CEN445 – Network Protocols and Algorithms

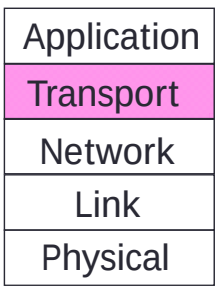
Chapter 6 – Transport Layer

6.1 Transport Layer Service

Dr. Mostafa Hassan Dahshan
Department of Computer Engineering
College of Computer and Information Sciences
King Saud University
mdahshan@ksu.edu.sa
http://faculty.ksu.edu.sa/mdahshan

Transport Layer

- Heart of protocol hierarchy (+ NL)
- Build on NL provide data transport from process at source to process at destination
- Reliability independent of physical network
- Abstraction needed by apps to use network

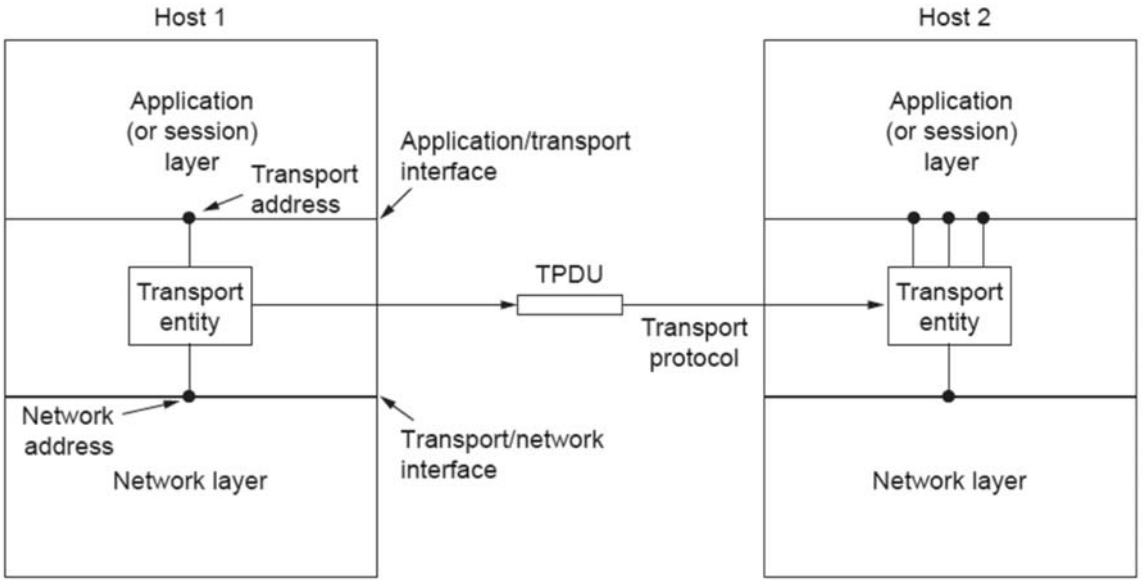


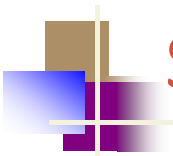
Services Provided to Upper Layers

- Efficient, reliable, transmission to app layer
- Use services provided by network layer
- TL work done by transport entity
- Transport entity: software or hardware in
  - operating system kernel
  - library package bound into network apps
  - separate user process
  - network interface card

Common in the Internet

Services Provided to Upper Layers



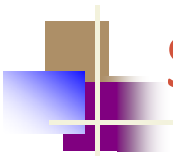


Services Provided to Upper Layers

- Similar services to network layer
  - connection-oriented and connectionless
  - addressing
  - flow control
- So why two distinct layers?

Services Provided to Upper Layers

- TL code runs entirely in user machines
- NL mostly runs in routers
- What if NL offers inadequate service?
- Users have no control over network
- TL can make service more reliable than NL
- TL hide net svc behind std set of primitives
- Programmer code work on wide variety of networks



Services Provided to Upper Layers

- Transport service provider
  - layers 1-4
- Transport service user
  - layers above 4
- TL in key position: boundary between two

Transport Service Primitives

- Connection oriented transport service
  - reliable service over unreliable network
  - as if two UNIX processes com over pipe
  - assume conn is 100% perfect
  - don't know about lost packets, congestion, ACK
- Connectionless provide unreliable service
  - only multiplexes connections over NL
  - will not focus on it in this chapter

Transport Service Primitives

- Simplified (hypothetical) set
- Apps call primitives to transport data
  - client: CONNECT, SEND, RECEIVE, DISCONNECT
  - server: LISTEN, RECEIVE, SEND, 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

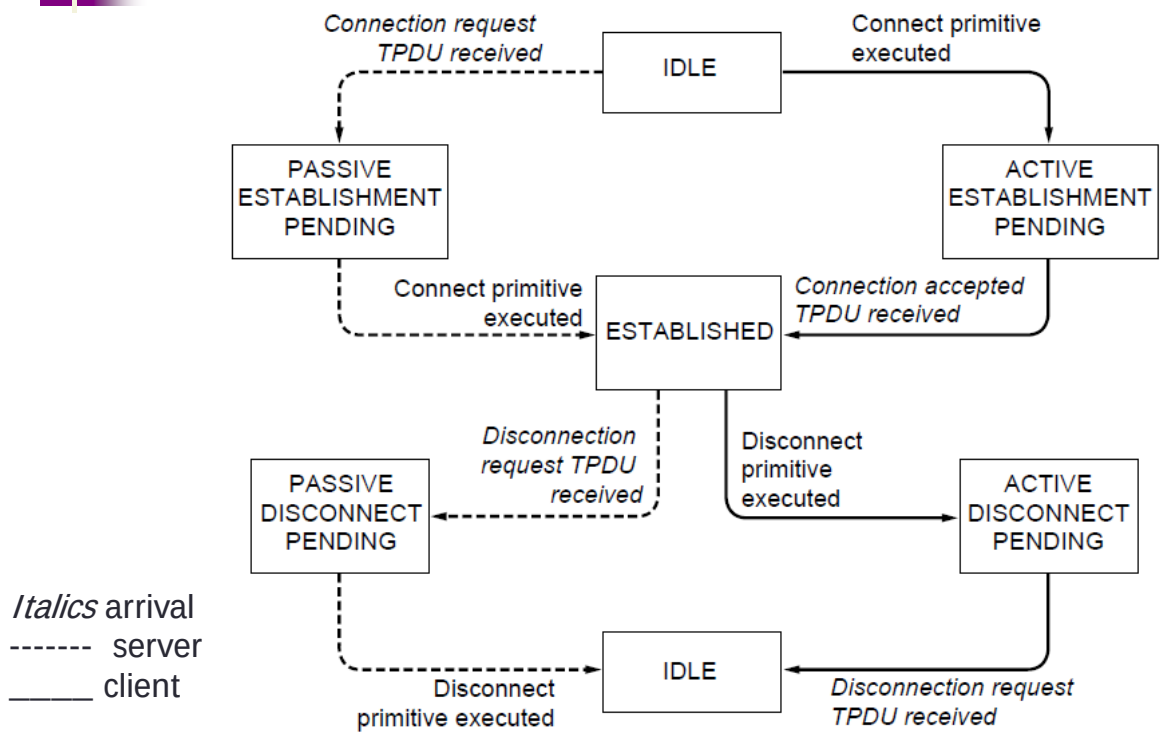
Transport Service Primitives

- CONNECT
  - client send CONNECTION REQUEST to server
  - if server is blocked on LISTEN
    - server sends CONNECTION ACCEPTED
    - when it arrives to client, connection is established
- Data exchanged using SEND, RECEIVE
  - one side blocks in RECEIVE, the other do SEND
- Conn no longer needed? DISCONNECT
  - either side send DISCONNECTION REQUEST
  - other side also do DISCONNECT

Transport Service Primitives

- Data exchange more complicated than NL
- Every data packet sent must be ACK'ed
- Control packet also acknowledged
- Transport entity must do timers, retrans
- All this is transparent to users
- User see connection as reliable bit pipe
- Put bit in a side, appear on other side
- Ability to hide complexity: main advantage of layered architecture

Transport Service Primitives



Berkeley Sockets

- Socket primitives used in UNIX for TCP
- Socket: connection endpoint
- Widely used for Internet programming
- Like simple set plus SOCKET, BIND, and ACCEPT

Primitive	Meaning
SOCKET	Create a new communication end point
BIND	Associate a local address with a socket
LISTEN	Announce willingness to accept connections; give queue size
ACCEPT	Passively establish an incoming connection
CONNECT	Actively attempt to establish a connection
SEND	Send some data over the connection
RECEIVE	Receive some data from the connection
CLOSE	Release the connection