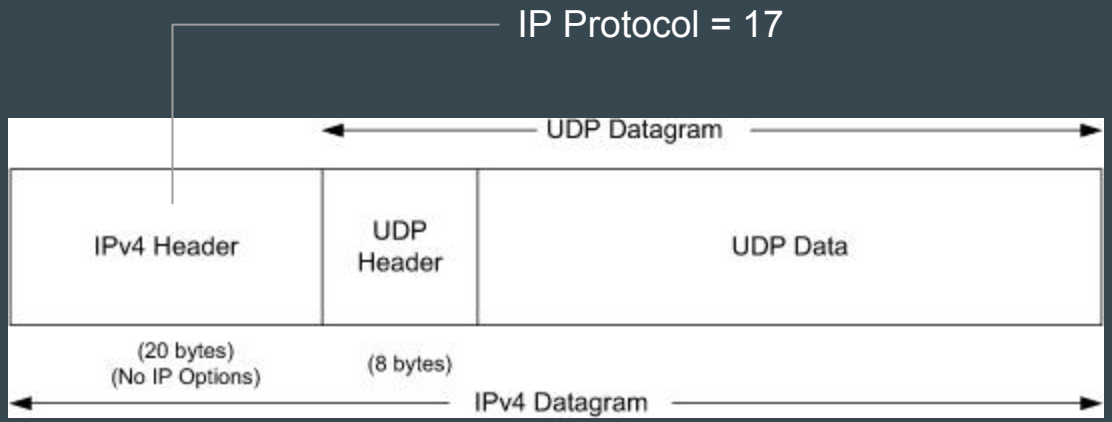
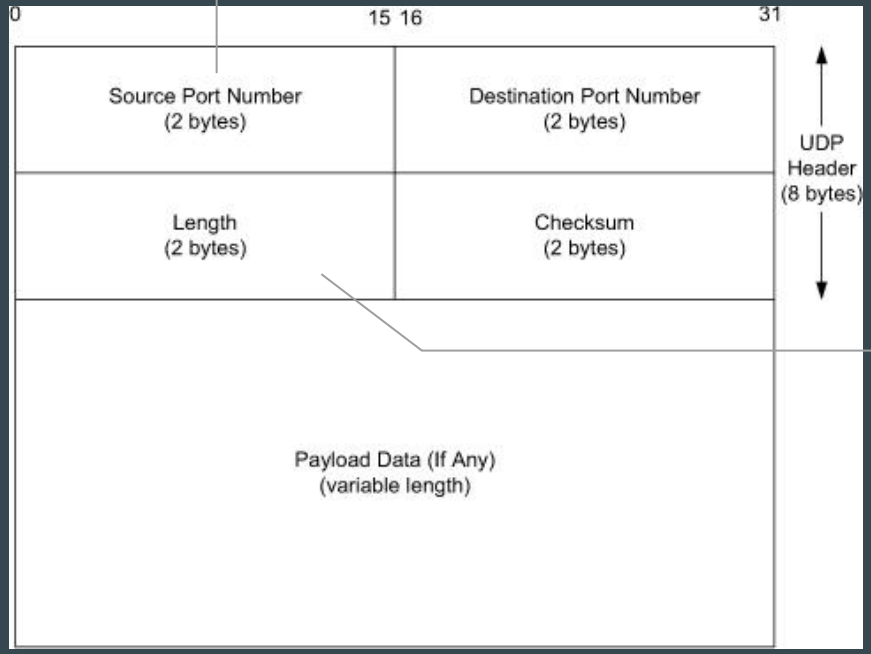
# UDP and Related Things



# UDP Characteristics

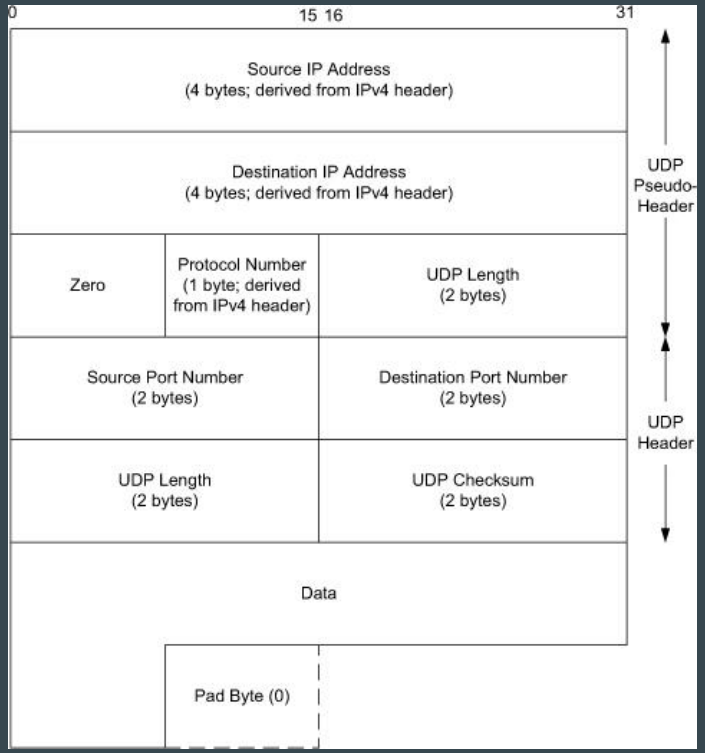
- Is a transport-layer protocol
  - End-to-end
- Connectionless, best-effort, no error correction, ...
  - Gives application control




- Allowed to be 0
- Receiver demuxes based on ⟨destination-ip, destination-port⟩ only.

- Units = bytes
- Length of header + data ⇒ min = 8
- Redundant if encapsulated in IP

# UDP Checksum

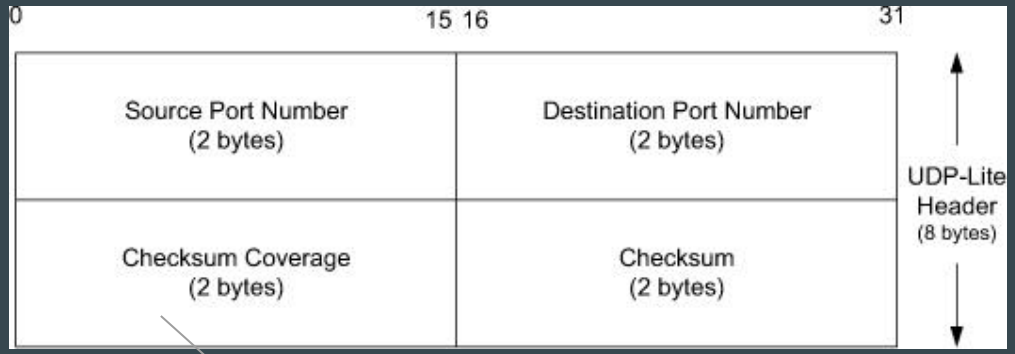



Scope of checksum

# UDP Checksum - some details

- 1's complement of 1's-complement addition
- Pseudo-header used to guard against mistakes by IP layer.
- Checksum field of 0's used in checksum computation
- Padded with 0-byte for even # of bytes; not transmitted
- In UDP, checksum is optional in IPv4 - indicated with checksum = 00 00
  - To distinguish from legitimate checksum of all 0's, transmit ff ff
  - UDP checksum required in IPv6
- Checksum is used end-to-end, not hop-by-hop

# UDP-lite




- 0 ⇒ entire UDP-lite datagram
- 1 - 7 - illegal values, header always covered

Protocol value in IP header = 136. So a different transport-layer protocol than UDP (17).

# Path MTU discovery with UDP

- UDP port 7 = echo service
- To test if MTU of, e.g., 1505, is supported, send UDP data
  - of 1477
  - to ⟨destination-ip, destination-port = 7⟩
  - Set DF (Do not Fragment) bit in IP header
- Response ⇒ supported
  - Is the converse necessarily true?

# Programming quirk

- *sendto(), recvfrom()*
- Not required to support particular datagram size
- Some implementations adopt 512 as max
  - Every IP host required to support MTU ≥ 576
- Other issues may result in fewer bytes in a single *recvfrom()* call

# UDP server design considerations

- Can *bind()* to “wildcard” local address
  - INADDR_ANY
- Server may want to know at which local address datagram was received
  - *getsockname()*
- Multiple servers can bind to same ⟨destination-ip, destination-port⟩
  - SO_REUSEADDR

# UDP traffic on the Internet

- No reliable statistics
- 10-40% of Internet traffic is UDP
  - RTP - Real Time Protocol
- % of fragments that are UDP is high: ≃ 65%
  - Overall only < 1% of Internet traffic is fragments
  - Why?
    - Packets in packets in packets
    - “Carelessness”

# Example applications

- DHCP
  - Initial requests have
    - source-ip = 0.0.0.0, source-port = 68
    - Destination-ip = 255.255.255.255, destination-port = 67
  - Response is unicast
- DNS
  - Used to resolve “names” to “numbers”
  - E.g., what is the IP address of ecelinux4.uwaterloo.ca
  - E.g., what is the mailserver associated with uwaterloo.ca?
  - E.g., what is the webserver associated with uwaterloo.ca?

# Aside - IP fragmentation + ARP

- Your book discusses an interesting experiment:
  - Send an IP packet to destination that you know will be fragmented, e.g., into 4
  - How many ARP requests are generated?
    - Maybe up to 4
  - Problem (RFC 1122):
    *The link layer SHOULD save (rather than discard) at least one (the latest) packet of each set of packets destined to the same unresolved IP address, and transmit the saved packet when the address has been resolved.*

- Not just a problem when we have fragmentation.