

Transport Layer

COMP90007

Internet Technologies

1

Transport Layer Function

- Main function
 - provide efficient, reliable & cost-effective data transmission service to the processes in the application layer...independent of physical or data networks
- Recall: To Achieve this
 - It calls services provided by the network layer

2

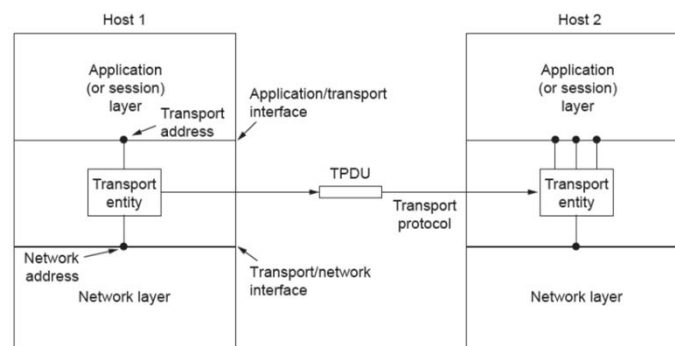
Transport Layer Services

- Transport Layer **Services** provide interfaces between the Application Layer and the Network Layer
- Transport **Entities** (the hardware or software which actually does the work) can exist in multiple locations:
- **Where and where it should not be (but sometimes is)?**
 - OS kernel
 - System library (library package bound into network applications)
- Not so much...
 - User process
 - Network interface card

3

Services Contd.

- Transport layer adds **reliability** to the network layer
 - Offers **connectionless** (e.g., UDP) and **connection-oriented** (e.g., TCP) services to applications
- Relationship between network, transport and application layers:



4

Transport Layer and Network Layer Services Compared

- If **Transport** and **Network** layers are so similar, why are there two layers?
- Transport layer code runs entirely on **hosts**, Network layer code runs almost entirely on **routers**.....
- ***Users have no real control over the network*** layer – Transport layer: we can improve QoS
- Transport layer ***fixes reliability problems*** caused by the Network layer (e.g., delayed, lost or duplicated packets)

5

Position of the Transport Layer

- The Transport Layer occupies a key position in the layer hierarchy because it clearly delineates
 - ***providers*** of data transmissions services
 - at the network, data link, and physical layers
 - ***users*** of reliable data transmission services
 - at the application layer
- In particular, ***users commonly access connection-oriented transport services*** for a reliable service on top of an unreliable network

6

Example:

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...*

7

Features of a *Simple* Transport Layer

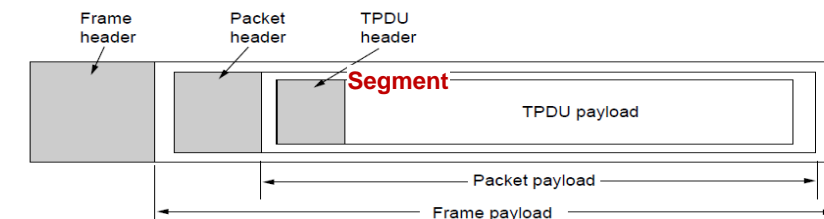
- Abstraction and primitives provide a **simpler API** for application developers independent of network layer

Primitive	Meaning
LISTEN	Block waiting for an incoming connection
CONNECT	Establish a connection with a waiting peer
RECEIVE	Block waiting for an incoming message
SEND	Send a message to the peer
DISCONNECT	Terminate a connection

8

Transport Layer Encapsulation

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



9

Transport Service Primitives/ Segments

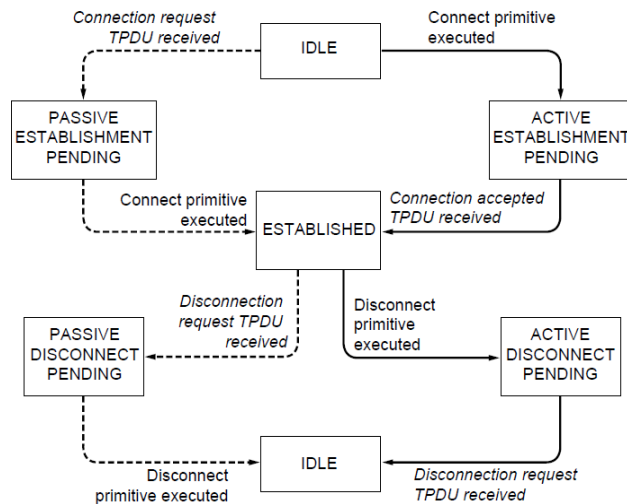
- Primitives that applications might call to transport data for a simple connection-oriented service:
 - Server executes **LISTEN**
 - Client executes **CONNECT**
 - Sends CONNECTION REQUEST TPDU to Server
 - Receives CONNECTION ACCEPTED TPDU to Client
 - Data exchanged using **SEND** and **RECEIVE**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

10

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



11

Elements of Transport Protocols

- Connection establishment
- Connection release
- Addressing

12

Connection Establishment in the Real World

- When networks can **lose, store and duplicate** packets, 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)

13

Reliable Connection Establishment

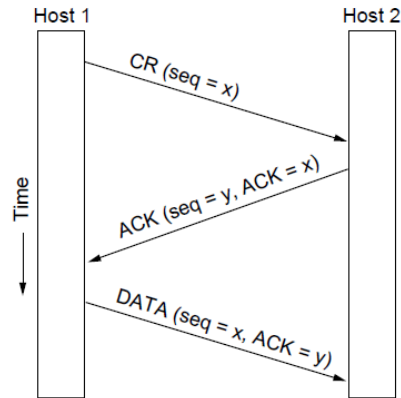
- Key challenge is to ensure reliability even though packets may be **lost, corrupted, delayed, and duplicated**
 - Don't treat an old or duplicate packet as new
 - (Use repeat requests and checksums for loss/corruption)
- **Approach:**
 - **Don't reuse sequence numbers within maximum segment lifetime**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

14

Three Way Handshake

- Three-way handshake used for initial packet

- Since no state from previous connection
- Both hosts contribute fresh seq. numbers
- CR = Connect Request

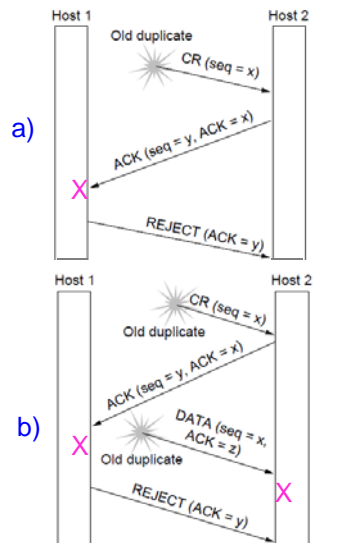


15

Three Way Handshake Contd.

- Three-way handshake protects against odd cases:

- Duplicate CR. Spurious ACK does not connect
- Duplicate CR and DATA. Same plus DATA will be rejected (wrong ACK).



16

Connection Release

■ Asymmetric Disconnection

- Either party can issue a DISCONNECT, which results in DISCONNECT TPDU and transmission ends in both directions

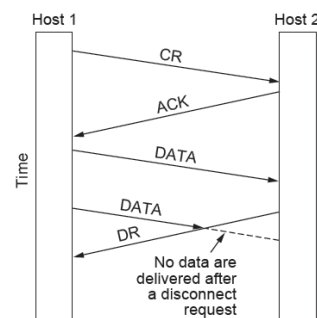
■ Symmetric Disconnection

- Both parties issue DISCONNECT, closing only *one direction at a time* - allows flexibility to remain in receive mode

17

Connection Release (Cont.)

- Asymmetric vs Symmetric connection release types
- **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



18

Generalizing the Connection Release Problem

- How do we decide the importance of the last message? Is it essential or not?
- No protocol exists which can resolve this ambiguity - Two-army problem shows pitfall of agreement

