The Transport Layer

Michael Brodskiy
Professor: E. Bernal Mor

October 5, 2023

- Transport Services and Protocols
  - Provide logical communication between application processes running on different hosts
  - Transport protocols actions in end systems:
    - Sender: breaks application messages into segments, passes to Network layer
    - Receiver: reassembles segments into messages, passes to Application layer
  - Two transport protocols available to internet applications
    1. TCP
    2. UDP
- Transport vs. Network Layer
  - Network layer: logical communication between two hosts
  - Transport layer: logical communication between processes
    - Relies on, enhances, network layer services
- Two Internet Transport Protocols
  - TCP: Transmission Control Protocol
    - Reliable, in-order delivery
    - Congestion Control
    - Flow Control
    - Connection set-up
  - UDP: User Datagram Protocol
    - Unreliable, unordered delivery
    - No-frills extension of "best-effort" IP

  - Services not available:
    * Delay guarantees
    * Throughput guarantees

- Multiplexing/Demultiplexing
  - Multiplexing at sender: Handle data from multiple sockets, add transport header (later used for demultiplexing)
  - How demultiplexing works
    * Host receives IP packets
      · Each packet has source IP address, destination IP address
      · Each packet carries one transport-layer segment
      · Each segment has source, destination port number
    * Host uses IP address and port numbers to direct segment to appropriate socket
  - Connectionless Demultiplexing
    * Create a socket in the client, the Transport layer automatically assigns a host-local port number to the socket
    * When data is sent into UDP socket, must specify
      · Destination IP address
      · Destination port number
    * When a host receives UDP segment, the Transport layer:
      · Checks destination port number in segment
      · Directs UDP segment to socket with that port number
    * IP datagrams with same destination port number but different source IP addresses and/or source port numbers will be directed to same socket at destination
  - Connection-Oriented Demultiplexing
    * TCP socket identified by 4-tuple:
      · Source IP address
      · Source port number
      · Destination IP address
      · Destination port number
    * Demultiplexing receiver users all four values to direct segment to appropriate socket
    * A server may support simultaneous TCP sockets:
      · Each socket identified by its own 4-tuple
      · Each socket associated with a different connecting client

        * Note: the TCP server has a welcoming socket
            · Each time a client initiates a TCP connection to the server, a new socket is created for this connection
            · To support n simultaneous connections, the server would need $n + 1$ sockets

- Connectionless Transport: UDP
    - "No frills", "bare bones" Internet transport protocol
    - "Best effort" service, UDP segments may be:
        * Lost
        * Delivered out-of-order to application
    - Connectionless:
        * No handshaking between UDP sender, receiver
        * Each UDP segment handled independently of others
    - Why is there a UDP?
        * No connection establishment (which can add RTT delay)
        * Simple: no connection state at sender, receiver
        * Small header size
        * No congestion control
            · UDP can blast away as fast as desired
            · It can function in the face of congestion
    - UDP used in:
        * Streaming multimedia apps (loss tolerant, rate sensitive)
        * DNS
        * HTTP/3
    - If reliable transfer or other services needed over UDP (like in HTTP/3)
        * Add needed reliability at application layer
        * Add congestion control at application layer