Transport Layer

Anand Baswade

anand@iitbhilai.ac.in

Chapter 3: roadmap

- Transport-layer services
- **Connectionless transport: UDP**
- Connection-oriented transport: TCP
  - segment structure
  - reliable data transfer
  - flow control
  - connection management
- Principles of congestion control
- TCP congestion control

UDP: User Datagram Protocol

- Simple and quick 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

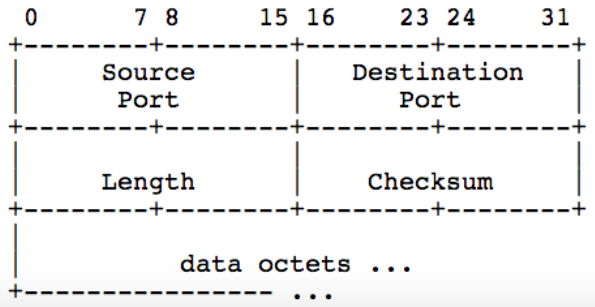
User Datagram Protocol

Introduction

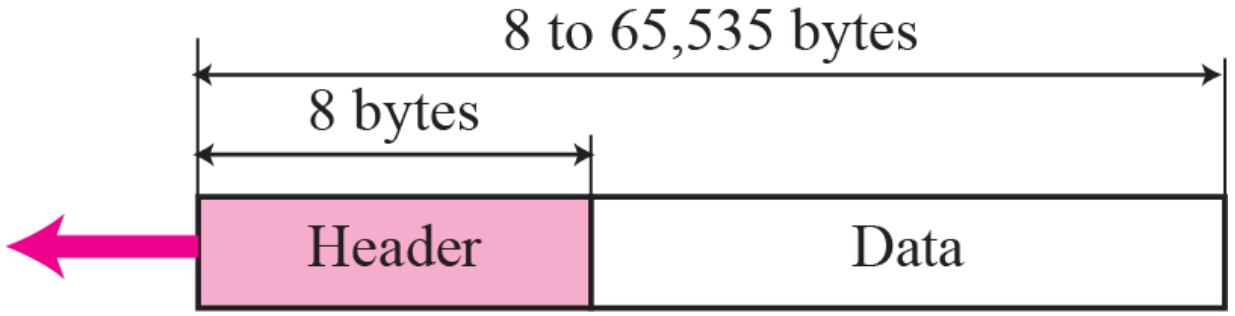
This User Datagram Protocol (UDP) is defined to make available a datagram mode of packet-switched computer communication in the environment of an interconnected set of computer networks. This protocol assumes that the Internet Protocol (IP) [1] is used as the underlying protocol.

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].

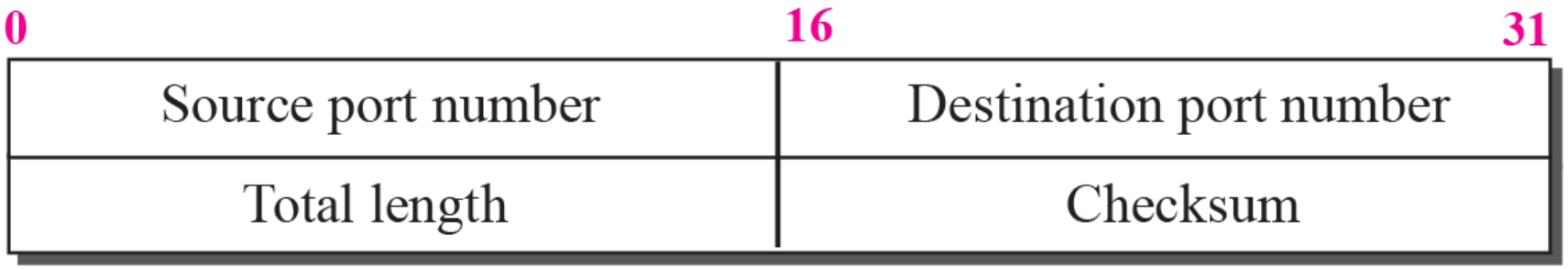
Format



UDP Header



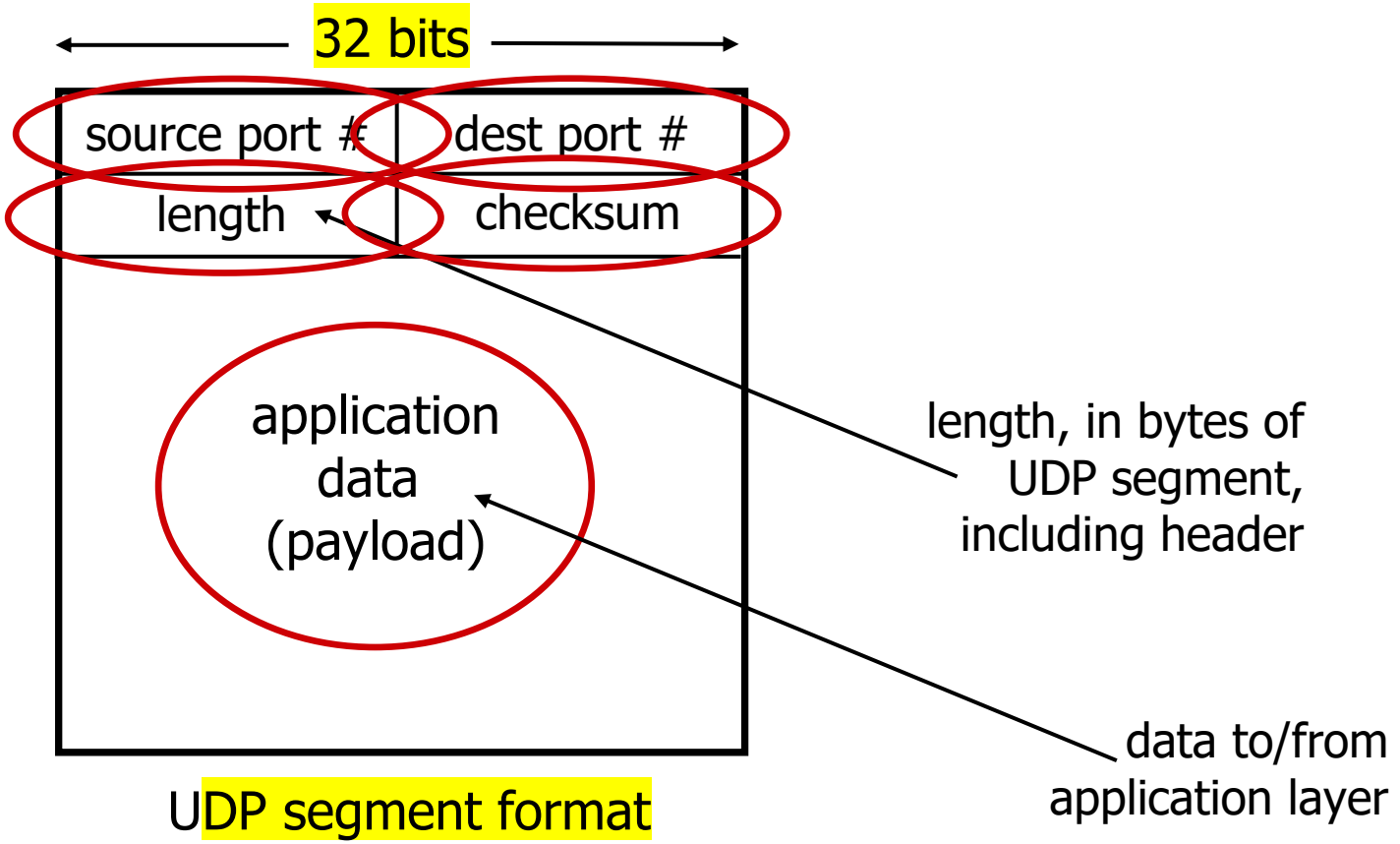
a. UDP user datagram



b. Header format

Source: TCP/IP Protocol Suite

UDP segment header cont..



UDP segment format

Questions

The following is a dump of a UDP header in hexadecimal format.

CB84000D001C001C

a. What is the source port number?
b. What is the destination port number?
c. What is the total length of the user datagram?
d. What is the length of the data?
e. Is the packet directed from a client to a server or vice versa?

Answers

Solution

a. The source port number is the first four hexadecimal digits $(CB84)_{16}$ or 52100.

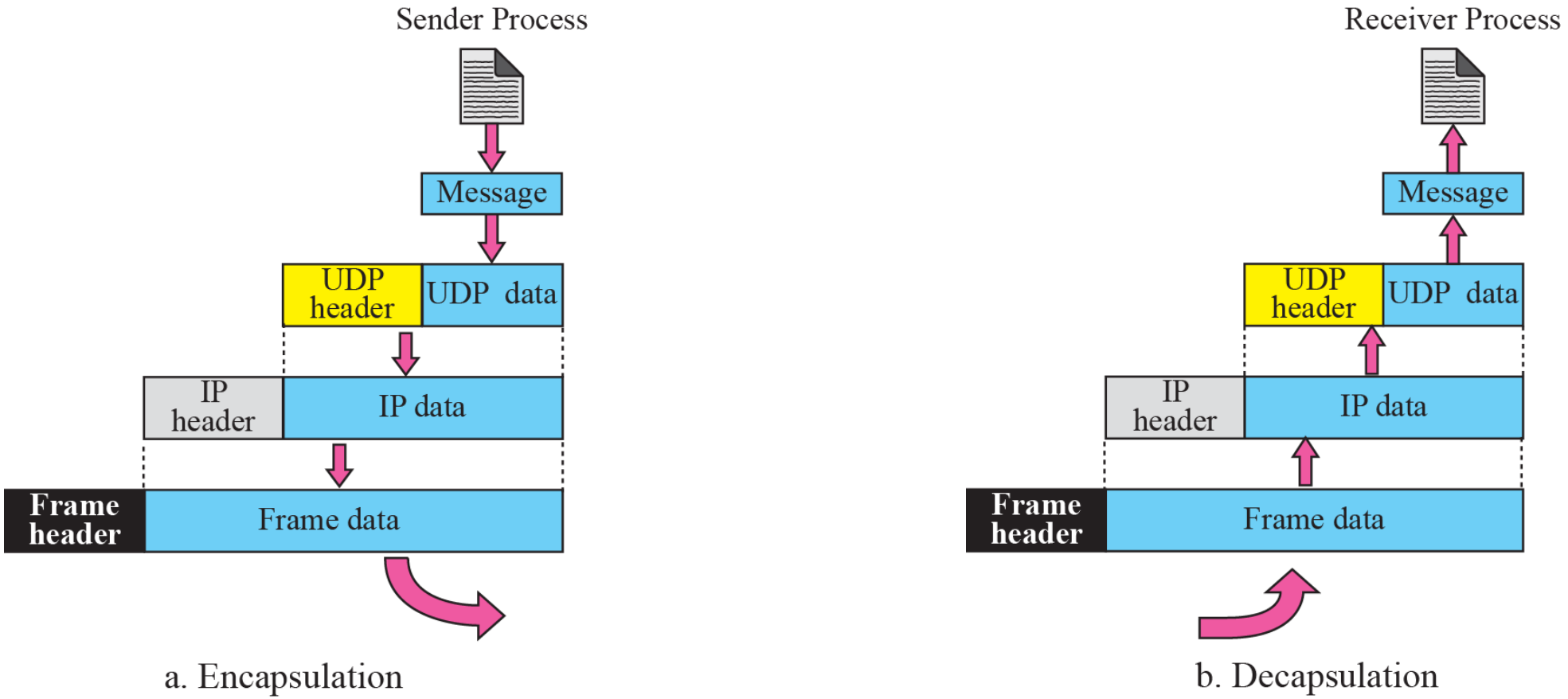
b. The destination port number is the second four hexadecimal digits $(000D)_{16}$ or 13.

c. The third four hexadecimal digits $(001C)_{16}$ define the length of the whole UDP packet as 28 bytes.

d. The length of the data is the length of the whole packet minus the length of the header, or 28 – 8 = 20 bytes.

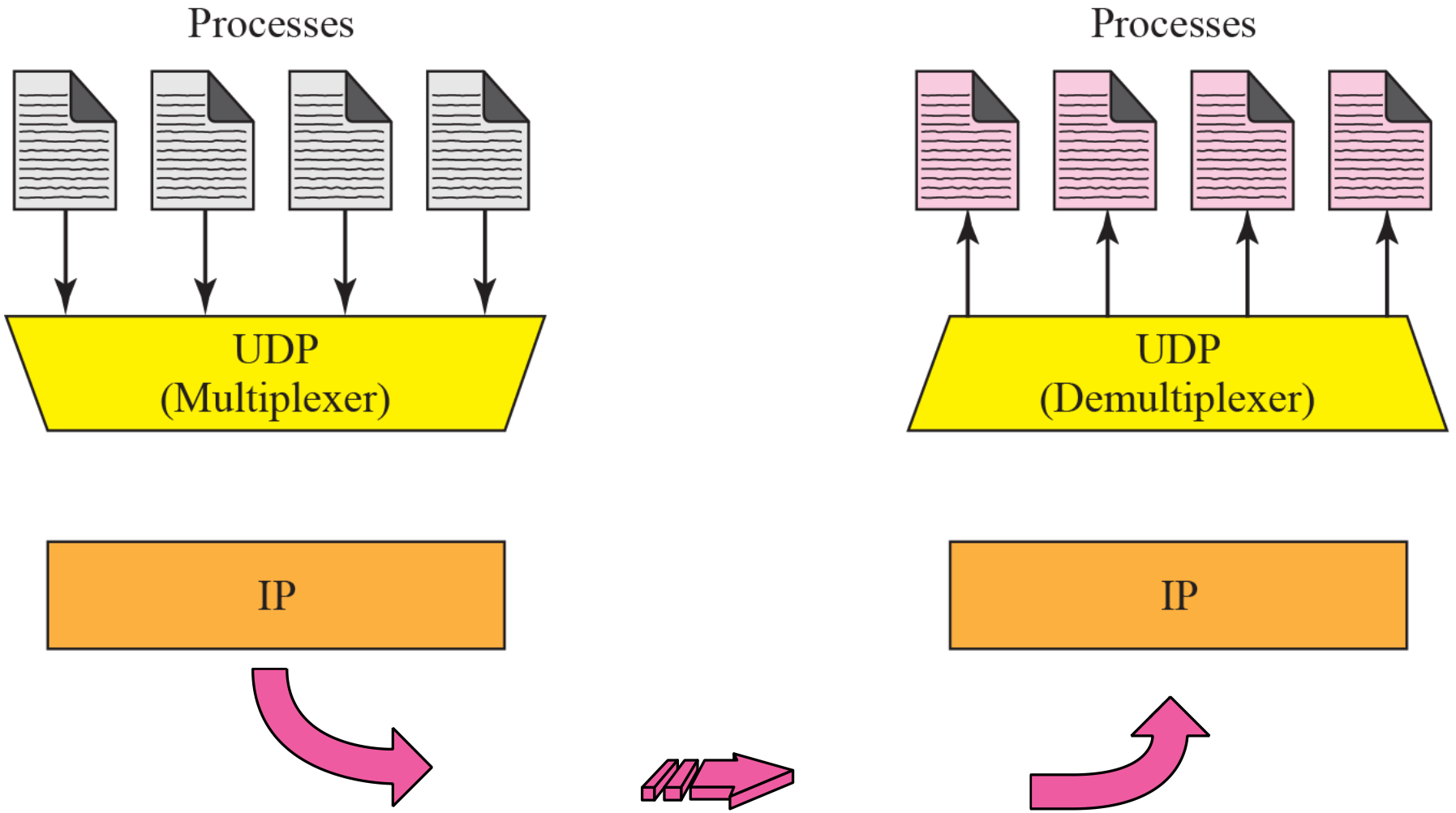
e. Since the destination port number is 13 (well-known port), the packet is from the client to the server.

Encapsulation and decapsulation



Source: TCP/IP Protocol Suite

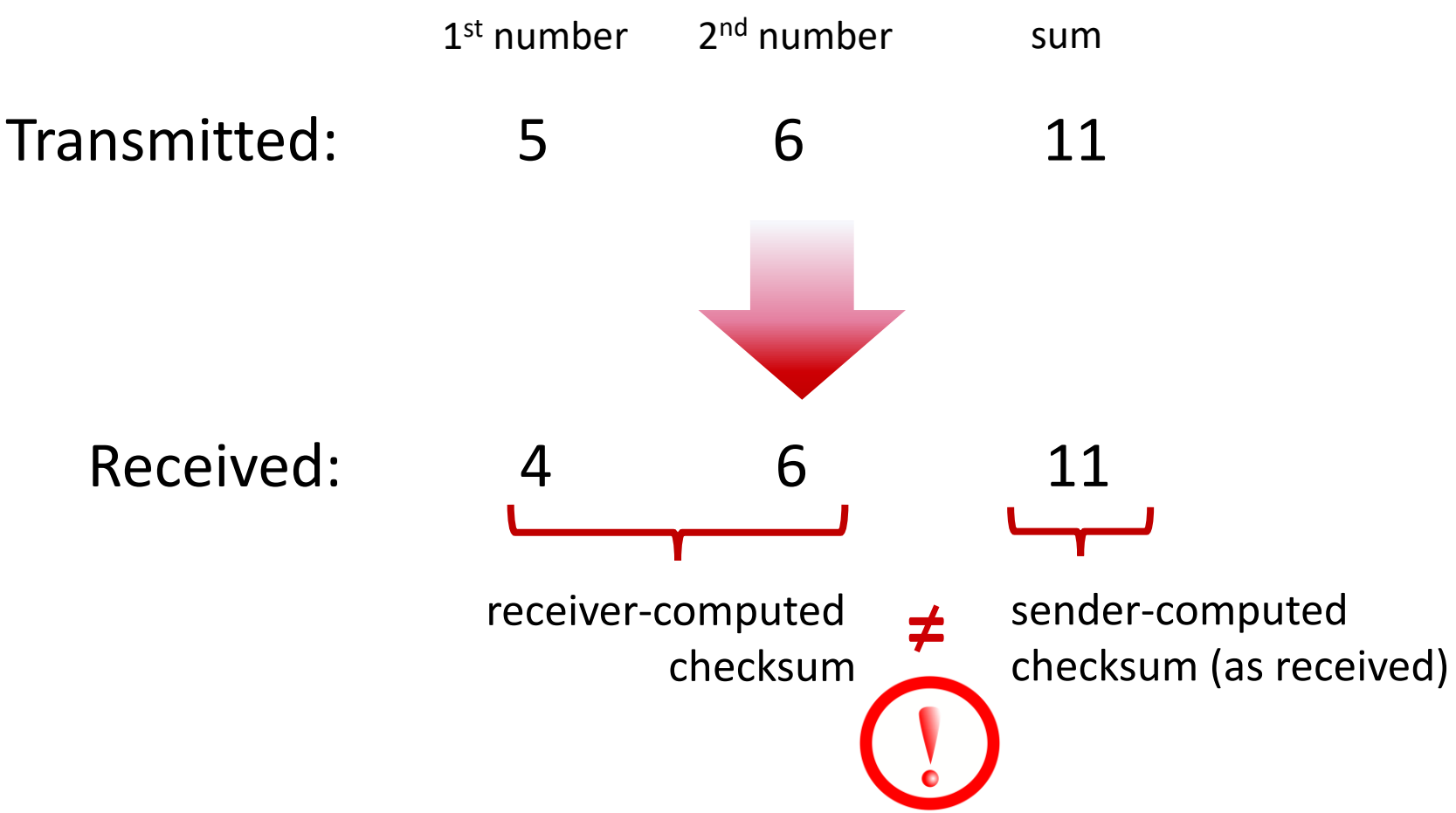
Multiplexing and demultiplexing



SOURCE: TCP/IP Protocol Suite

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:

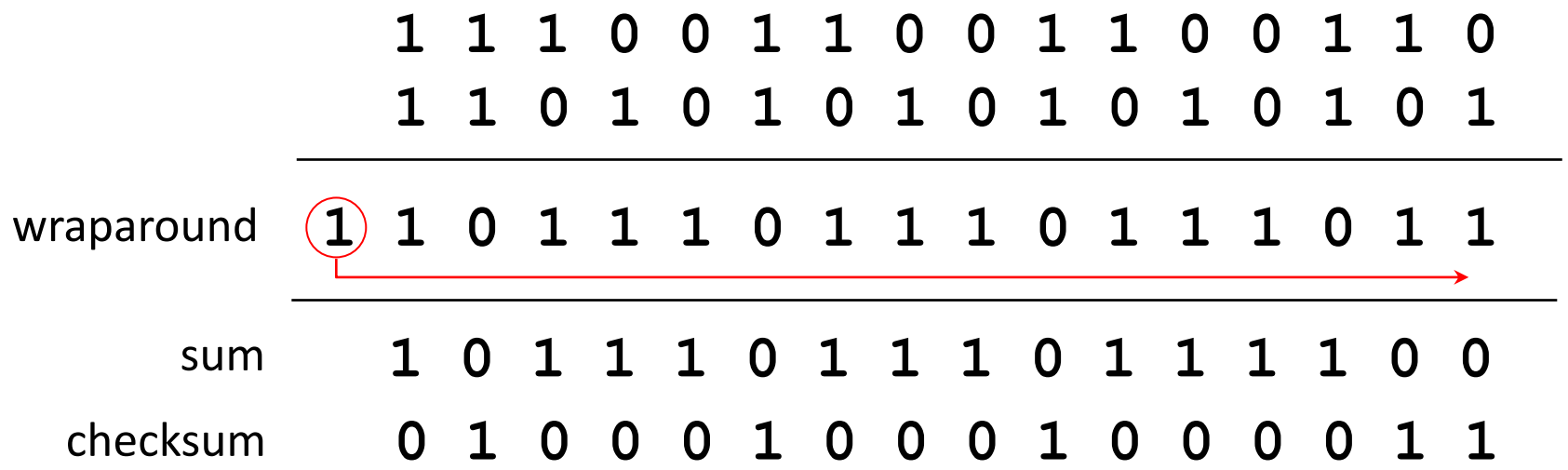
- treat contents of UDP segment (including UDP header fields and IP addresses) as sequence of 16-bit integers
- checksum: addition (one's complement sum) of segment content
- checksum value put into UDP checksum field

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

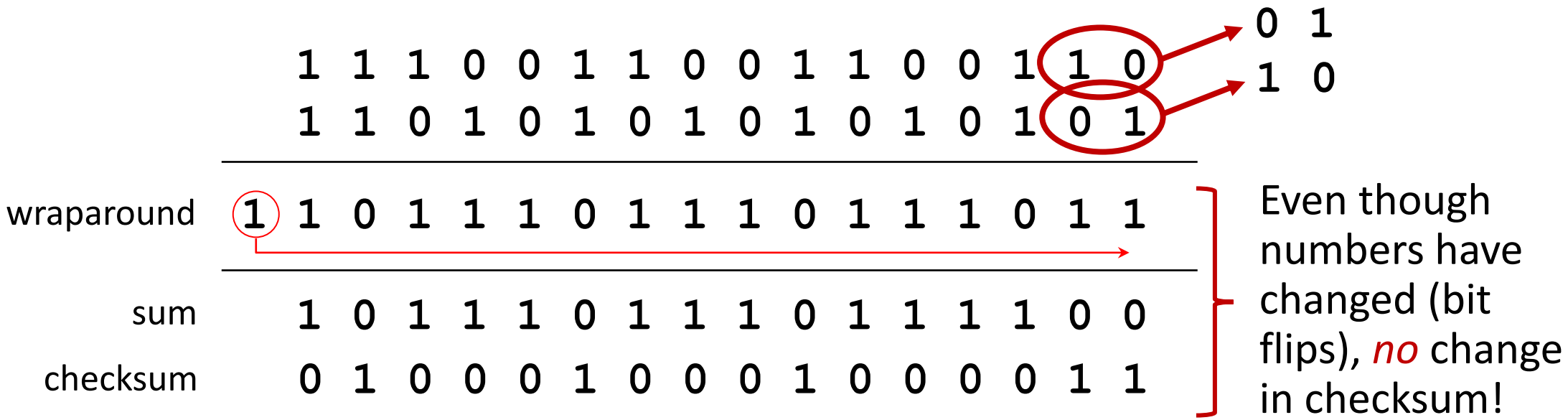


Note: when adding numbers, a carryout from the most significant bit needs to be added to the result

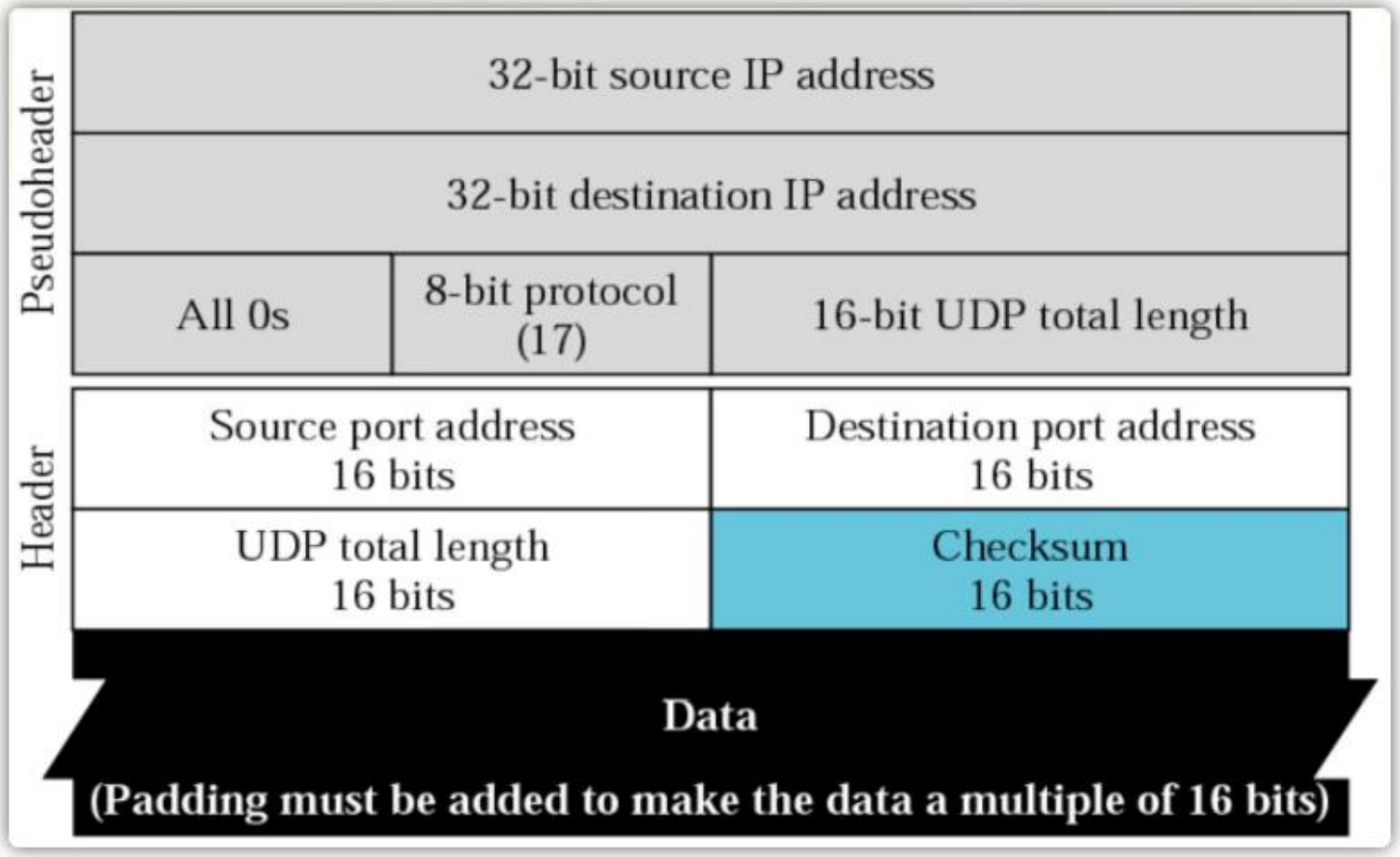
* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

Internet checksum: weak protection!

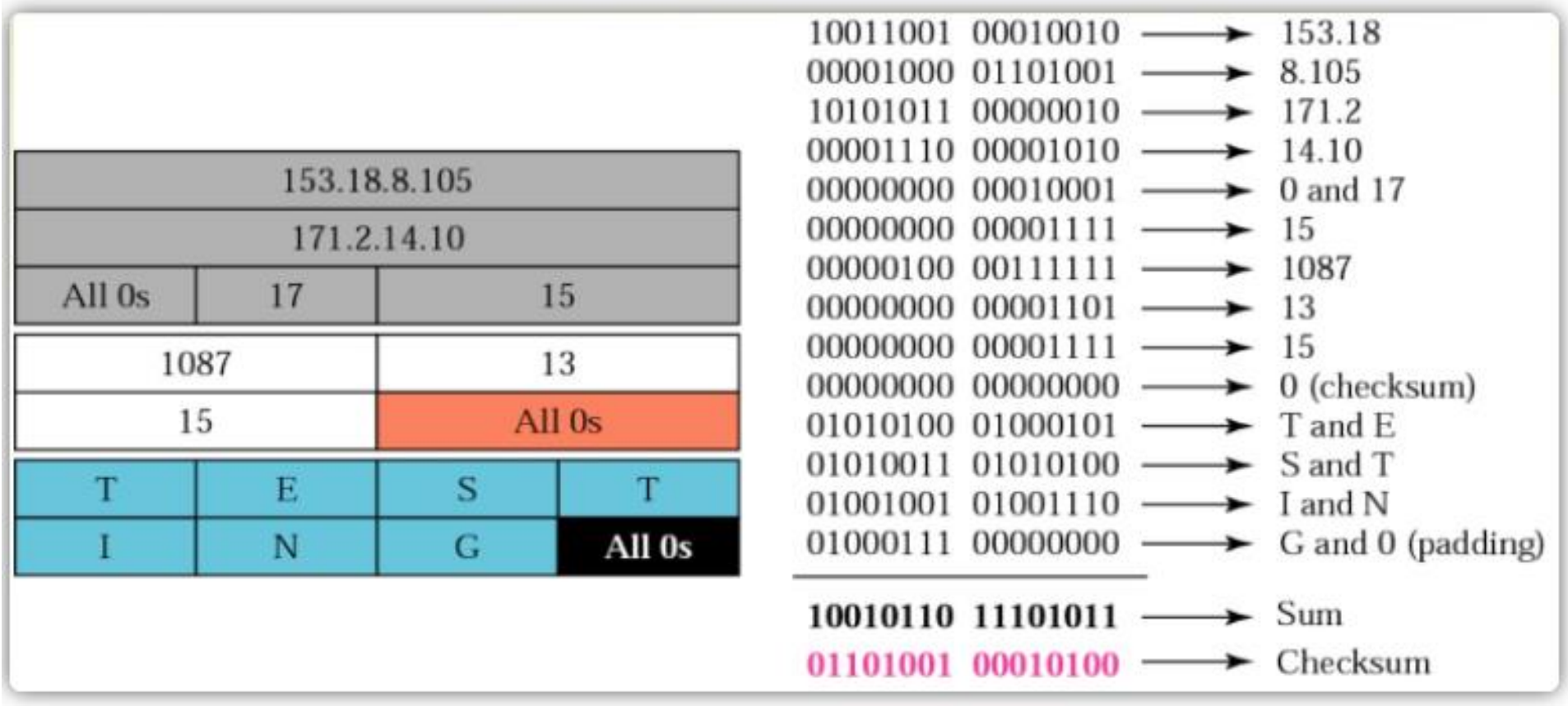
example: add two 16-bit integers



UDP Checksum Calculations



Cont..



At receiver, add everything including checksum and complement if solution is zero then packet is correctly received.

Source: TCP/IP Protocol Suite, Forouzan

Questions

What value is sent for the checksum in one of the following hypothetical situations?

a. The sender decides not to include the checksum.

b. The sender decides to include the checksum, but the value of the sum is all 1s.

c. The sender decides to include the checksum, but the value of the sum is all 0s.

Answers

Solution

a. The value sent for the checksum field is all 0s to show that the checksum is not calculated.

b. When the sender complements the sum, the result is all 0s; the sender complements the result again before sending. The value sent for the checksum is all 1s. The second complement operation is needed to avoid confusion with the case in part a.

c. This situation never happens because it implies that the value of every term included in the calculation of the sum is all 0s, which is impossible; some fields in the pseudoheader have nonzero values.

Point to Note

- UDP is an example of the connectionless simple protocol we discussed in as a part of Transport layer services with the exception of an optional checksum added to packets for error detection.

Summary: UDP

- Simple protocol:
  - segments may be lost, delivered out of order
  - best effort service: “send and hope for the best”
- UDP has its plusses:
  - no setup/handshaking needed (no RTT incurred)
  - can function when network service is compromised
  - helps with reliability (checksum) → **Optional**
- build additional functionality on top of UDP in application layer (e.g., HTTP/3)

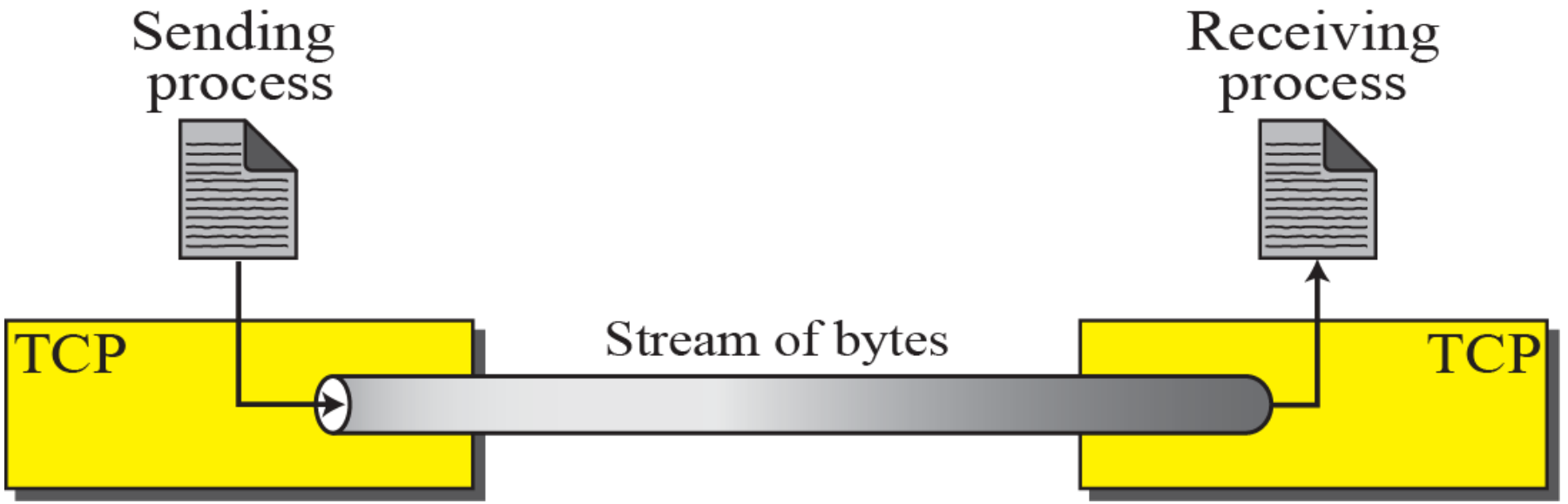
Chapter 3: roadmap

- Transport-layer services
- Connectionless transport: UDP
- Connection-oriented transport: TCP
  - segment structure
  - reliable data transfer
  - flow control
  - connection management
- Principles of congestion control
- TCP congestion control

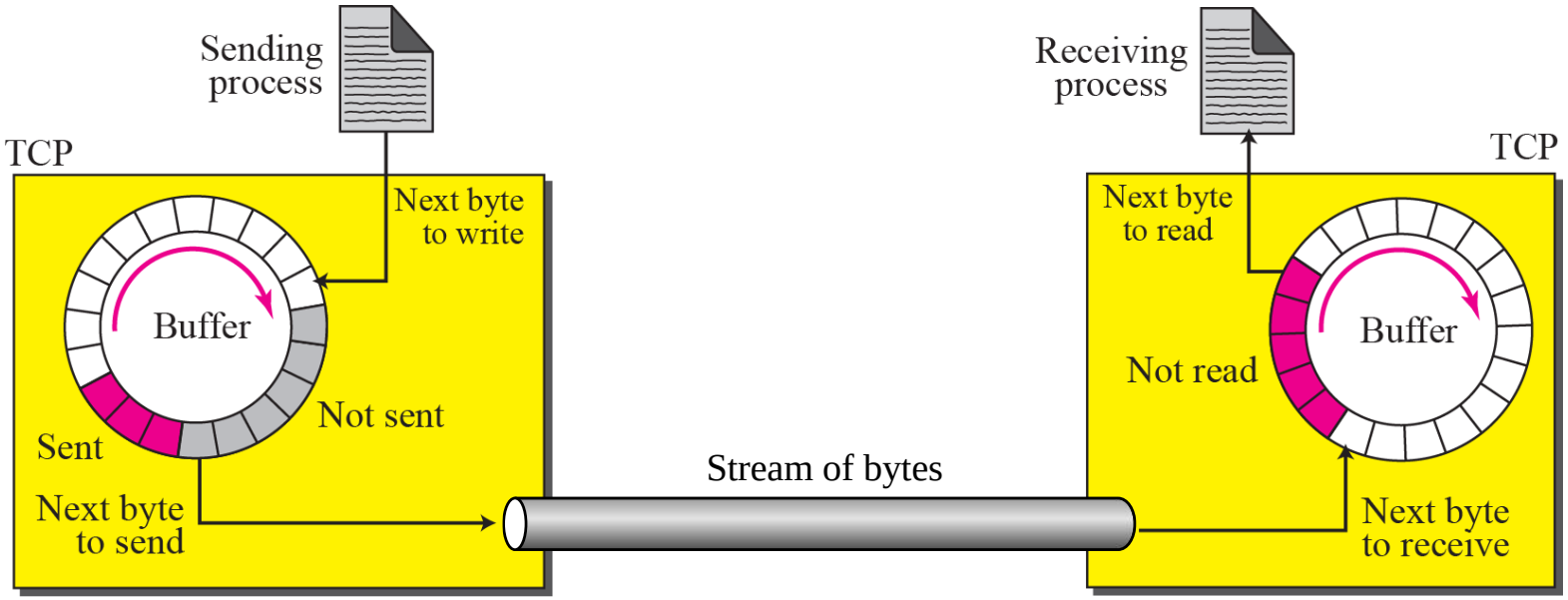
TCP: overview RFCs: 793,1122, 2018, 5681, 7323

- point-to-point:
  - one sender, one receiver
- reliable, in-order *byte steam:*
  - no “message boundaries"
- full duplex data:
  - bi-directional data flow in same connection
  - MSS: maximum segment size
- cumulative ACKs
- pipelining:
  - TCP congestion and flow control set window size
- connection-oriented:
  - handshaking (exchange of control messages) initializes sender, receiver state before data exchange
- flow controlled:
  - sender will not overwhelm receiver

Stream delivery

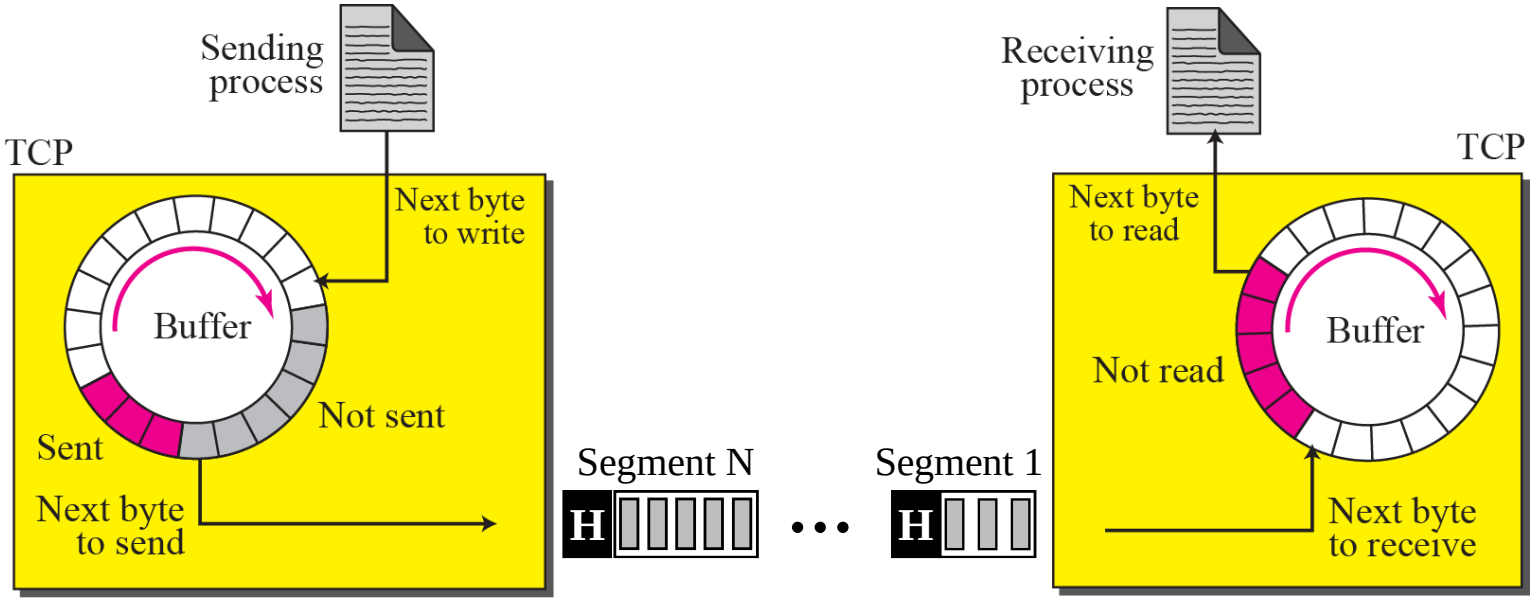


Sending and receiving buffers



TCP/IP Protocol Suite

TCP segments



TCP FEATURES

- To provide the services mentioned in the previous section, TCP has several features that are briefly summarized in this section and discussed later in detail.

✓ Numbering System
✓ Flow Control
✓ Error Control
✓ Congestion Control

Numbering System

- The bytes of data being transferred in each connection are numbered by TCP.
- The numbering starts with an arbitrarily generated number.
- Suppose a TCP connection is transferring a file of 5,000 bytes. The first byte is numbered 10,001. What are the sequence numbers for each segment if data are sent in five segments, each carrying 1,000 bytes?

Solution

The following shows the sequence number for each segment:

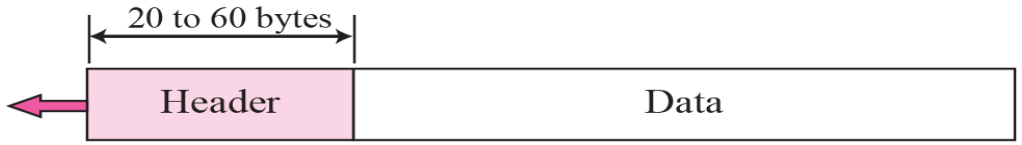
Segment 1	→	**Sequence Number:**	**10,001**	**Range:**	**10,001**	to	**11,000**
Segment 2	→	**Sequence Number:**	**11,001**	**Range:**	**11,001**	to	**12,000**
Segment 3	→	**Sequence Number:**	**12,001**	**Range:**	**12,001**	to	**13,000**
Segment 4	→	**Sequence Number:**	**13,001**	**Range:**	**13,001**	to	**14,000**
Segment 5	→	**Sequence Number:**	**14,001**	**Range:**	**14,001**	to	**15,000**

Cont..

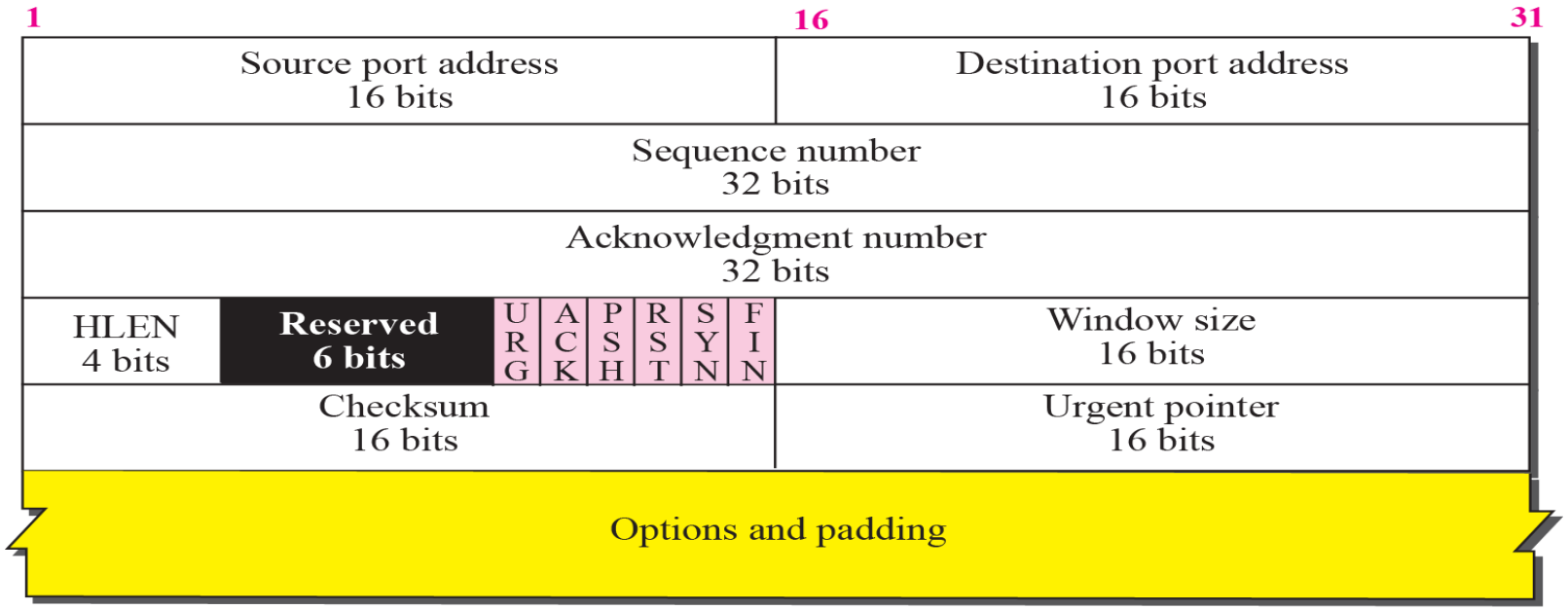
- The value in the sequence number field of a segment defines the number assigned to the first data byte contained in that segment.
- The value of the acknowledgment field in a segment defines the number of the next byte a party expects to receive.
- The acknowledgment number is cumulative.

TCP segment format

- Before discussing TCP in more detail, let us discuss the TCP packets themselves. A packet in TCP is called a segment.



a. Segment



b. Header