Computer Communication Networks

Transport Layer

Dr. Raja Vara Prasad

Assistant Professor

IIIT Sri City

Transport Layer

Transport Layer

how two entities can communicate reliably over a medium that may lose and corrupt data ?

controlling the transmission rate of transport-layer entities in order to avoid

Or

recover from, congestion within the network.

Transport Layer Services

- **logical communication**
- Transport-layer **segments**
- Transport-layer protocol provides logical communication between *processes* running on different hosts
- a network-layer protocol provides logical communication between *hosts*
- services that a transport protocol can provide are often constrained by the service model of the underlying network-layer protocol
- a transport protocol can offer reliable data transfer service to an application even when the underlying network protocol is unreliable
- can use encryption

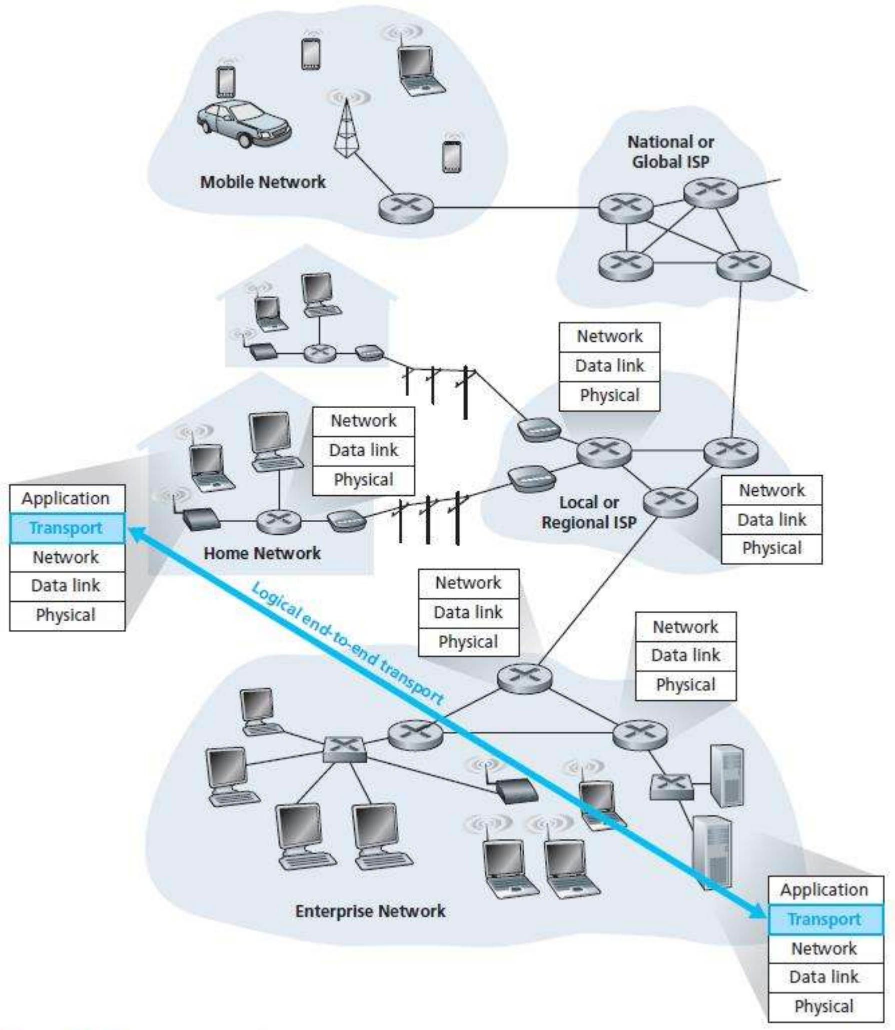


Figure 3.1 ♦ The transport layer provides logical rather than physical communication between application processes

Transport Layer in the Internet

- Internet Protocol. IP provides logical communication between hosts.
- IP service model is a **best-effort delivery service**
- “best effort” to deliver segments between communicating hosts → *makes no guarantees.*
- not guarantee segment delivery
- it does not guarantee orderly delivery of segments
- does not guarantee the integrity of the data in the segments

UDP Services:

process-to-process data delivery and error checking

TCP:

- reliable data transfer
- correct and in order → using flow control, sequence numbers, acknowledgments, and timers
- **congestion control**

Multiplexing and Demultiplexing

- host-to-host delivery service provided by the network layer
- process-to-process delivery service for applications running on the hosts – Transport Layer

- a process can have one or more **sockets**
- transport layer in the receiving host does not deliver data directly to a process → to an intermediary socket
- more than one socket in the receiving host → each socket → unique identifier
- Each transport-layer segment has a set of fields

Demultiplexing:

- receiving end → the transport layer examines these fields to identify the receiving socket
- directs the segment to that socket
- **Multiplexing**

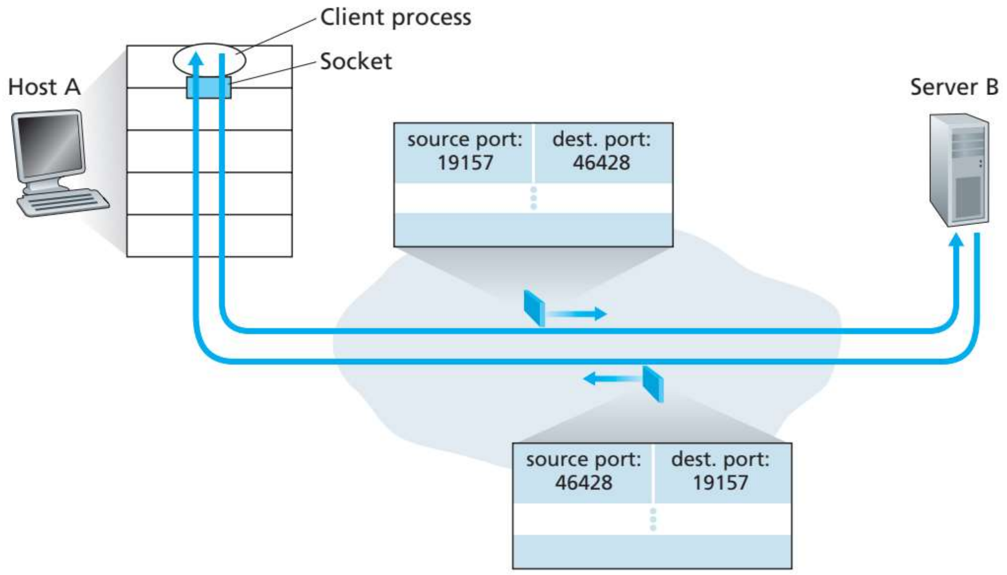
  gathering data chunks at the source host from different sockets
- encapsulating each data chunk with header information to create segments
- passing the segments to the network layer is called

Connectionless Multiplexing and Demultiplexing

UDP socket:

- transport layer assigns a port number in the range 1024 to 65535 that is currently not being used by any other UDP port in the host

  Ex: A process in Host A, with UDP port 19157 → to send a chunk of application data to a process with UDP port 46428 in Host B.
- UDP socket: identified by a two-tuple → a destination IP address and a destination port number



if two UDP segments have different source IP addresses and/or source port numbers, but have the same *destination* IP address and *destination* port number ?

Connection Oriented Multiplexing and Demultiplexing

TCP socket:

- TCP socket is identified by a four-tuple
  source IP , source port number, destination IP, destination port number
- host uses all four values to direct the segment to the appropriate socket

  server host may support many simultaneous TCP connection sockets, with
  each socket attached to a process, and with each socket identified by its own four tuple.

Web Servers and TCP:

---all segments will have destination port 80.
---Web servers often use only one process, and create a new thread with a new connection socket for each new client connection .
---client and server using persistent HTTP → same server socket
---non-persistent HTTP → a new TCP connection is created and closed for every request/response
---frequent creating and closing of sockets --- severely impact the performance of a busy Web server