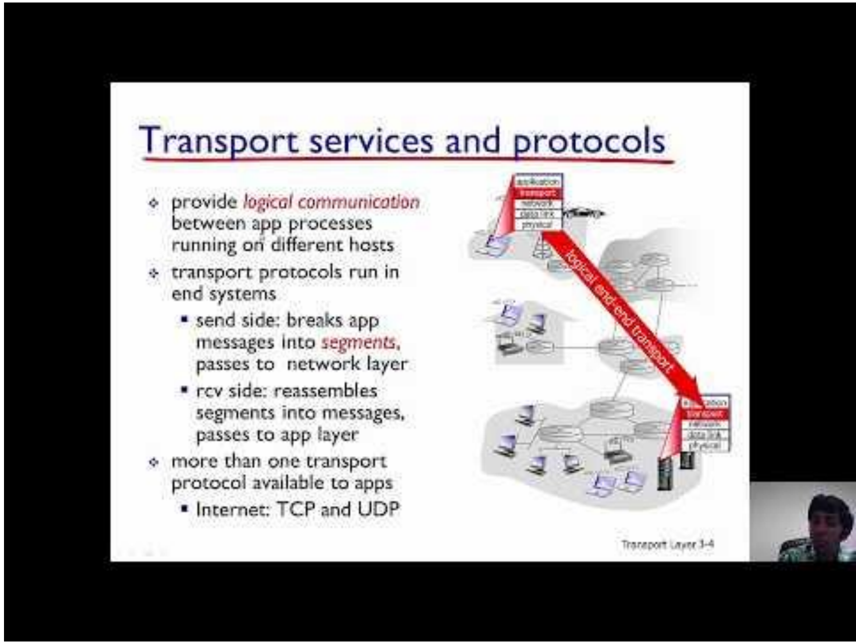
Computer Networks

Transport Layer Services
User Datagram Protocol

Jianping Pan
Fall 2022



Transport services and protocols

- provide *logical communication* between app processes running on different hosts
- transport protocols run in end systems
  - send side: breaks app messages into *segments*, passes to network layer
  - rcv side: reassembles segments into messages, passes to app layer
- more than one transport protocol available to apps
  - Internet: TCP and UDP

Transport Layer 3-4

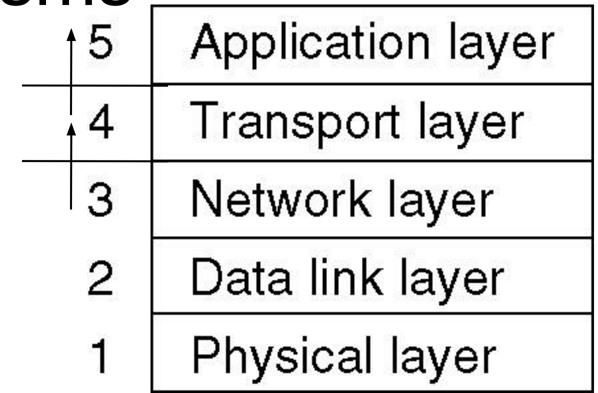
Review: application layer

- HTTP: hypertext transfer protocol
  - client-server model
  - request-reply transaction
  - normally based on TCP
- DNS: domain name system
  - DNS hierarchy
  - DNS queries
  - normally based on UDP

Q: why TCP for HTTP, UDP for DNS?

Today's topics

- Transport-layer protocol elements
  - services provided to application layer
    - to support HTTP, DNS, etc
  - services provided by network layer
    - e.g., by IP
  - transport-layer protocol mechanisms
    - i.e., how to fill the gap



5 Application layer

4 Transport layer

3 Network layer

2 Data link layer

1 Physical layer

Transport layer services

- Services provided by transport layer
  - – endpoint-to-endpoint communication
    - *endpoint*: an application **process** in end-hosts
  - – connection-oriented vs connectionless
  - – data transfer: reliable vs unreliable
- Example: Internet transport-layer services
  - – connection-oriented, reliable by TCP
  - – connectionless, unreliable by UDP

Apps:
HTTP
DNS

Network layer services

- Services provided by network layer
  - move packets from one end-host to another
  - possibly through many intermediate systems*
- Example: Internet network-layer services
  - IP: store-and-forward packet switching
  - packets may get
    - lost at communication link, router or receiver buffer
    - duplicated
    - corrupted
    - reordered

Q: possible causes?

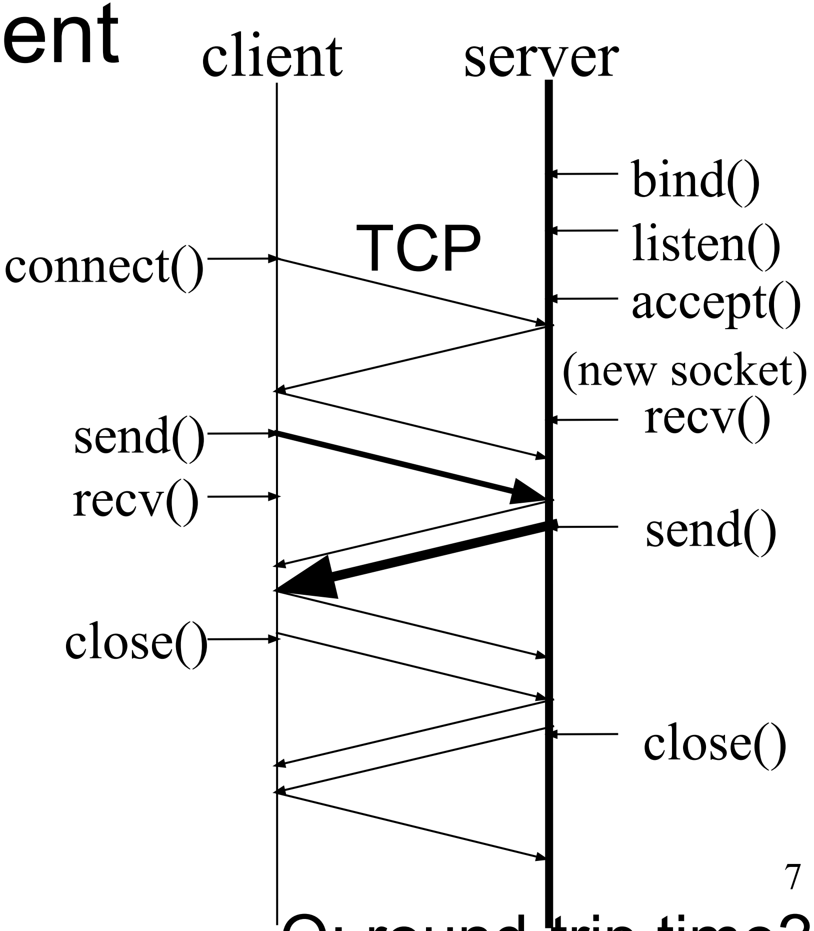
* routers

Transport layer protocols

- Protocol mechanisms
  - addressing and multiplexing
    - how to identify an *endpoint* in an *end-host*
  - connection management
    - for connection-oriented transport services
  - flow control: avoid overwhelming the receiver
  - error control
    - for reliable transport services
  - congestion control: avoid overloading the network

Example: Socket API

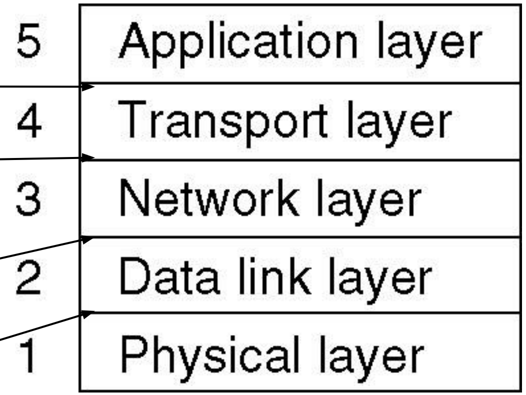
- Connection establishment
  - bind(), listen()
  - connect()
  - accept()
- Data transfer
  - send(), recv()
- Connection release
  - close()



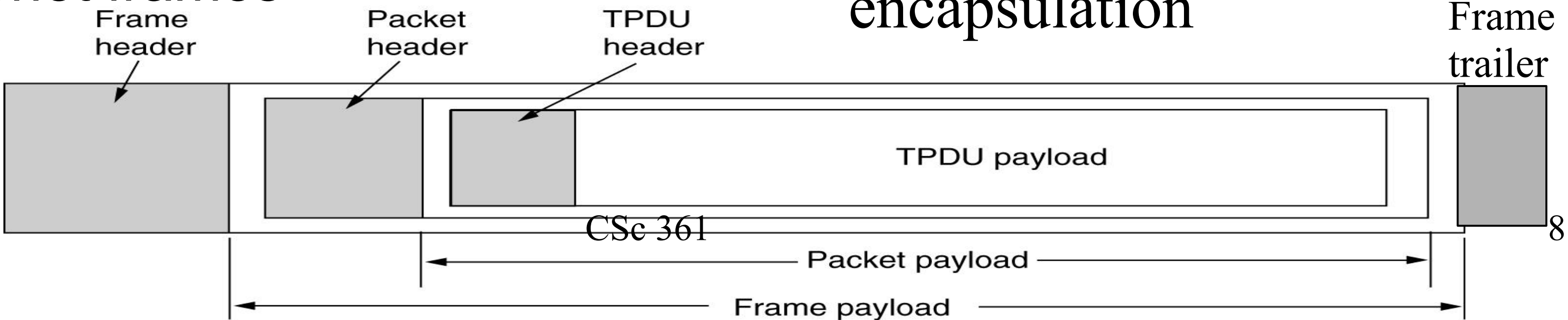
Q: round-trip time?

What's under Socket?

- Socket
  - socket messages
- TCP
  - TCP segments with TCP header
- IP
  - IP packets with IP header
- Ethernet
  - Ethernet frames



encapsulation



User Datagram Protocol

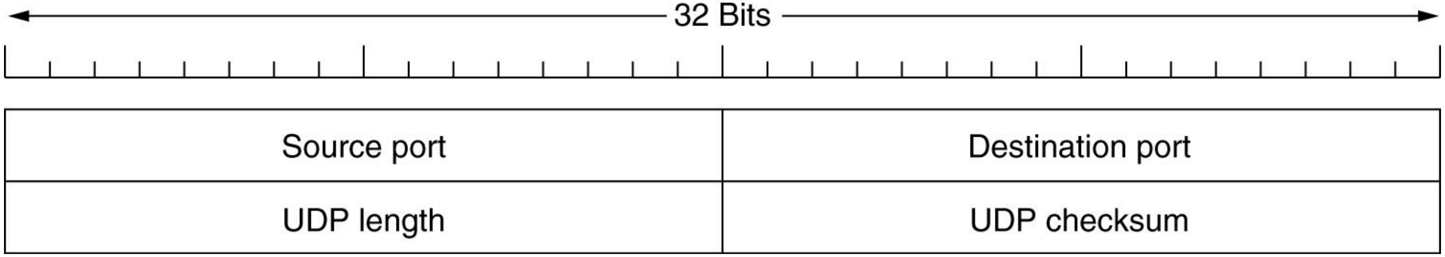
- Service provided by UDP
  - connectionless
    - no connection management
  - unreliable
    - no flow, error, congestion control
- Service provided by IP
  - connectionless, best-effort packet delivery
- Why UDP?

Why UDP?

- Sometimes TCP is an overkill
  - TCP is an all-in-one package
    - connection management
    - flow, error and congestion control
- Not all applications need TCP
  - e.g., voice over IP
    - loss tolerable to a certain degree, delay sensitive
- Why not just IP?
  - transport-layer multiplexing

UDP header

- Multiplex
  - source/destination port number
- Error checking (optional)
  - checksum (TCP/IP-style)
- Why “UDP length”?



Source port	Destination port
UDP length	UDP checksum

Internet checksum

- Mandatory in TCP
  - including TCP pseudo header
- Optional in UDP
- Also used in IP header checksum
- Checksum generation
  - 16-bit aligned, one's complement sum with carry
    - most significant carry bit wrapping around
  - “one's complement of one's complement sum”
- Checksum verification

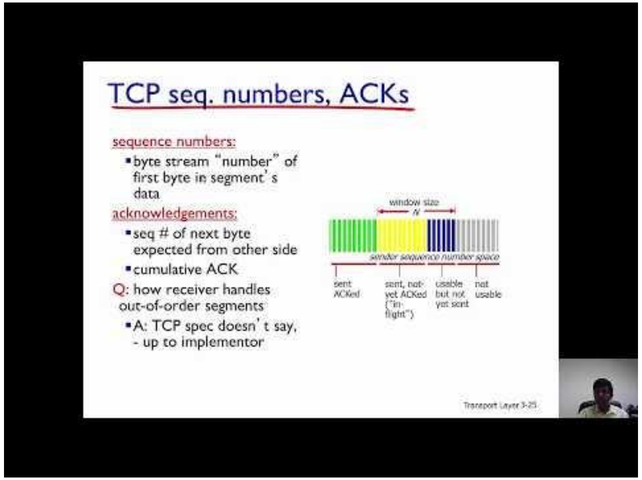
Stone, Jonathan, and Craig Partridge. "When the CRC and TCP checksum disagree." ACM SIGCOMM Computer Communication Review. Vol. 30. No. 4. ACM, 2000.

This lecture

- Transport layer services
  - addressing and multiplexing
  - connection management
  - flow, error and congestion control
- User Datagram Protocol (UDP)
  - protocol header fields
    - port number: multiplexing
  - checksum algorithms
    - checksum: error control

Next lecture

- Transmission Control Protocol (TCP)
  - connection management
  - read KR4: Computer Networking
    - Chapter 3 (all sections required this month)



TCP seq. numbers, ACKs

sequence numbers:

- byte stream "number" of first byte in segment's data

acknowledgements:

- seq # of next byte expected from other side
- cumulative ACK

Q: how receiver handles out-of-order segments

- A: TCP spec doesn't say, - up to implementor

window size
N
sender sequence number space
sent ACKed
sent, not-yet ACKed ("in-flight")
usable but not yet sent
not usable

Transport Layer 3-25