Chapter 3: roadmap

- Transport-layer services
- Multiplexing and demultiplexing
- Connectionless transport: UDP
- Principles of reliable data transfer
- Connection-oriented transport: TCP
- Principles of congestion control
- TCP congestion control
- Evolution of transport-layer functionality

UDP: User Datagram Protocol

- “no frills,” “bare bones” Internet transport protocol
- “best effort” service, UDP segments may be:
  - lost
  - delivered out-of-order to app
- *connectionless:*
  - no handshaking between (UDP) sender and receiver
  - each UDP segment is handled independently of others

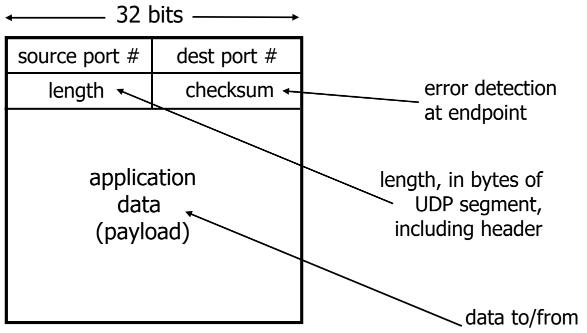
Why is there a UDP?

- no connection establishment
  - which can add RTT delay
- simple (no connection state at endpoint)
  - server can support more active clients when running over UDP
- small header overhead
- finer application-layer control
  - no congestion control
  - UDP can blast away as fast as desired!

UDP: User Datagram Protocol

- UDP was ratified in 1980. [RFC 768]
- UDP is used by:
  - streaming multimedia apps (loss tolerant, rate sensitive)
  - DNS
  - SNMP
  - HTTP/3 (over QUIC over UDP)
- if reliable transfer needed over UDP (e.g., HTTP/3):
  - add needed reliability at application layer
  - add congestion control at application layer

UDP segment header



32 bits

source port #	dest port #
length	checksum

application data (payload)

error detection at endpoint

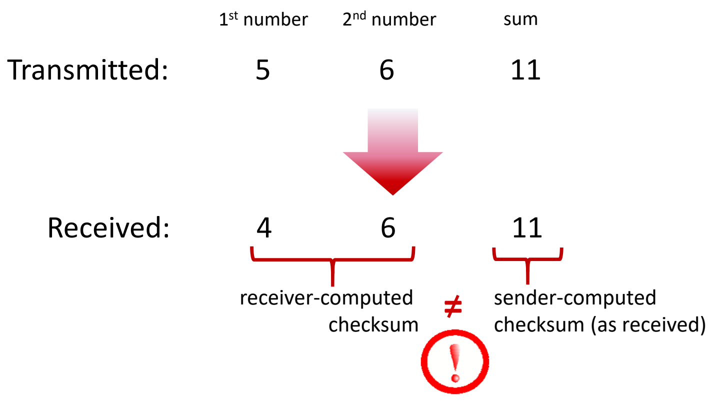
length, in bytes of UDP segment, including header

data to/from application layer

UDP segment format

UDP checksum

Goal: detect errors (*i.e.,* flipped bits) in transmitted segment



1st number
2nd number
sum
Transmitted:
5
6
11
Received:
4
6
11
receiver-computed checksum
≠
sender-computed checksum (as received)
!

UDP checksum

Goal: detect errors (*i.e.,* flipped bits) in transmitted segment

sender:

- treat contents of UDP segment as sequence of 16-bit integers
  - including pseudo header (IP addresses, ...)
- checksum:
  - add segment content (1's complement sum)
  - and then take 1's complement
- checksum value put into UDP checksum field

receiver:

- compute checksum of received segment
- check whether computed checksum equals checksum field value:
  - Not equal – error detected
  - Equal – no error detected. *But maybe errors nonetheless?* More later ....

Internet checksum: an example

example: add two 16-bit integers

```
                1 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0
                1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
              ---------------------------------
wraparound    1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1
              ---------------------------------
       sum      1 0 1 1 1 0 1 1 1 0 1 1 1 1 0 0
  checksum      0 1 0 0 0 1 0 0 0 1 0 0 0 0 1 1
```

Note: when adding numbers, a carryout from the most significant bit needs to be added to the result

* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

Internet checksum: weak protection!

example: add two 16-bit integers

```
                                                  0 1
                1 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0   1 0
                1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
              ---------------------------------
wraparound    1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1
              ---------------------------------
       sum      1 0 1 1 1 0 1 1 1 0 1 1 1 1 0 0
  checksum      0 1 0 0 0 1 0 0 0 1 0 0 0 0 1 1
```

Even though numbers have changed (bit flips), *no* change in checksum!

Summary: UDP

- "no frills" protocol:
  - segments may be lost, delivered out of order
  - best effort service: "send and hope for the best"
- UDP has its pluses:
  - no setup/handshaking needed (no RTT incurred)
  - smaller header overhead
  - finer application-layer control
  - helps with reliability (checksum)
- If needed, additional functionality can be built on top of UDP in application layer (e.g., HTTP/3)