

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, 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!
 - can function in the face of congestion

UDP: User Datagram Protocol

- UDP use:
 - streaming multimedia apps (loss tolerant, rate sensitive)
 - DNS
 - SNMP
 - HTTP/3
- if reliable transfer needed over UDP (e.g., HTTP/3):
 - add needed reliability at application layer
 - add congestion control at application layer

UDP: User Datagram Protocol [RFC 768]

INTERNET STANDARD

RFC 768

J. Postel

ISI

28 August 1980

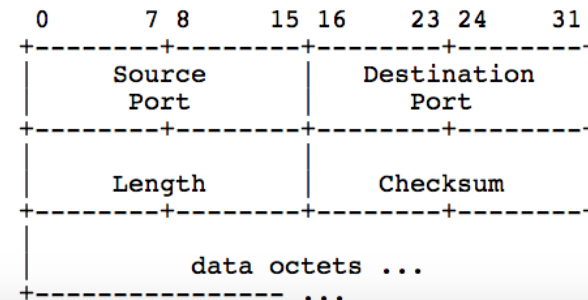
User Datagram Protocol

Introduction

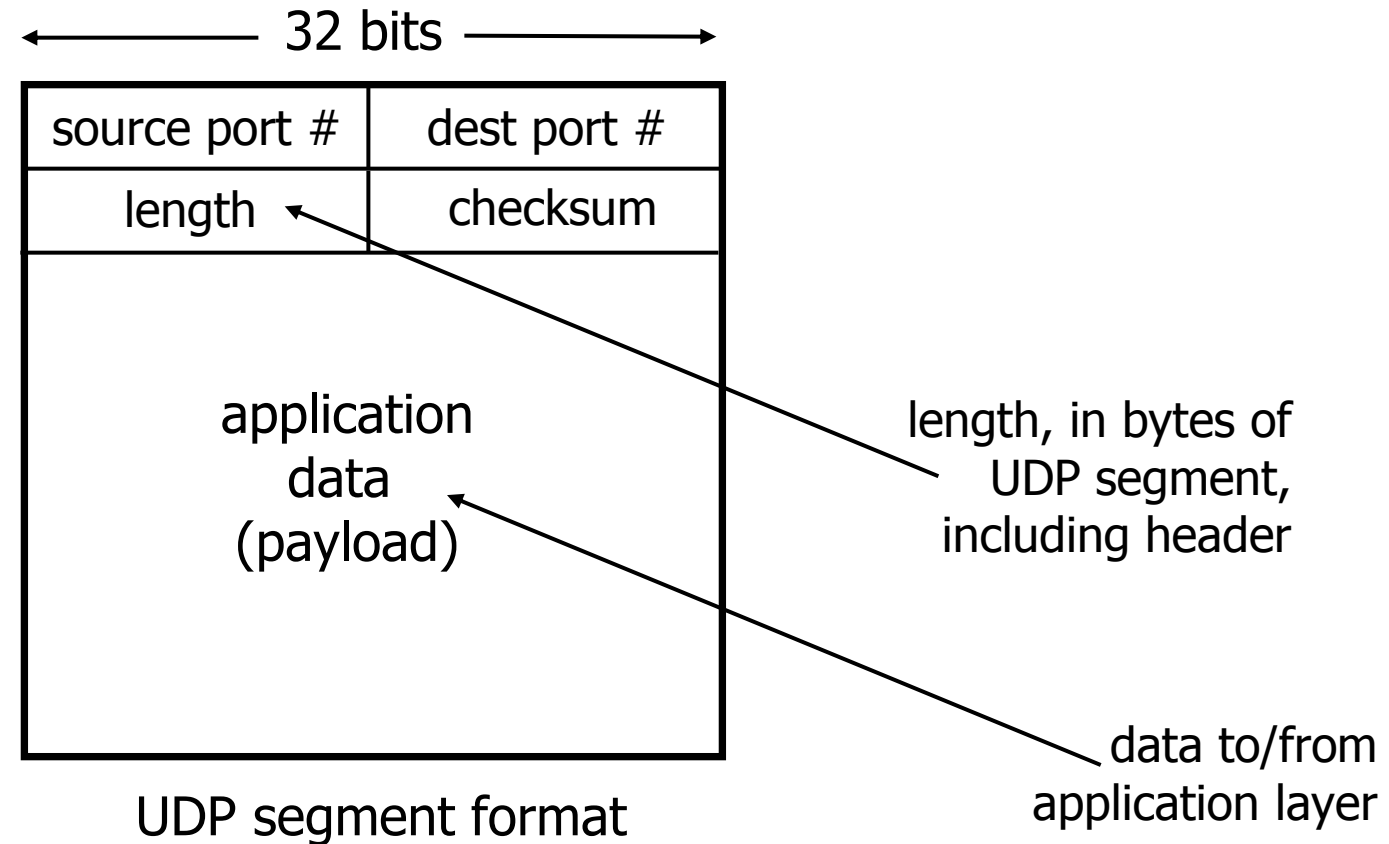
This User Datagram Protocol (UDP) is defined to make available a datagram mode of packet-switched computer communication in the environment of an interconnected set of computer networks. This protocol assumes that the Internet Protocol (IP) [1] is used as the underlying protocol.

This protocol provides a procedure for application programs to send messages to other programs with a minimum of protocol mechanism. The protocol is transaction oriented, and delivery and duplicate protection are not guaranteed. Applications requiring ordered reliable delivery of streams of data should use the Transmission Control Protocol (TCP) [2].

Format

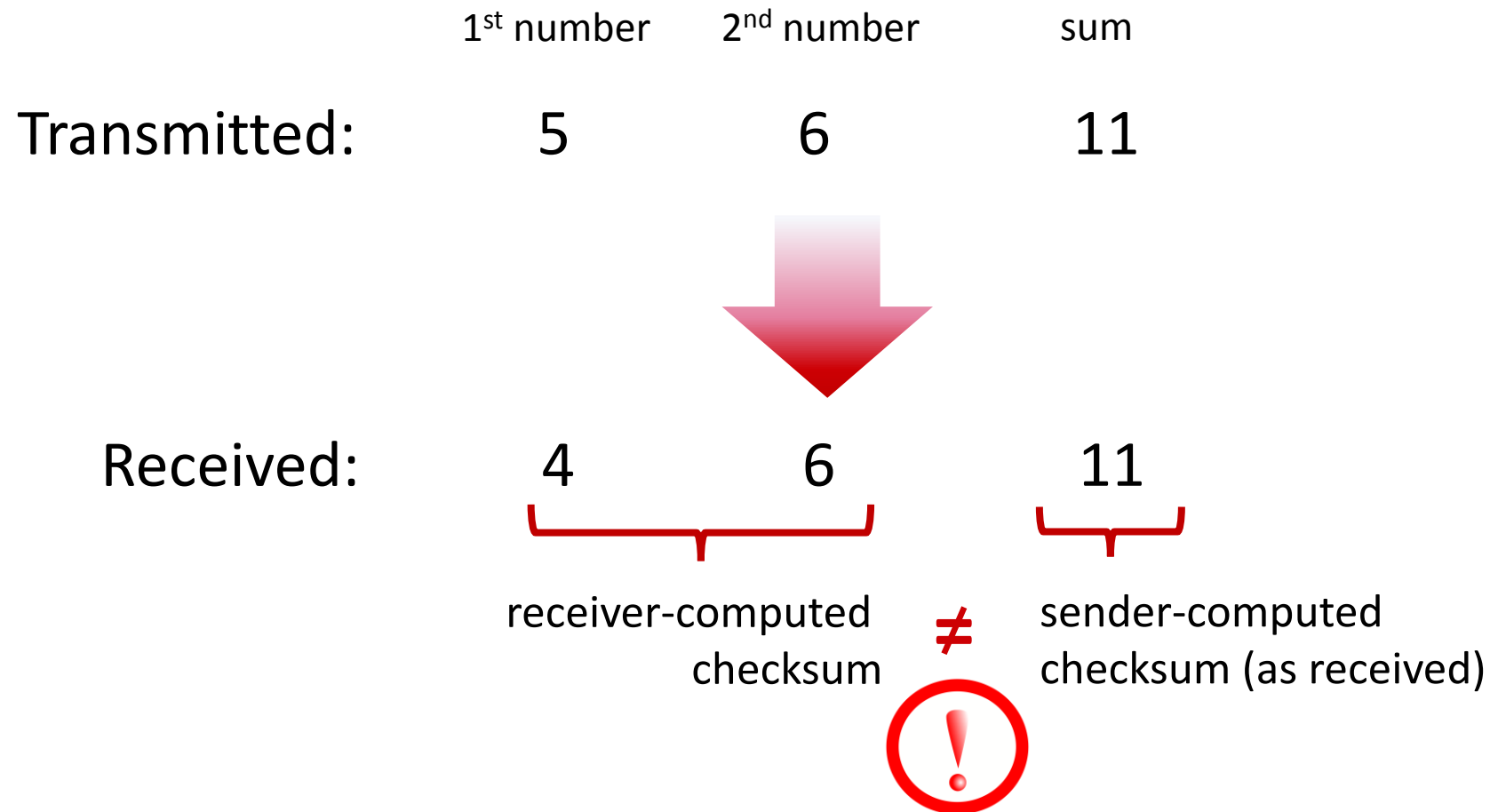


UDP segment header



UDP checksum

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



UDP checksum

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

sender:

- treat contents of UDP segment (including UDP header fields and IP addresses) as sequence of 16-bit integers
- **checksum:** addition (one's complement sum) of segment content
- checksum value put into UDP checksum field

receiver:

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

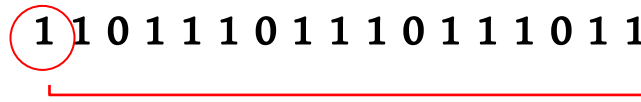
Internet checksum: an example

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

A red circle highlights the first '1' in the 'wraparound' row. A red arrow points from this circle to the 15th bit (the second-to-last bit) of the 'checksum' row.

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

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

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

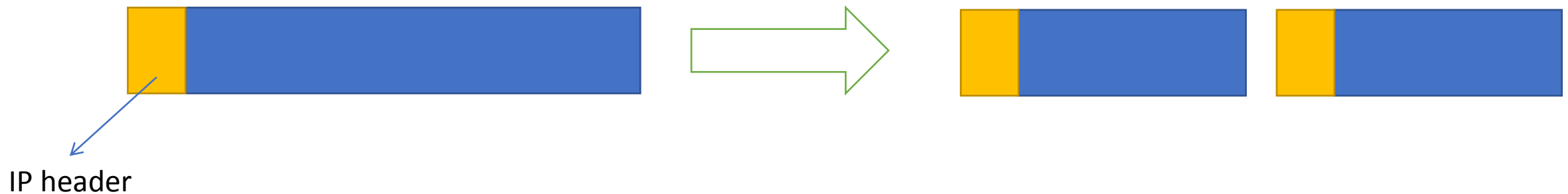
UDP & Fragmentation

- Fragmentation

- An IP (layer 3) functionality
- Breaking an IP packet into smaller ones in order to fit into link layer frames
- MTU: maximum transmission unit

In Ethernet: 1500 Byte

- Fragmentation can happen anywhere in the path. Reassembly only at receiver's network layer



UDP & Fragmentation

- Fragmentation
 - Must be avoided as much as possible
 - No recovery procedure: if one fragment is missed, no try is made for retransmission
 - It is easy in UDP to cause fragmentation!
 - TCP tries to avoid fragmentation (we will see later)

