Computer Networking

COMP 177 | Fall 2020 | University of the Pacific | Jeff Shafer

UDP

User Datagram Protocol

Recap

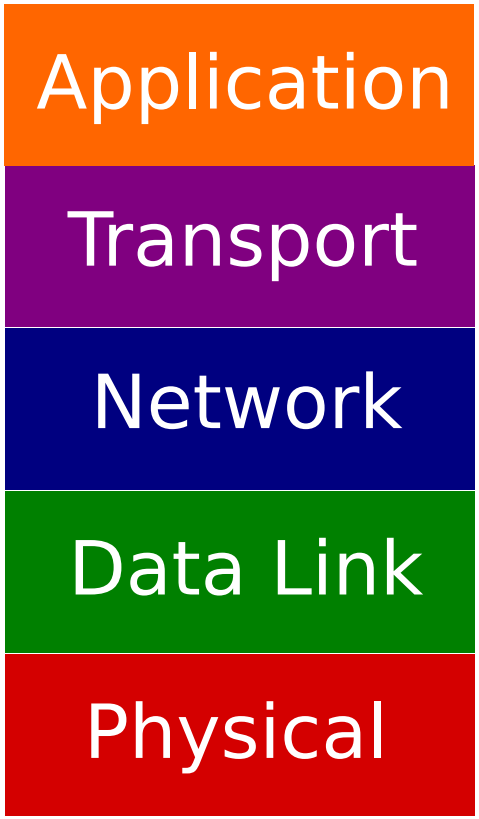
Past Topics

- Overview of networking and layered architecture
- Wireshark packet sniffer and Scapy packet manipulation
- Wired LAN, Wireless LANs, VLANs
- IPv4, IPv6 ARP, ICMP
- DHCP

Today's Topics

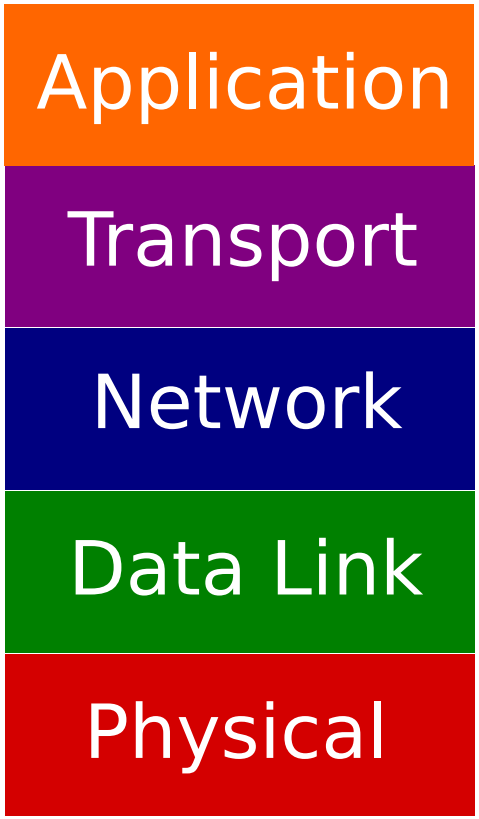
- Transport layer protocols: UDP

Layers



- Application Layer
- Transport Layer
- *Network Layer – "End to End Packet Transfer"*
  - *Global network: IPv4, IPv6*
- *Data Link – "Transfer between Neighbors"*
  - *LAN: Ethernet, WiFi, …*
- *Physical layer – "Bits on a Wire"*

Transport Layer



- The *Network layer* is used to route packets through the network of routers to the ultimate destination machine (IPv4, IPv6)
- The *Transport layer* uses the service of the network layer to provide end-to-end communication between **two processes** on **two (remote) hosts**.
- Unlike protocols that we have studied so far, transport layer (and application layer) protocols are only used in *hosts*
  - Network layer protocols are used in routers and hosts
  - Link layer protocols are used in switches, routers, and hosts

Transport Layer Ports

- Each process that is accessible through the network is identified by a *port number*
  - *Transport layer* ports are different than *physical* ports for network interfaces on switches, routers and host machines.
- Transport layer ports are assigned to processes by the operating system
- Port numbers can be used to refer to processes within a machine
- To identify a remote process, two pieces of information are needed:
  - IP address of the remote machine
  - Port number associated with the process

Transport Layer Services

- A transport layer service may provide the following to applications:
- **Reliability**: Reliable data transfer consists of three components:
  - Protection against data loss
  - Protection against out-of-order transmission and receipt of packets
  - Protection against bit errors (bit flips)
- **Connection orientation**: has two phases
  - *Before* two processes start to communicate, transport layer may establish a connection between the processes
  - *After* two processes communicate, transport layer may tear down the connection between the processes
- **Stream orientation**: Transport layer may treat application data as a *stream of bytes*, divided into appropriate-size packets.

Transport Layer Services

- A transport layer service may provide the following to applications:
- **Port numbers**: used to identify the application instances (processes) at the two ends, at runtime
- **Congestion control**: mechanisms to avoid flooding intermediary nodes (between the two end systems) with packets during the communication
- **Flow control**: Transport layer service may provide mechanisms to avoid flooding the other end host, with packets, during the communication.

User Datagram Protocol (UDP)

- User Datagram Protocol (UDP) is one the two prominent transport layer protocols in the Internet
  - The other protocol is TCP
- UDP is an "almost null" protocol:
  - Provides a very basic service model to the applications
  - Everything from IP layer (best effort), plus
    - Identification of bit errors
    - Process identification by port numbers
- UDP is a best-effort protocol
- *In contrast, TCP provides all of the discussed services!*

UDP Applications

- Numerous applications use UDP rather than TCP, despite its limited service model. Why?
  - UDP is light and fast!
- Two major use cases for UDP:
  - Application communicates *small messages* in an *infrequent manner*
    - The application itself can handle unreliable data transmission by retransmitting the packets
  - Application is *loss-tolerant* and can handle packet loss (to some extent)
    - Video and audio streaming applications can tolerate a few glitches!
    - In general, UDP is appropriate for real-time transport, e.g., audio, video, real-time games

UDP Unreliability

- UDP does not provide a reliable mechanism to transport packets from one process to another
- If an application uses UDP as its transport layer protocol
  - There is no guarantee that packets of that application will be delivered to the destination process
  - There is no guarantee that the packets will be delivered to the destination process *in the same order* that the sender process has sent them
  - There is no guarantee that the bits of the packets will be correctly transmitted to the destination process.
- For the first two challenges, there are no mechanisms in UDP
- For the final challenge there is a rather weak mechanism that is usually disabled

UDP **Un**reliability

- In reliable data transfer,
  - When a packet is lost, the following packets that are received should be buffered temporarily
  - Buffering gives time for the lost packet to be identified by the sender and retransmitted.
  - Upon receiving the lost packet, the destination transport layer builds up the full message and hands it to the upper layer application
  - This process is time consuming!
- UDP does **not** buffer packets
  - As soon as a packet arrives, the payload is sent to the upper layer application
  - Good for *loss-tolerant* and *delay-intolerant* applications!

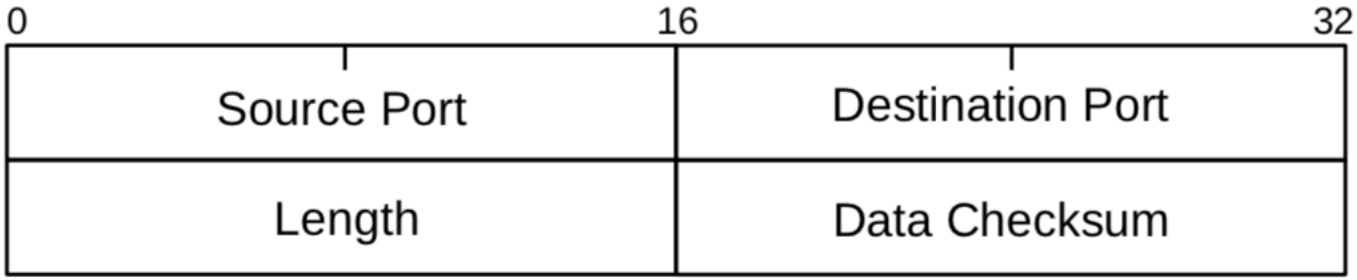
UDP Connectionless Service

- UDP is **not** a connection-oriented service
  - Connectionless service
  - UDP does not establish a connection between the two end processes before those processes begin to transmit data
  - UDP does not close a connection between the two end processes after those processes finish transmitting data to each other
- As long as a process has opened a UDP port on a host, any process can send/receive packets from/to that process, without negotiation

Congestion Control and Flow Control

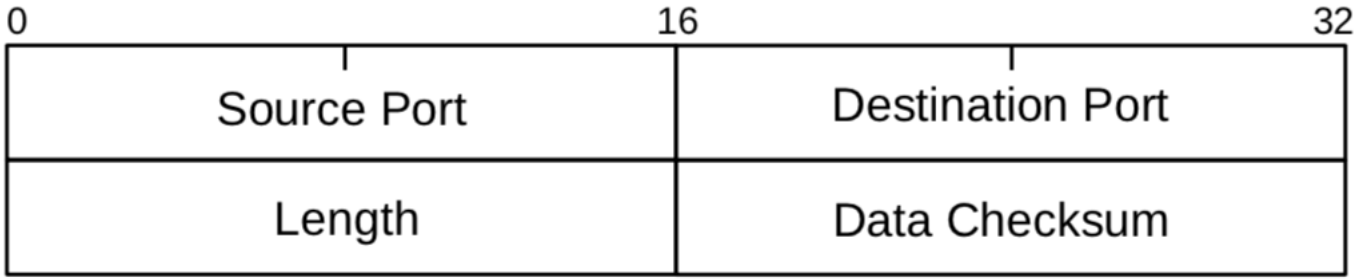
- UDP does not provide *congestion control*
  - UDP packets can be dropped due to buffer overflows in the intervening routers in the path
- UDP does not provide *flow control*
  - UDP packets can be dropped due to buffer overflow at the destination host
- UDP cannot identify when packets are dropped (whether in a router or the final host) due to unreliable data transmission service model

UDP Header Format



- **Source Port** (16 bits): The port number assigned to the sender process in the sender host
- **Destination Port** (16 bits): The port number assigned to the receiver process in the destination host
  - There are 2^{16} = 65536 ports available on the source and destination hosts, i.e., port numbers 0 to 65535
  - In other words, theoretically 65536 simultaneous processes can be executed on a given host

UDP Header Format



- **Length** (16 bits): Size of the full UDP datagram in bytes
  - 65535 bytes (64KB) is the max size for a UDP packet
- **Data checksum** (16 bits): Internet checksum used for the identification of bit flips in the UDP datagram (header + payload)
  - Same algorithm used in IPv4 to identify bit errors in the IP header
  - Rather weak mechanism to identify bit errors. There could be numerous cases where bits are flipped but are not identifiable by checksum
  - Due to its weakness, this field is frequently disabled by setting the field to 0x0000.

Sockets

- Socket: Sockets are data structures that the operating system uses to handle the communication between
  - The application instances at runtime, and
  - The underlying transport layer service (implemented in the OS)
- The OS provides an API ("socket API") to the user-level applications
- User-level applications use socket API to establish connection to (potentially remote) processes
- Socket API provides different functions for applications to talk to the underlying transport layer service, therefore
  - Different transport layer protocols have different socket APIs
  - UDP sockets are (modestly) different from TCP sockets

Closing Thoughts

Recap

- Today we discussed
  - Transport layer services
  - UDP service model
  - UDP header format

Next Class

- Socket Programming!

Class Activity

CA.12 – UDP & Wireshark

Due tonight at 11:59pm

Homework 3

Due October 14th