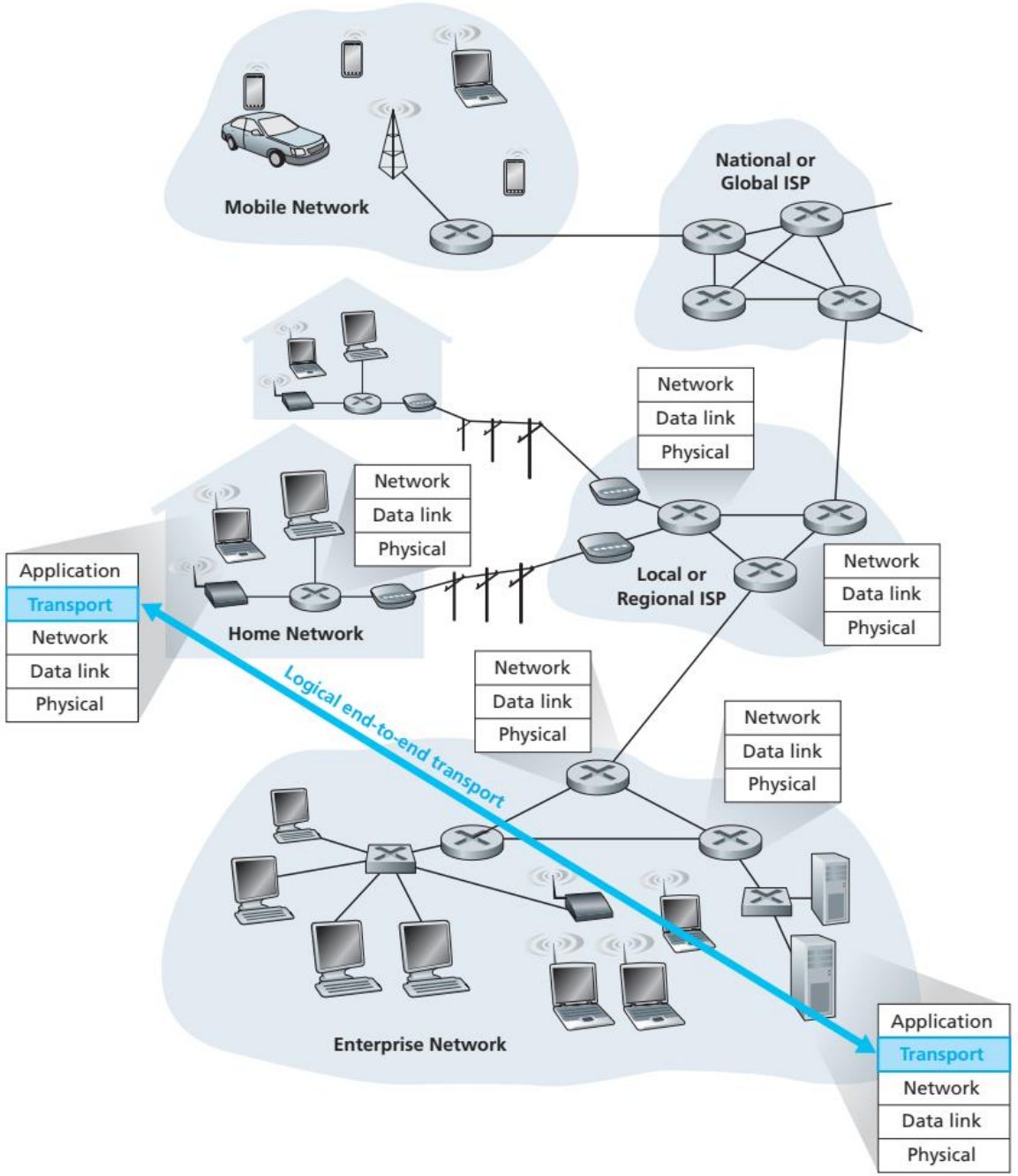
Module 3: Transport Layer (Lecture – 1)

Dr. Nirnay Ghosh
Assistant Professor
Department of Computer Science & Technology
IIEST, Shibpur

Transport Layer Services

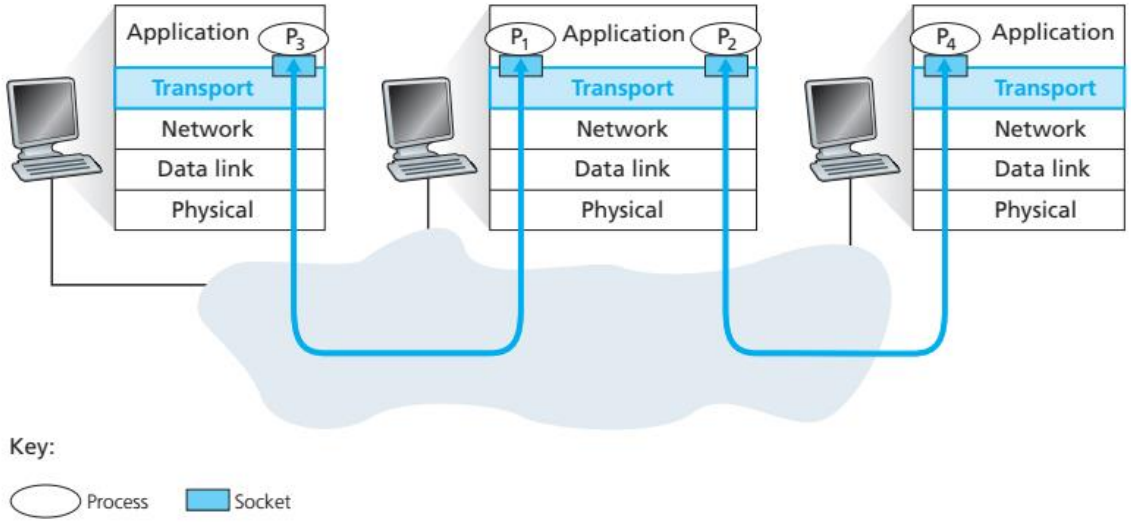
- Establishes logical connections between application processes running on different hosts
- Protocols are implemented at end systems
  - Transfer messages from application processes to the network layer and vice-versa
- Transport protocols: two distinct protocols
  - User Datagram Protocol (UDP) – unreliable, connectionless service
  - Transmission Control Protocol (TCP) – reliable, connection-oriented service
- Can offer reliable data transfer service even if the underlying network protocol is unreliable (packets get lost, garbled, or duplicated)
- Converts application layer messages into segments



Logical Connection Provided by Transport Layer Protocols

Transport Layer Services: Multiplexing and Demultiplexing

- IP: network layer protocol
  - Establishes logical connection between hosts (host-to-host delivery)
  - "Best effort" delivery service
  - Unreliable: does not guarantee orderly delivery of segments, integrity of the data in the segment
- TCP & UDP: transport layer protocols
  - Extends host-to-host delivery to process-to-process delivery
  - Integrity checking by including error detection fields in segment headers
- Process (a part of the network application): can have one or more sockets
  - Door through which data passes from the network to the process and vice-versa
  - Two types: UDP or TCP
  - Has unique identifier

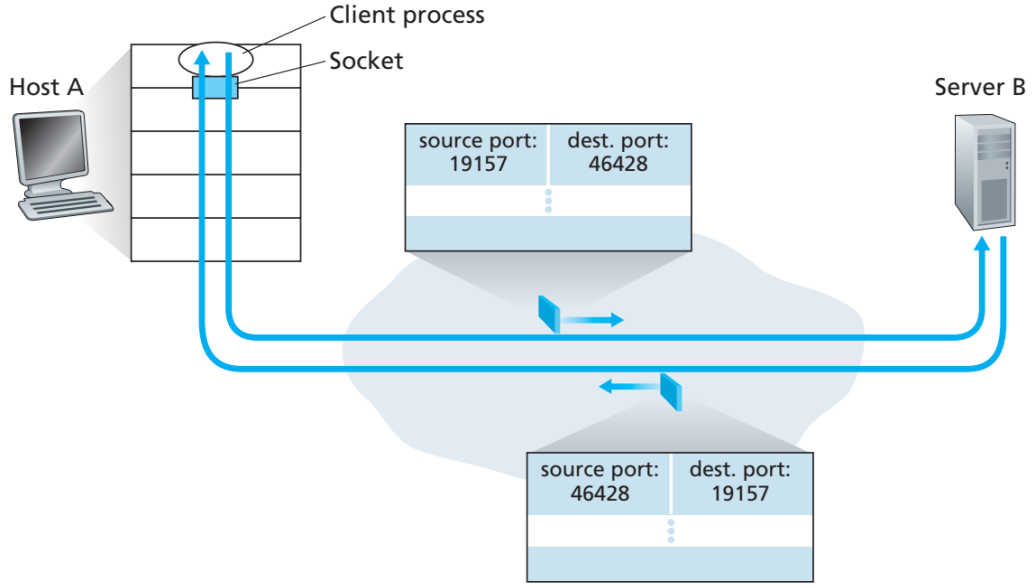


Transport Layer: Multiplexing & Demultiplexing

- Multiplexing:
  - Task of gathering data chunks at the source host from different sockets
  - Create segments by encapsulating each data chunk with header information
  - Passing the segments to the network layer
  - Requirements: socket with unique identifiers, special field in each segment that identifies to socket for delivery of the segment

Transport Layer Services: Multiplexing and Demultiplexing

- Multiplexing
  - Special fields in segments are: source port number, destination port number
    - Port number: 16-bit field (0 – 65535)
    - Well-known port numbers: 0 -1023 (reserved for use by well-known application protocols – HTTP, FTP, etc.)
- Demultiplexing
  - Task of delivering the data carried in a transport layer segment to the correct socket
  - Each socket – assigned a port number
  - Transport layer examines the destination port number in the incoming segment
  - Directs the segment to the corresponding port
  - Data passes through the socket into the attached process

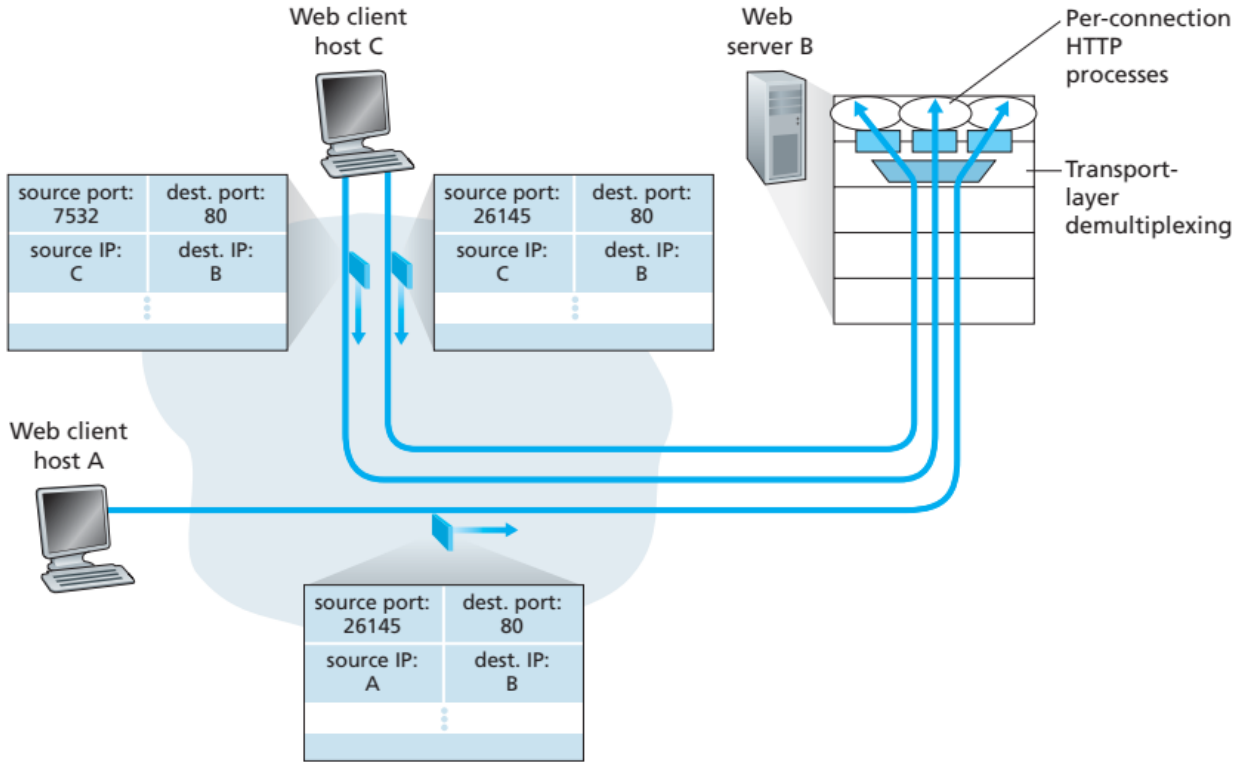


Source and Destination Port Numbers

- Connection-less multiplexing & demultiplexing
  - Service provided by an UDP socket
  - Transport layer at Host A: assigns arbitrary port number to the UDP socket (in the range 1024 to 65535) at client-end (19157)
  - Server-side socket: assigned specific port number to a user program (46428)
  - Transport layer at Server B: directs the segment to destination port (46428)
- UDP socket: identified by a two-tuple
  - Destination IP address and destination port number
  - UDP segments with different source IP address and/or port number but same destination IP address and port number: directed to the same destination process via the same destination socket

Transport Layer Services: Multiplexing and Demultiplexing

- Connection-oriented multiplexing & demultiplexing
  - Service provided by TCP socket
  - A server host may support many simultaneous TCP connection sockets
  - TCP socket
    - Attached to a process
    - Identified by a four-tuple – source IP address, source port number, destination IP address, destination port number
  - All four values are used to direct (demultiplex) the segment to the appropriate socket
  - TCP segments with different source IP addresses or source port numbers will be directed to two different sockets



Two Clients using the same Destination Port Number 80 to Communication with the same Web Server Application

Connectionless Transport: UDP

- UDP: connectionless process-to-process data delivery and error-checking
- Unreliable like the IP protocol (network layer)
- Does not guarantee orderly data delivery
- Connectionless: no handshaking mechanism between sender and receiver before sending a segment
- Many applications are better suited for UDP (e.g., DNS, SNMP, streaming multimedia, etc.):
  - Real-time applications do not want to overly delay segments – can tolerate some data loss
  - No delay in establishing connection
  - Better suited for applications which do not need reliability
  - No overhead in maintaining connection state: receive and send buffers, congestion control parameters, sequence and acknowledgement number parameters
  - Small packet overhead: UDP segment has only 8 bytes of header information (compared to TCP which is 20 bytes)

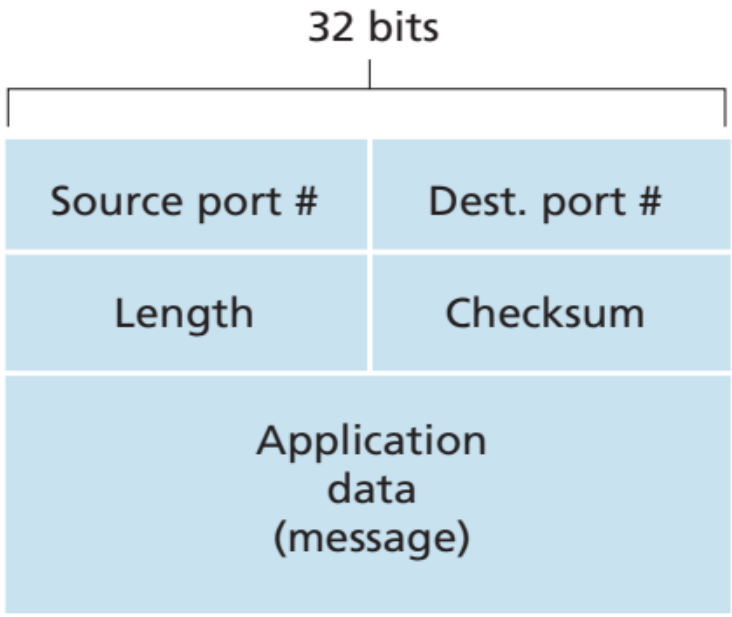
Application	Application-Layer Protocol	Underlying Transport Protocol
Electronic mail	SMTP	TCP
Remote terminal access	Telnet	TCP
Web	HTTP	TCP
File transfer	FTP	TCP
Remote file server	NFS	Typically UDP
Streaming multimedia	typically proprietary	UDP or TCP
Internet telephony	typically proprietary	UDP or TCP
Network management	SNMP	Typically UDP
Routing protocol	RIP	Typically UDP
Name translation	DNS	Typically UDP

Popular Internet Applications and their Underlying Transport Protocols

- Email, remote terminal access, the Web, file transfer – need reliable data transfer service – TCP is preferred over UDP
- Routing Information Protocol (RIP): uses UDP
  - RIP updates are sent periodically (typically after every five minutes)
  - Lost, out-of-date updates become useless and will be replaced by more recent updates

Connectionless Transport: UDP

- Network management application (SNMP) – uses UDP
  - Such applications run when the network is in a stressed state
  - Reliable, congestion-controlled data transfer is difficult to achieve
- Multimedia data – use both TCP and UDP
  - E.g.: Internet phone, real-time video conferencing, streaming of stored audio and video
  - Can tolerate a small amount of packet loss
  - Reliability of data is not absolutely critical
- Drawbacks: running multimedia applications over UDP
  - UDP lacks congestion control: high loss rates between UDP sender and receiver
  - Other TCP-based applications may reduce sending rates in the face of congestion



UDP Segment Structure

- UDP Segment Structure: Header - 4 fields (2 bytes each)
  - Source and Destination Port Numbers: allows the destination host to pass the application data to correct process running on the destination system (demultiplexing)
  - Length: Number of bytes in the UDP segment (header + data)
  - Checksum: used by the receiving host to check whether errors have been introduced into the segment
  - Message: data from the application (e.g., query/response message from DNS, audio samples for streaming audio applications, etc.)