Week 7: Transport Layer

Internet Technologies COMP90007

Lecturer: Tom Drummond

Semester 2, 2021

Introduction to Instructor

Prof Tom Drummond

Since June 2021: Melbourne Connect Chair of Digital Innovation for Society

From 2010-2021: Prof at Monash University
(2016-2021 Head of Department of Electrical and Computer Systems Engineering)

2008-2010 Lecturer in Information Engineering at University of Cambridge

2004-2008 PhD at Curtin University (Perth, WA)

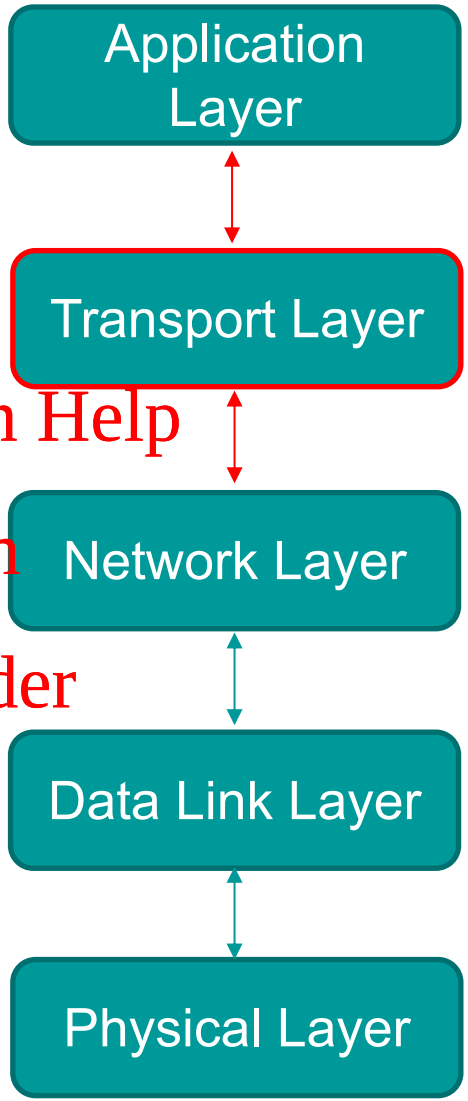
1985-1988 BA Mathematics, University of Cambridge

Areas of Interest: Computer Vision, Robotics, Machine Learning

Layered Network

Tentative Schedule

Week	Topic	
1	Introduction	
2	Physical Layer	
3	Data Link Layer	
4	Medium Access Control	Ling
5	Network Layer	
6	Network Layer	
7	Transport Layer	
8	Transport Layer	
9	Application Layer	
	Non-teaching period	Tom
10	Application Layer	
11	Network Security	
12	Review	



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

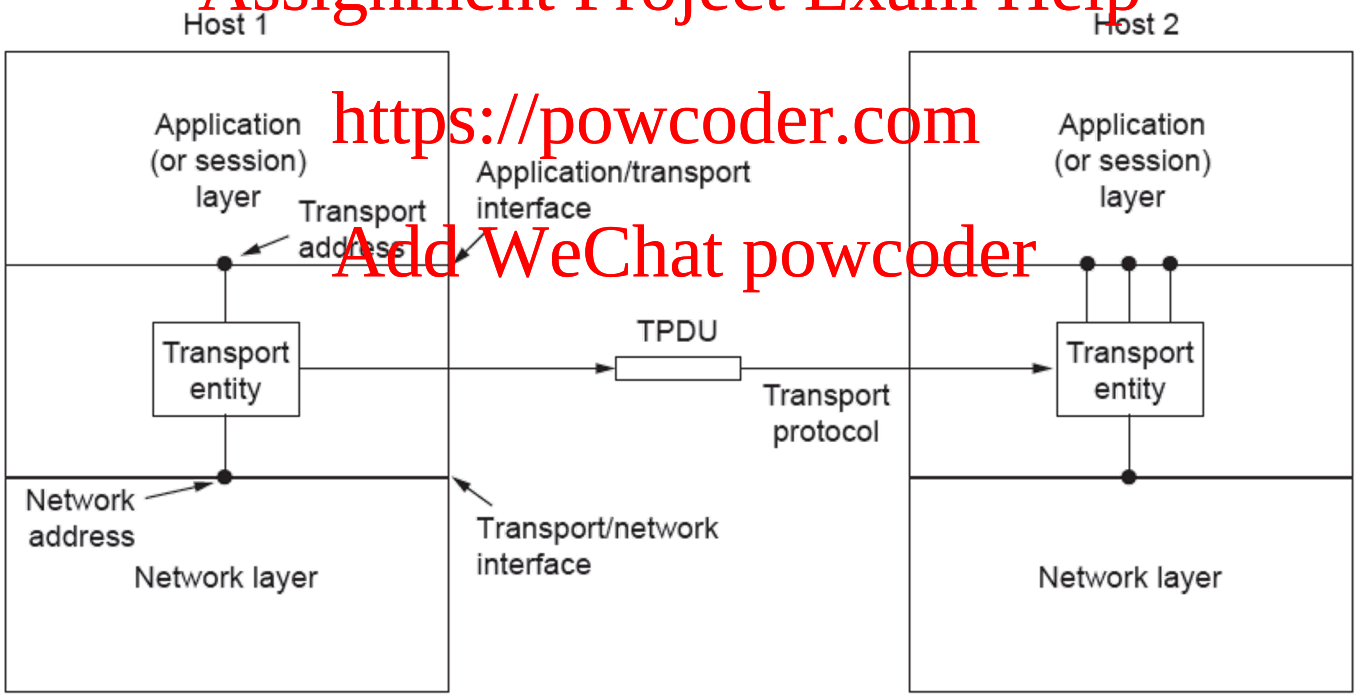
- Recall: To Achieve this
  - It calls services provided by the network layer

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

Services Contd.

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



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)

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

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…

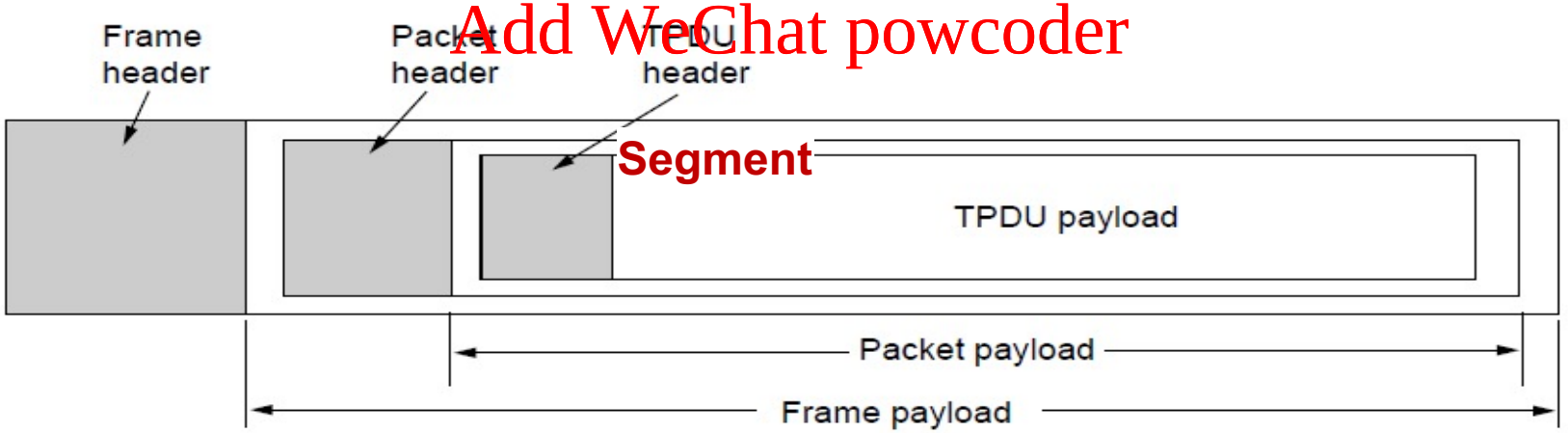
Features of a *Simple* Transport Layer

- 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

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)



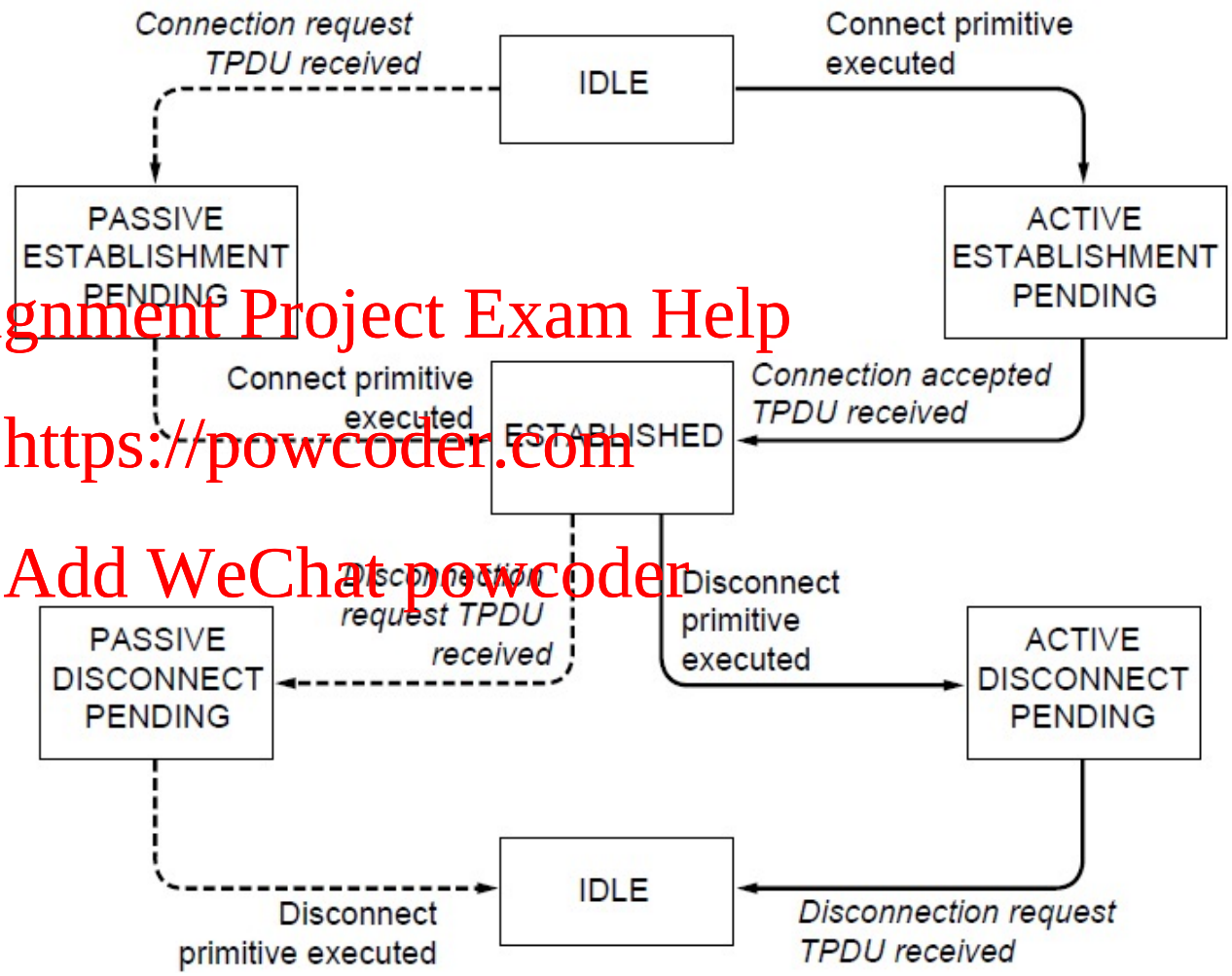
Transport Service Primitives/ Segments

- 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

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



Elements of Transport Protocols

- Connection establishment
- Connection release
- Addressing

Connection Establishment in the Real World

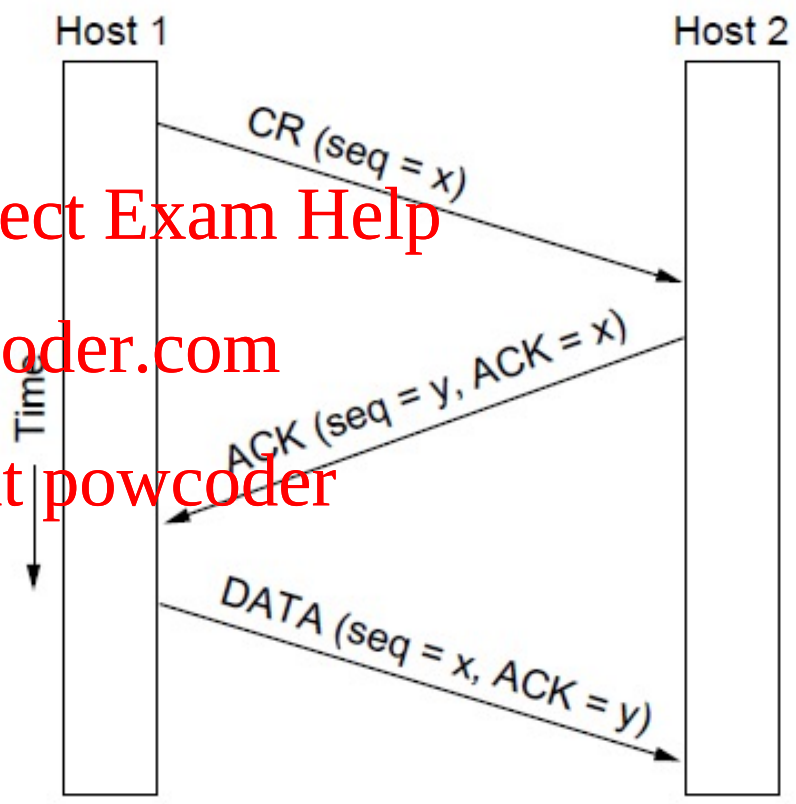
- When networks can lose, store and duplicate packets, connection establishment can be complicated
  - congested networks may delay acknowledgements
  - incurring repeated multiple transmissions
  - any of which may not arrive at all or out of sequence – delayed duplicates
  - applications degenerate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

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



Three Way Handshake Contd.

- Three-way handshake protects against odd cases:

a) Duplicate CR. Spurious ACK does not connect

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