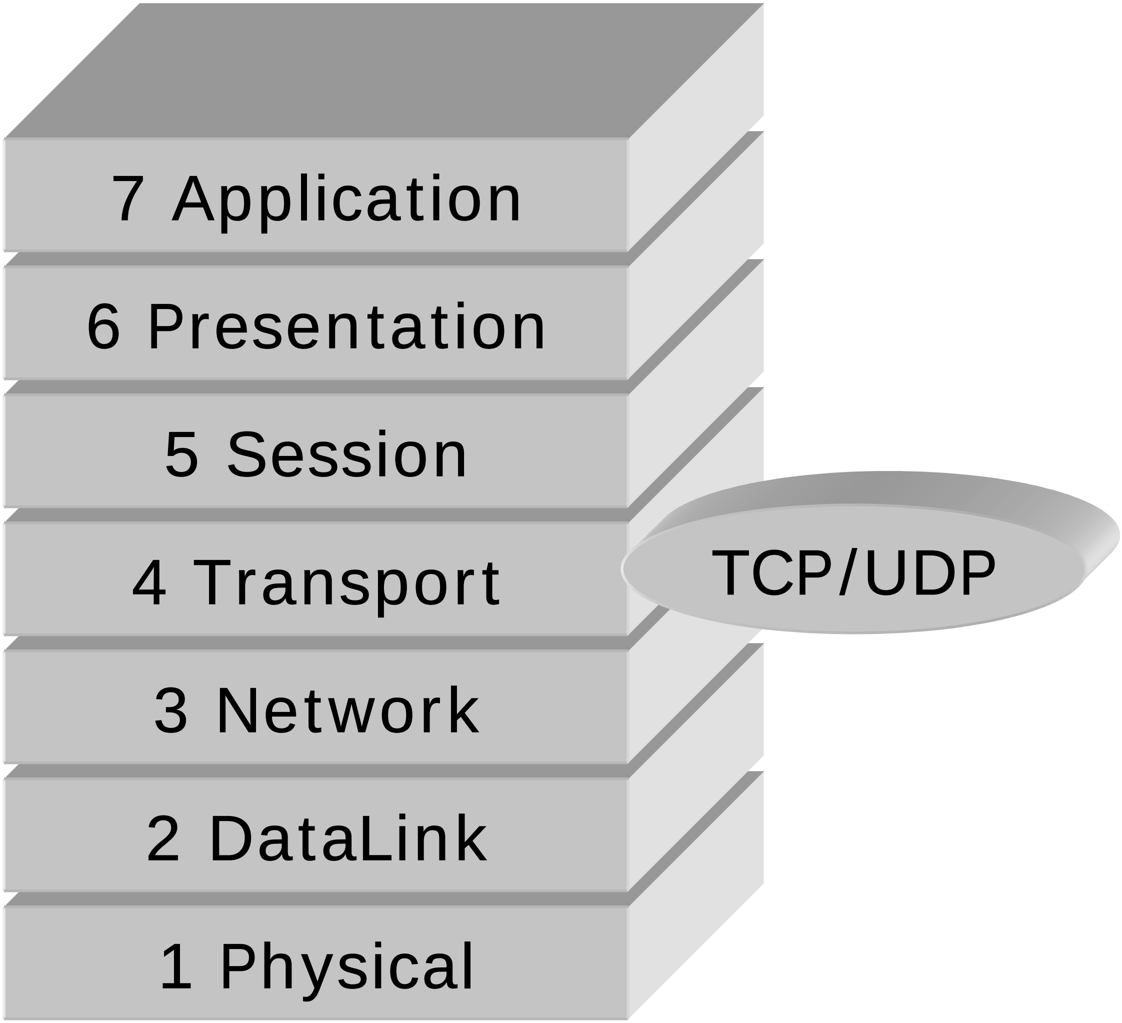
Where Are We?


7 Application
6 Presentation
5 Session
4 Transport
TCP/UDP
3 Network
2 DataLink
1 Physical

Transport vs Network Layer

- Network layer:

- Provides host-to-host communication
- Source and destination addresses are computers (attachment points)
- *Machine-to-machine* networking
- Datagrams

- Transport layer:

- Logical communication between processes
- Relies on and enhances the services provided by the network layer
- Segment: Unit of data exchanged between transport layer entities

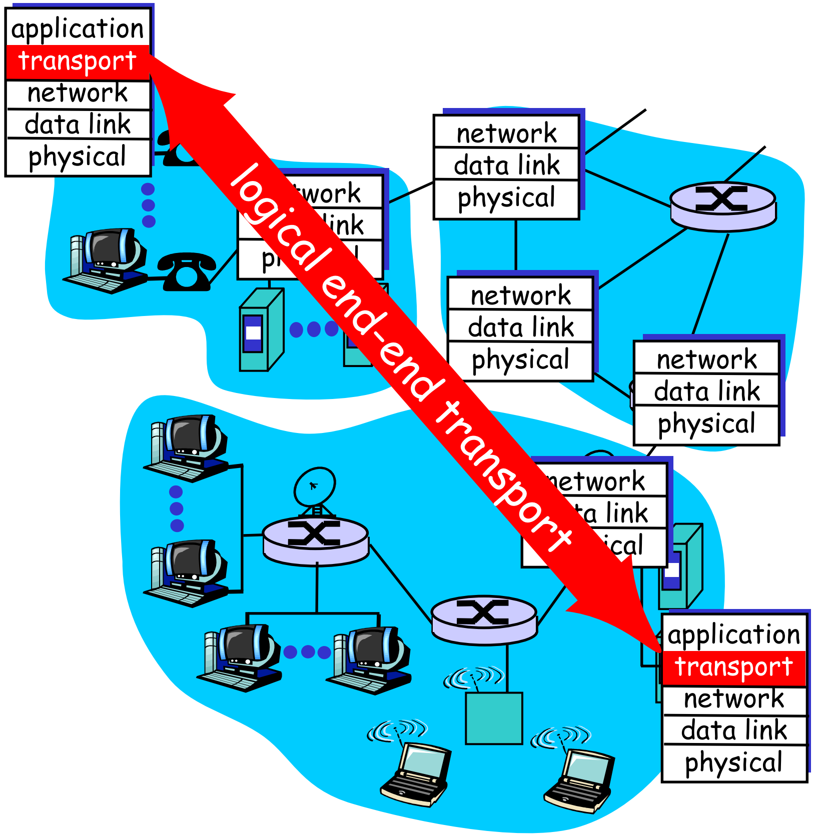
Transport Layer Services and Protocols

- Provide *logical communication* between application processes running on different hosts

  - Application-to-application communication

- Need extended addressing mechanism to identify applications
- Called end-to-end
- Optionally provide:

  - Reliability
  - Flow control
  - Congestion control



TCP/IP Transport Layer Protocols

- UDP (User Datagram Protocol):

 - Unreliable, unordered delivery

 - Transport layer extension of ``best-effort’’ IP; connectionless

- TCP (Transmission Control Protocol):

 - Reliable, ordered delivery

 - Connection-oriented

- Services not available:

 - Delay guarantees

 - Bandwidth guarantees

Ports

- A *connection* is identified by 2 end-points
  - An end-point is an *IP address* and an associated *port*
  - There can be many connections coming through a single port
  - Cannot use OS or application related quantity (process ID, task number, job name)
- TCP/IP uses numeric port numbers
  - Identifies which service within the host machine you wish to connect to
- The standard defines ports for both
  - UDP and TCP for common applications
- Other port numbers are available for private use
  - Both communicating machines must agree on what port numbers mean what

Multiplexing

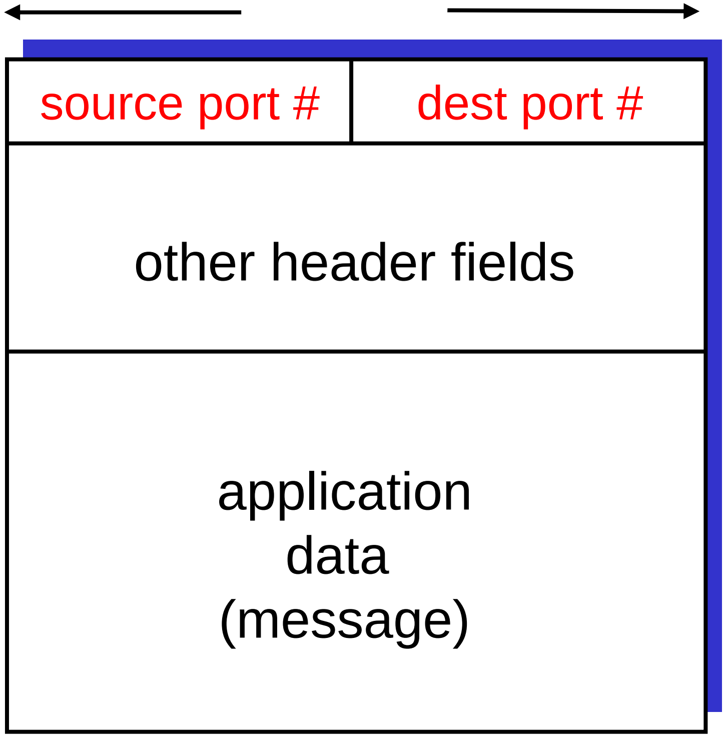
- Multiplexing:

  - Gathering data from multiple application processes, enveloping data with header (later used for demultiplexing)

  - Based on sender, receiver port numbers, IP addresses

    * Source, destination port numbers in each segment

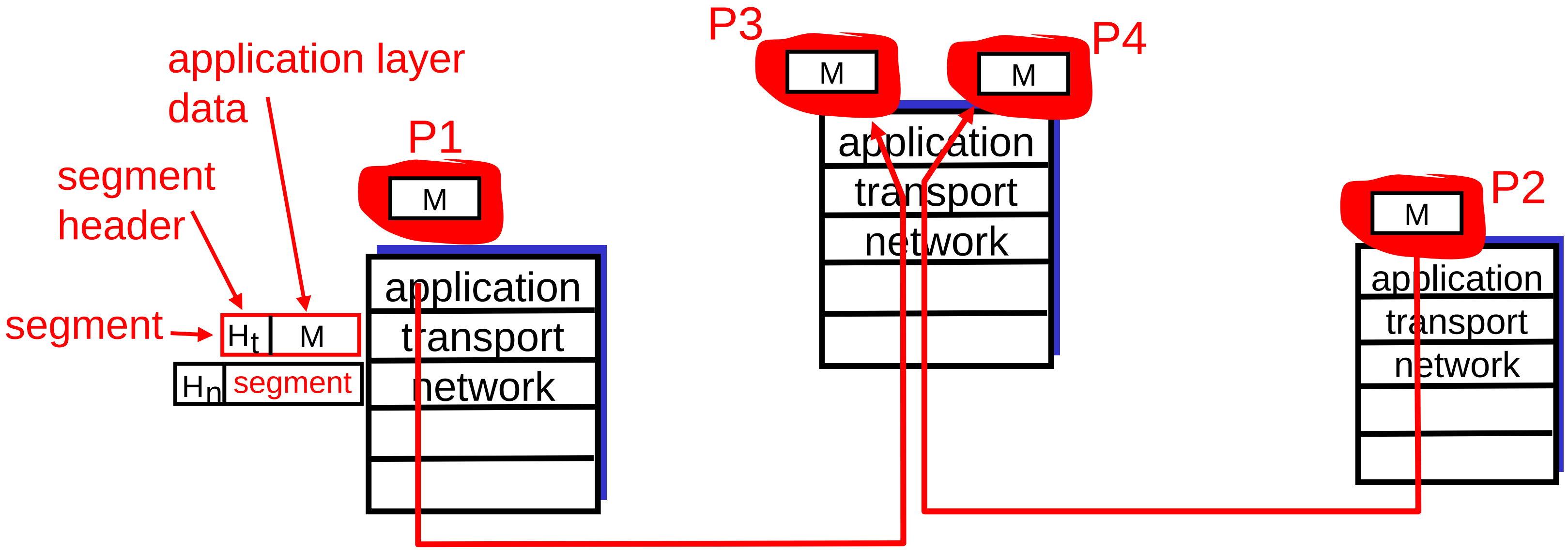
    * Recall: well-known port numbers for specific applications



TCP/UDP segment format

Demultiplexing

- Delivering received segments to correct application layer processes



How Demultiplexing Works

- Host receives IP datagrams

  - Each datagram has source IP address, destination IP address

  - Each datagram carries one transport layer segment

  - Each segment has source, destination port numbers

- Host uses IP addresses and port numbers to direct the segment to the appropriate application/process

User Datagram Protocol (UDP)

- In TCP/IP protocol suite, using IP to transport datagram (similar to IP datagram)
- Allows an application to send datagram to other application on the remote machine
- Delivery and duplicate detection are not guaranteed
- Connectionless; each UDP segment handled independently of others
- Low overhead: Faster than TCP
- RFC 768

UDP Characteristics

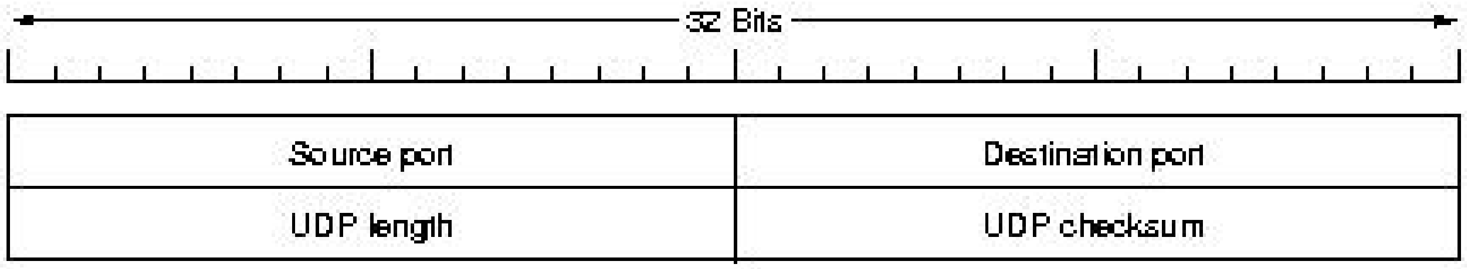
- **End-to-end**: An application sends/receives data to/from another application
- **Connectionless**: Application does not need to preestablish communication before sending data; application does not need to terminate communication when finished
- **Message-oriented**: Application sends/receives individual messages (UDP datagram), not packets
- **Best-effort**: Same best-effort delivery semantics as IP, i.e. a message can be lost, duplicated, and corrupted
- **Arbitrary interaction**: Application communicates with many or one other application(s)
- **Operating system independent**: Identifying applications does not depend on OS

The Need for UDP

- No connection establishment (which can add delay)
- Simple: No connection state at sender, receiver
- Small segment header
- No congestion control: UDP can blast away as fast as desired
- Often used for streaming multimedia applications

 - Loss tolerant, rate sensitive

UDP Header



Source port	Destination port
UDP length	UDP checksum

- **Source port (16 bits):** Port number of the sender
- **Destination port (16 bits):** Port number of the intended recipient; UDP software uses this number to demultiplex a datagram to the appropriate higher-layer software
- **Length (16 bits):** Length of the entire UDP datagram, including header and data
- **Checksum (16 bits):** Checksum of entire datagram (including data and pseudo header)

UDP Checksum

- Goal: Detect ``errors’’ (e.g., flipped bits) in transmitted segment
- Sender:

  - Treat segment contents as sequence of 16-bit integers

  - Checksum: Addition (1’s complement sum) of segment contents

  - Sender puts checksum value into UDP checksum field
- Receiver:

  - Compute checksum of received segment

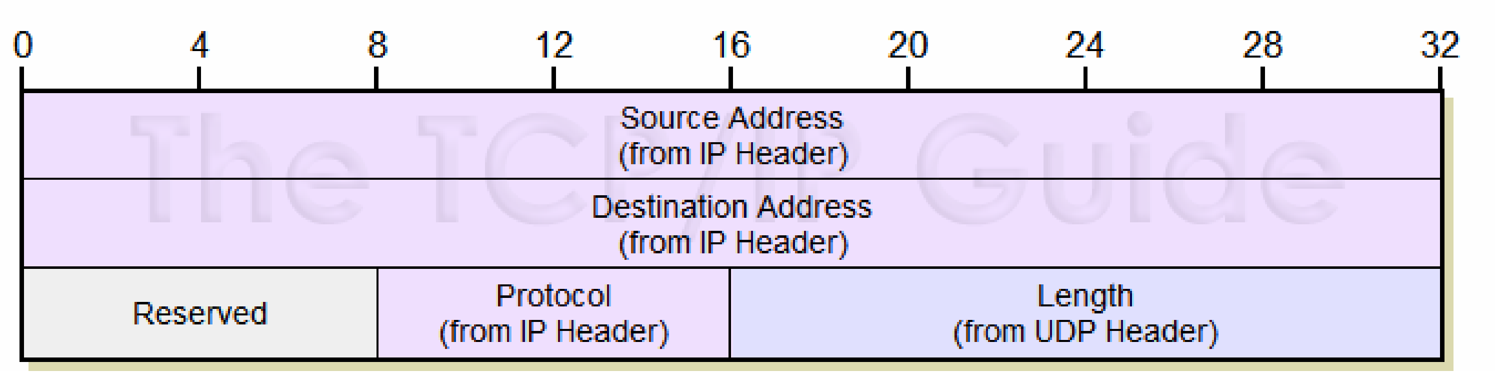
  - Check if computed checksum equals checksum field value

    * NO: Error detected

    * YES: No error detected

UDP Checksum and Pseudo Header

- UDP uses a pseudo header to verify that the UDP message has arrived at both the correct machine and the correct port



Source Address (from IP Header)		
Destination Address (from IP Header)		
Reserved	Protocol (from IP Header)	Length (from UDP Header)

- Prepended to the real UDP message
- Computed over the combination of the pseudo header and the real UDP message
- The pseudo header is used only for this calculation and is then discarded; it is not actually transmitted
- Destination creates the same pseudo header when calculating its checksum to compare to the one transmitted in the UDP header

Connectionless Demultiplexing

- Socket: An abstraction that provides an interface for processes to specify and access a connection point (TCP or UDP)
- UDP socket identified by two-tuple:

 - (dest IP address, dest port number)

- When host receives UDP segment:

 - Checks destination port number in segment

 - Directs UDP segment to socket with that port number

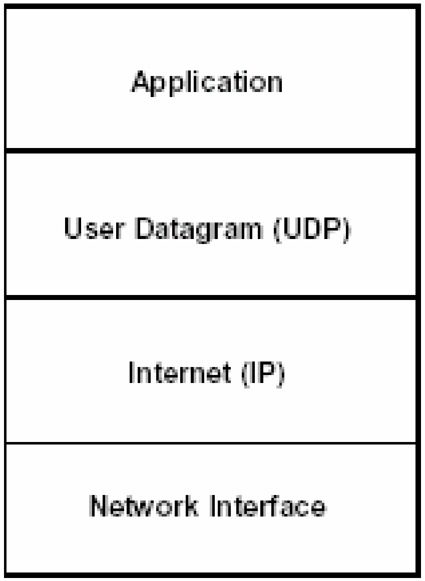
- IP datagrams with different source IP addresses and/or source port numbers directed to same socket

Encapsulation and Layering

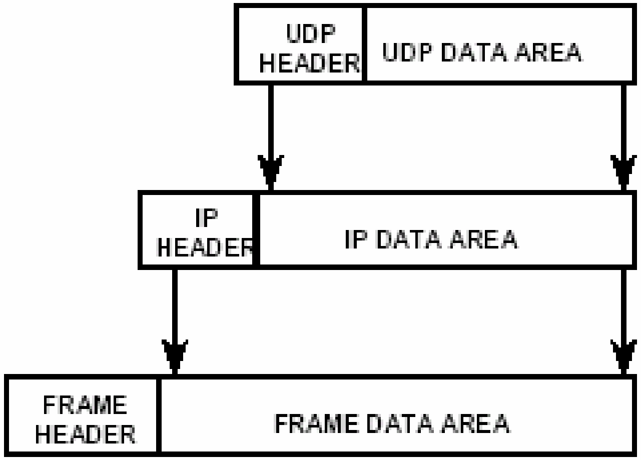
- UDP segment is encapsulated into an IP datagram
- IP datagram in turn is encapsulated into a physical frame for actually delivery

- Protocol layering

Conceptual Layering



- UDP encapsulation


UDP HEADER
UDP DATA AREA
IP HEADER
IP DATA AREA
FRAME HEADER
FRAME DATA AREA