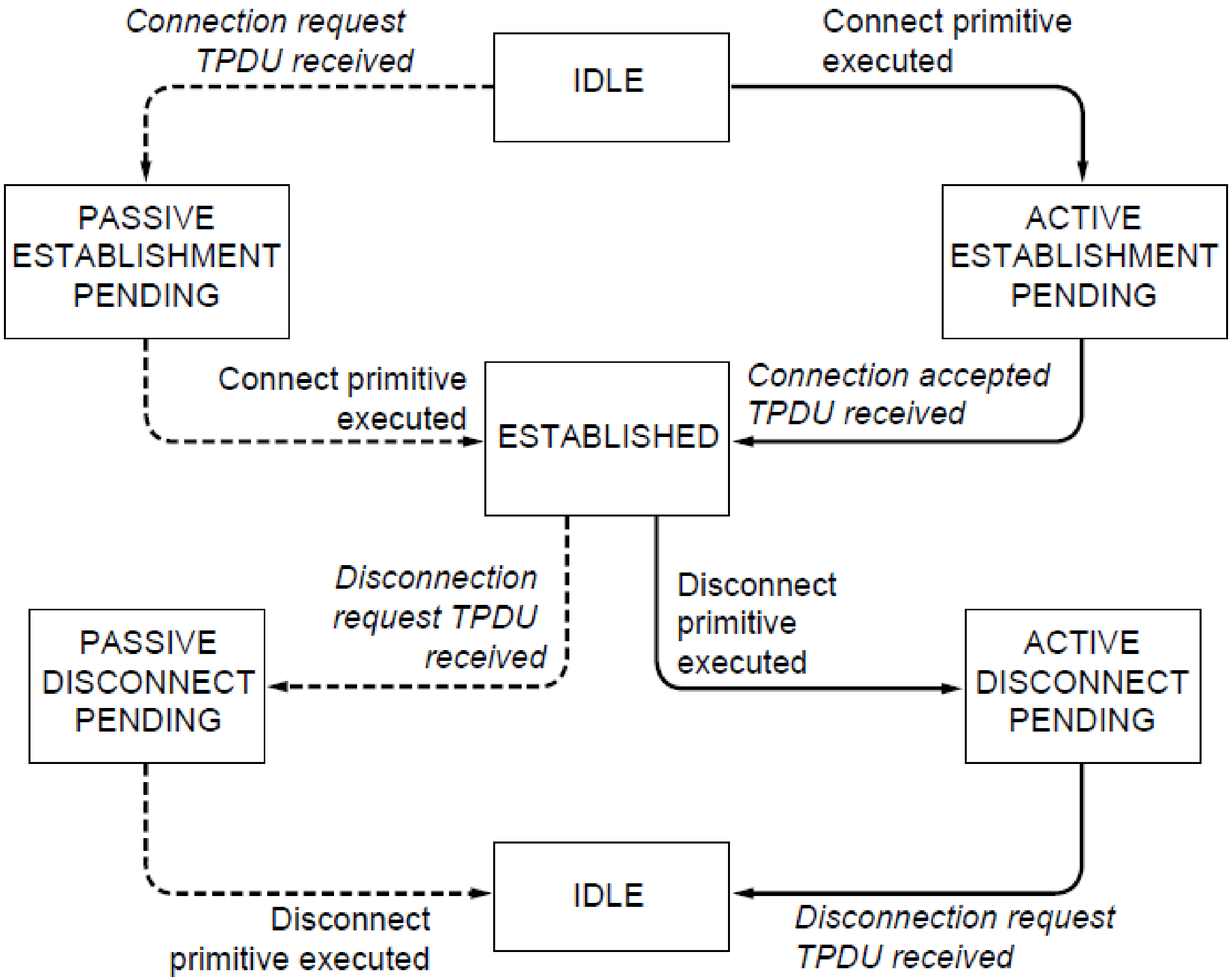
… there is also a server component for this client that runs on another host that listens etc…

Elements of Transport Protocols

- Connection establishment
- Connection release
- Addressing

Simple Connection Illustrated

- Solid lines (right) show client state sequence
- Dashed lines (left) show server state sequence
- Transitions in italics are due to segment arrivals



Connection request TPDU received
IDLE
Connect primitive executed
PASSIVE ESTABLISHMENT PENDING
ACTIVE ESTABLISHMENT PENDING
Connect primitive executed
ESTABLISHED
Connection accepted TPDU received
Disconnection request TPDU received
Disconnect primitive executed
PASSIVE DISCONNECT PENDING
ACTIVE DISCONNECT PENDING
IDLE
Disconnect primitive executed
Disconnection request TPDU received

Connection Establishment in Real World

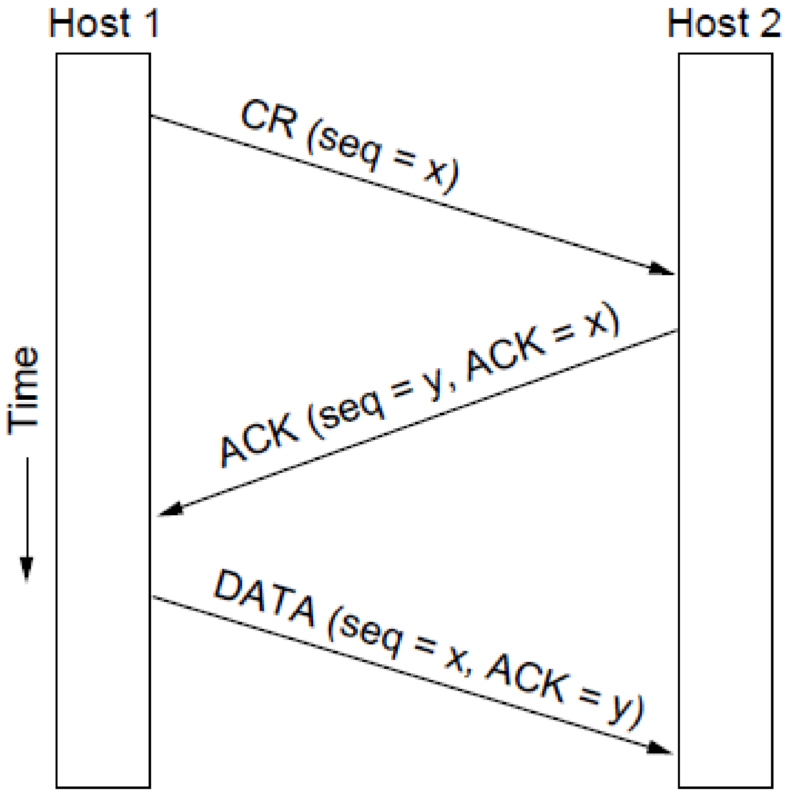
- Networks can lose, store and duplicate packets and thus connection establishment can be complicated
  - congested networks may delay acknowledgements
  - incurring repeated multiple transmissions
  - any of which may not arrive at all or out of sequence – delayed duplicates
  - applications degenerate with such congestion (eg. imagine duplication of bank withdrawals)

Reliable Connection Establishment

- Key challenge is to ensure reliability even though packets may be lost, corrupted, delayed, and duplicated
  - Don’t treat an old or duplicate packet as new
  - Use repeat requests and checksums for loss/corruption
- Approach:
  - Don’t reuse sequence numbers within maximum segment lifetime
  - Use a sequence number space large enough that it will not wrap, even when sending at full rate
  - Three-way handshake for establishing connection

Three Way Handshake

- Three-way handshake used for initial packet
  - Both hosts contribute fresh sequence(seq) numbers
  - CR = Connect Request

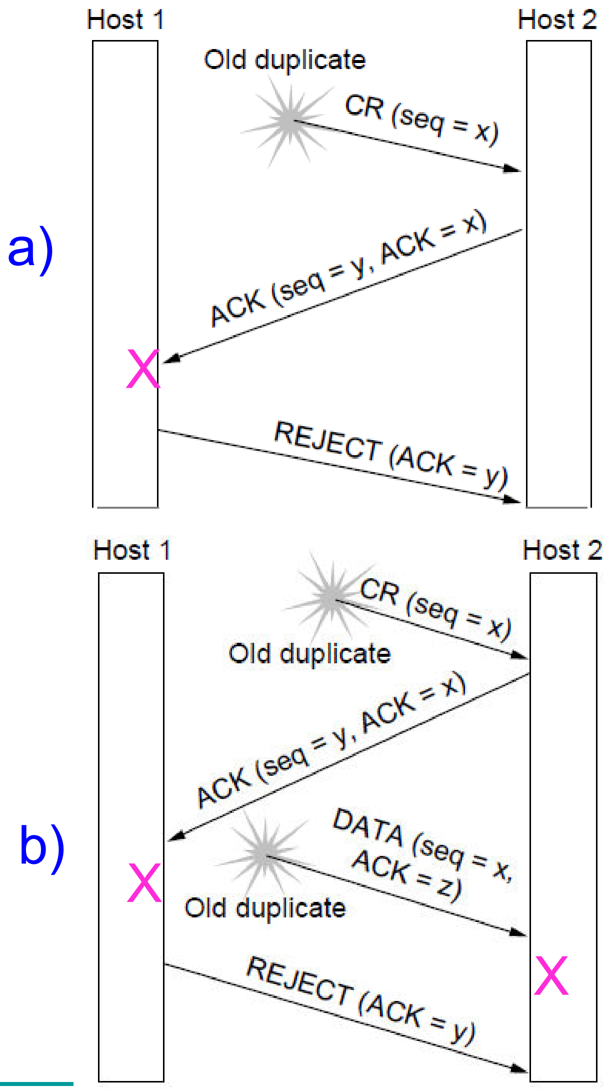


Three Way Handshake Contd.

- Three-way handshake protects against odd cases:

a) Duplicate CR. ACK cannot connect

b) Duplicate CR and DATA. Same plus DATA will be rejected (wrong ACK)

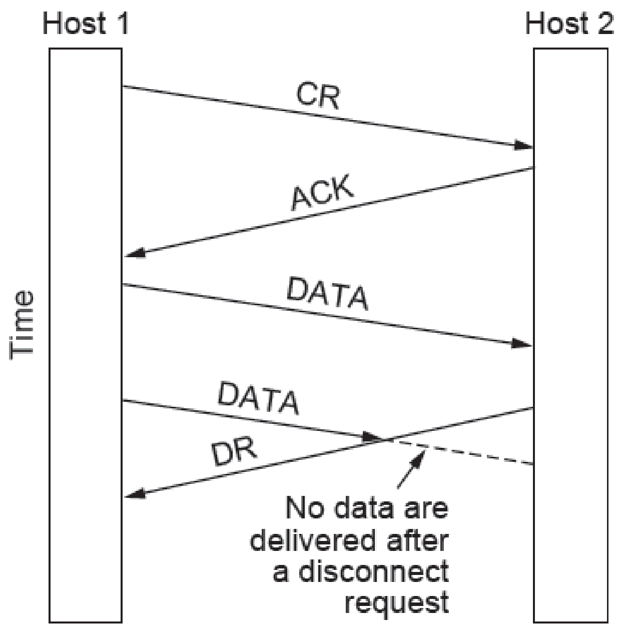


Connection Release

- Asymmetric Disconnection
  - Either party can issue a DISCONNECT, which results in DISCONNECT TPDU and transmission ends in both directions eventually
- Symmetric Disconnection
  - Both parties issue DISCONNECT, closing only *one direction at a time* - allows flexibility to remain in receive mode

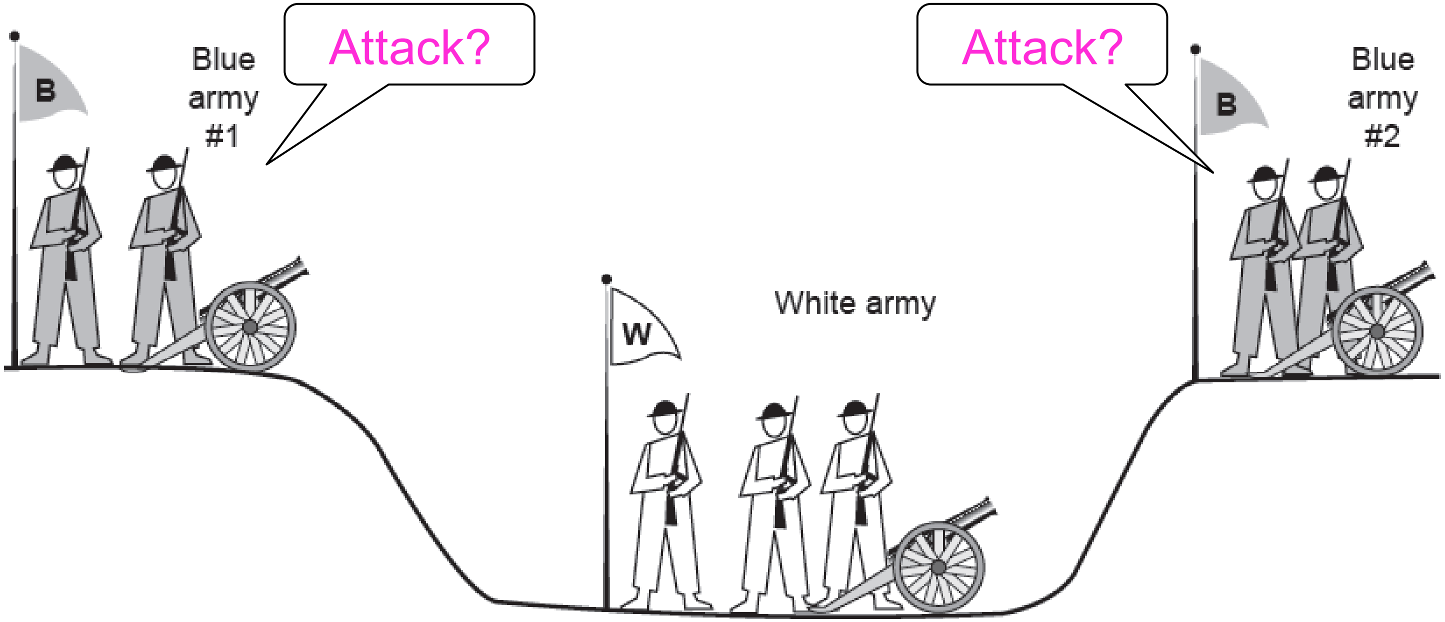
Connection Release (Cond.)

- Asymmetric release may result in data loss hence symmetric release is more attractive
- Symmetric release works well where each process has a set amount of data to transmit and knows when it has been sent



Can we do more with symmetric release: Generalizing the Problem

- **No protocol exists which can resolve the following ambiguity**
  - *Two-army* problem shows the pitfall of trying to reach an agreement


B
Blue army #1
Attack?
Attack?
B
Blue army #2
W
White army