Chair of Distributed Systems and Security
School of Computation, Information and Technology
Technical University of Munich

Computer Networking and IT Security (INHN0012)

Tutorial 11

Problem 1 Flow and congestion control with TCP

The most widely used transport protocol on the Internet is TCP. This implements mechanisms for flow and congestion control.

a)* Discuss the differences between flow and congestion control. What are the objectives of each mechanism?

b) Assign each of the following terms to TCP flow or congestion control:

- Slow-Start
- Receive window
- Congestion-Avoidance
- Multiplicative-Decrease

To analyze the data rate that can be achieved with TCP, we consider the course of a contiguous data transmission in which the slow-start phase has already been completed. TCP is therefore in the congestion-avoidance phase. We designate the individual windows as follows:

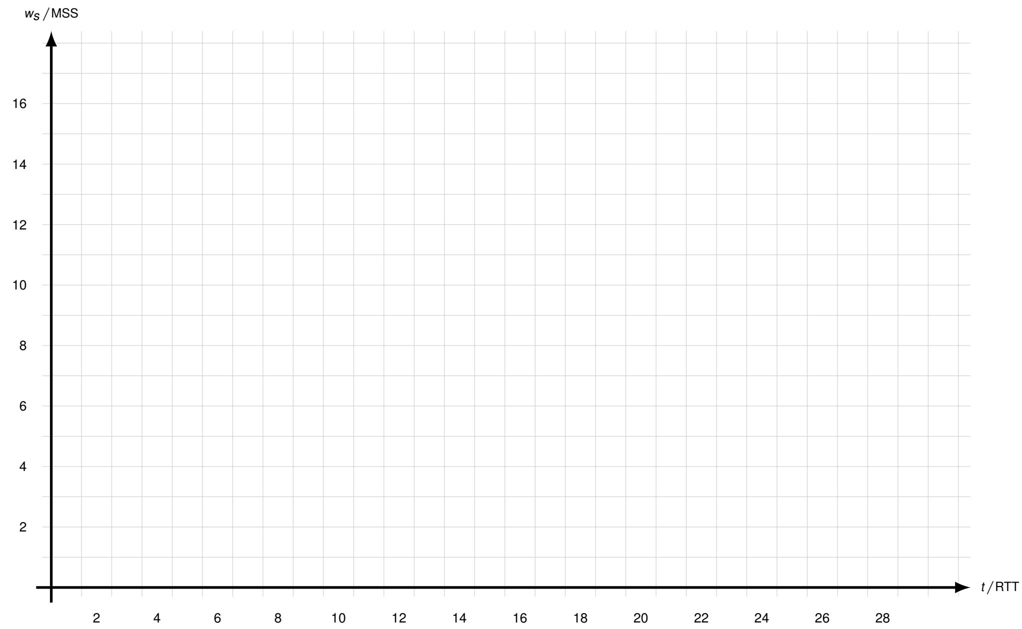
- Send window W_s, $|W_s| = w_s$
- Receive window W_r, $|W_r| = w_r$
- Traffic-control window W_c, $|W_c| = w_c$

We assume that the receive window is arbitrarily large, so that the transmit window is determined solely by the congestion control window, i. e.,h. $W_s = W_c$. No losses occur as long as the transmit window is smaller than a maximum value x, i. e., $w_s < x$.

If a full send window is confirmed, the currently used window will increase by exactly 1 MSS. If the send window has reached the value x, exactly one of the sent TCP segments is lost. The sender detects the loss by receiving the same ACK number multiple times. The station then halves the congestion control window, but still remains in the congestion-avoidance phase, i. e., no new slow-start takes place. This approach corresponds to a simplified variant of TCP-Reno (cf. lecture).

As concrete numerical values, we assume that the maximum TCP segment size (MSS) is 1460 B and the RTT is 200 ms. Let the serialization time of segments be negligible compared to the propagation delay. Segment loss occurs as soon as the transmit window reaches a size of $w_s \geq x = 16$ MSS.

c)* Create a graph plotting the current size of the transmit window w_s measured in MSS over the time axis t measured in RTT. In your diagram, at the time $t_0 = 0$ s, you want the transmit window size to have just been halved, so $w_s = x/2$ applies. Draw the diagram in the time interval $t = \{0, \dots, 27\}$.



d)* How much time elapses before the congestion control window is reduced again after a segment loss as a result of another segment loss?

e)* In general, determine the average loss rate L. Note: Since the behavior of TCP is periodic in this idealized model, it is sufficient to consider only one period. Set the total number of transmitted segments in relation to the number of lost segments (specification as a truncated fraction is sufficient).

f) Using the results from subtasks (c) and (e), determine the average achievable transmission rate in kB/s during the TCP transmission phase under consideration. **Note:** Use the exact value (fraction) from subtask e).

g)* What is the maximum transmission rate you could send over the channel with UDP without creating a congestion? Take into account that the UDP header 12 B is smaller than the TCP header without options.

Problem 2 **Network Address Translation**

This task will look at the forwarding of IP packets (IPv4) when using a NAT-enabled router. For the mapping between public and private IP addresses, a NAT-enabled router has a mapping table that stores the relationship between the local and the global port. Many NAT-enabled devices also store additional information such as the remote IP address or the router's own global IP address (e.g., if the router has more than one global IP). We will refrain from doing this here.
Figure 2.1 shows the network topology. Router R1 has NAT enabled, using a private IP address on IF1 and a public IP address on IF2. Router R2 does not use NAT. PC2 had already communicated with server 2, which created the entry in R1's NAT table (see the 2.1 figure). Where you have the freedom, choose sensible values for the IP addresses and port numbers.

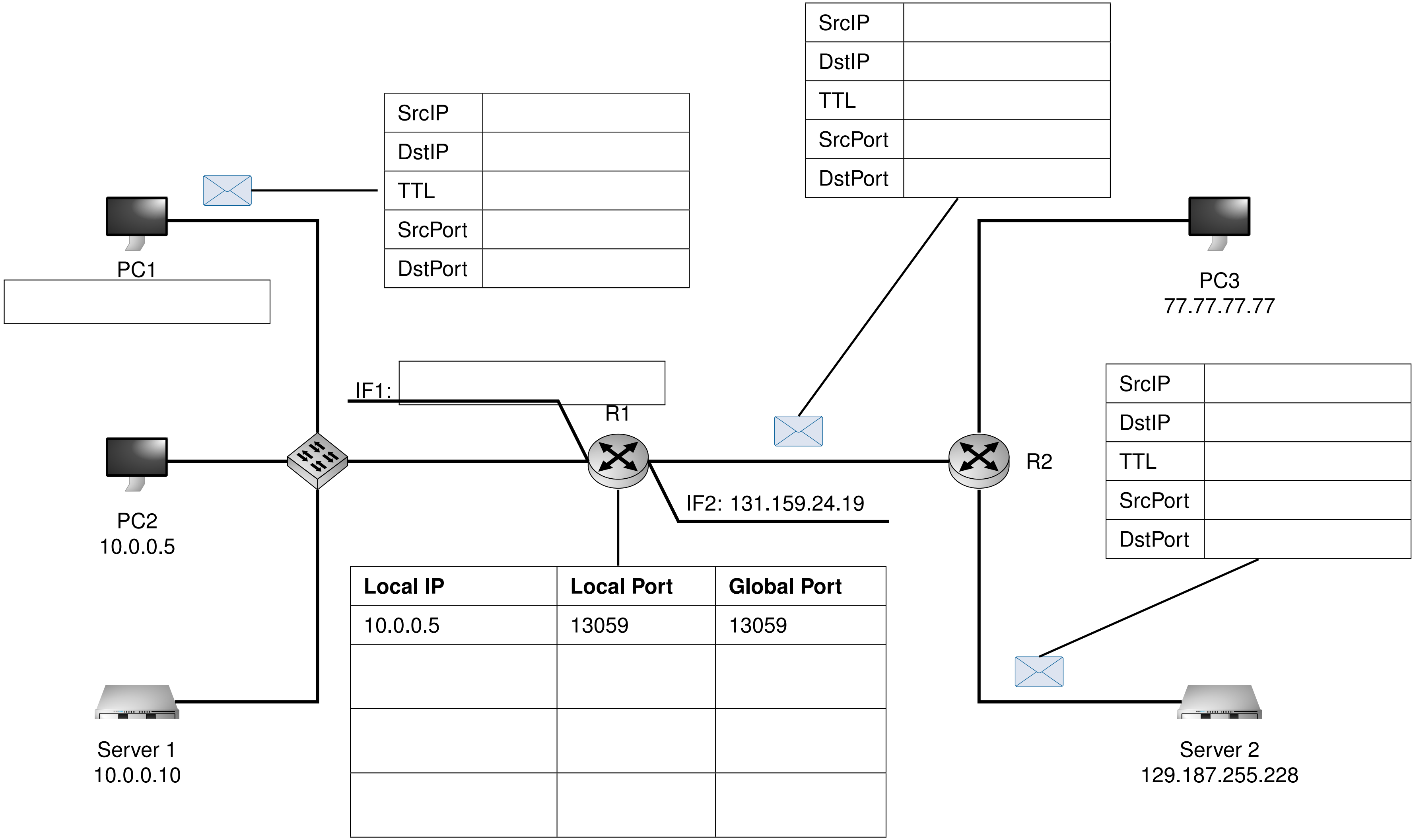
a)* Give PC1 and interface 1 of R1 a suitable IP address. The subnet is 10.0.0.0/24.

b)* PC1 now establishes an HTTP connection to server 2. Specify the source IP, destination IP, source port, destination port, and TTL fields of the IP or TCP header for the packets in the three highlighted locations in Figure 2.1. Also, specify newly created entries in the NAT table of R1.

Please enter in figure 2.1. The box is only used to give you an explanation at this point in the solution proposal!

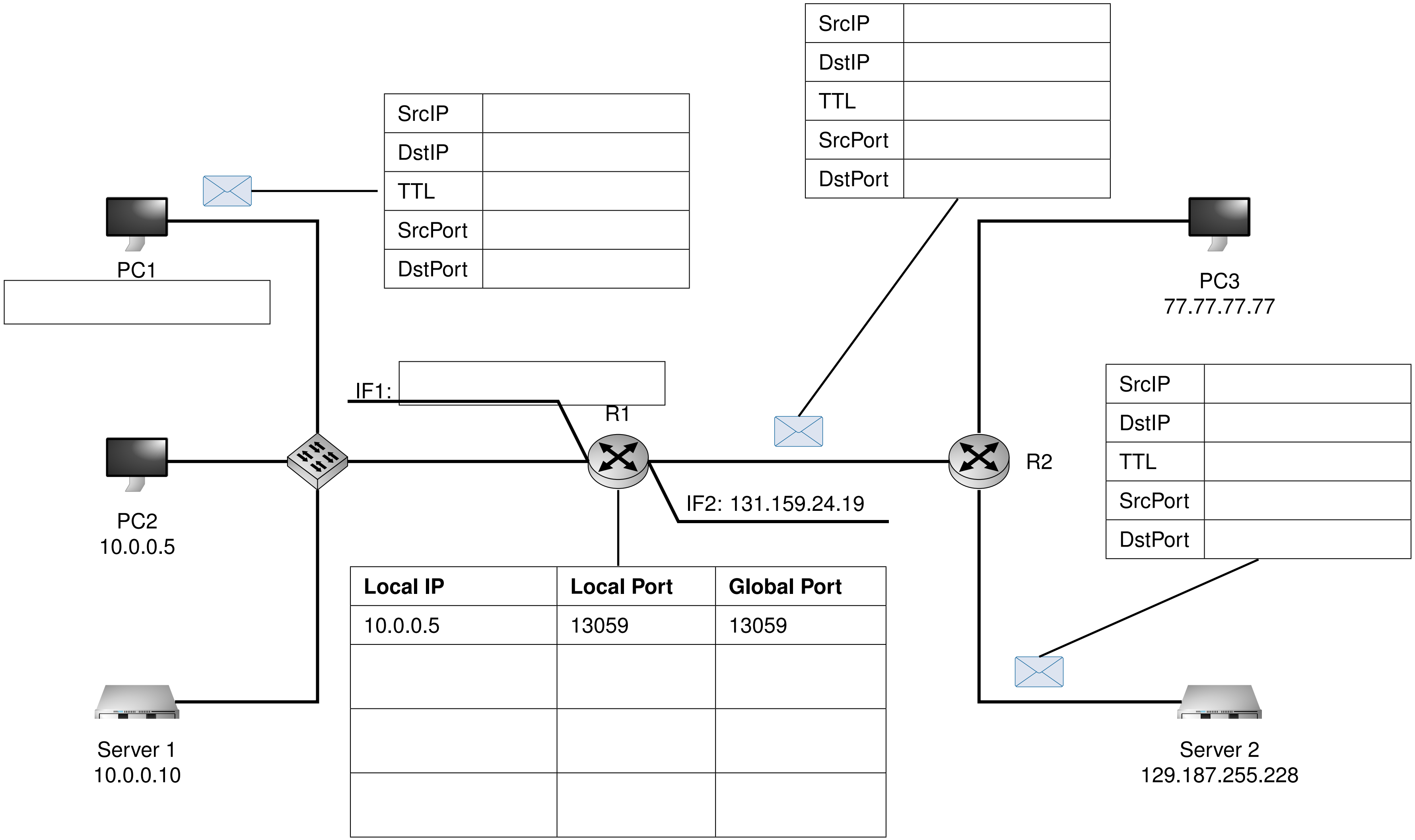
c) Server 2 now answers PC1. In Figure 2.2, specify the header fields at the three named locations, as well as newly created entries in the NAT table of R1, analogous to the previous subtask.

Please enter in figure 2.1. The box is only used to give you an explanation at this point in the solution proposal!



Local IP	**Local Port**	**Global Port**
10.0.0.5	13059	13059

Figure 2.1: Lösungsblatt für Aufgabe 2a)/b)



Local IP	**Local Port**	**Global Port**
10.0.0.5	13059	13059

Figure 2.2: Solution sheet for problem 2c)

d)* Server 1 now also establishes a TCP connection to server 2 on port 80. In doing so, it randomly chooses the sender port 13059. Describe the problem that occurs on the NAT and how it is solved.

e)* R1 receives a packet from PC3 addressed to `131.159.24.19:13059`. What will R1 do with this packet? What problems may arise from it?

f) Does a problem arise for PC2 when it receives a „random“ packet with TCP payload on a port with an existing connection?

g)* What other distinguishing criteria could be used by a NAT router?

h)* What problem arises when PC1 sends an echo request to server 2?

i) Describe a possible solution for the problem which arose in the previous subproblem.

j) What problem arises if a NAT router receives ICMP TTL-exceeded messages and wants to forward it to the intended receiver (sender who caused the problem)? How can this problem be worked around?

k)* Now PC3 wants to establish a HTTP connection with Server 1. Can this happen under the given circumstances? (Explain!)

l) How can this problem be avoided while mantaining a NAT?

Problem 3 TCP and Long Fat Networks (Homework)

In this task we consider so-called *Long Fat Networks*. This refers to connections that have a high transmission rate but, in particular, also a high delay. Examples include satellite links which as a result of the long distance have high propagation delays. In particular, we want to investigate the impact on TCP congestion control.

a)* For TCP, the send window is selected depending on the receive window as well as the congestion control window. What is the exact relationship between the windows?

Let two users be connected to the Internet at high transmission rates via a geostationary satellite. The RTT between the two users is 800 ms, the transmission rate be $r = 24\,\text{Mbit/s}$.

b)* How large should the transmission window (measured in bytes) be selected so that continuous transmission is possible?

c)* Why is the situation in subproblem b) a problem for TCP flow control? Hint: Have a look at the TCP header.

d)* Read Section 2 of RFC 1323 (`http://www.ietf.org/rfc/rfc1323.txt`, see Appendix). Describe the solution to the problem from subproblem c).

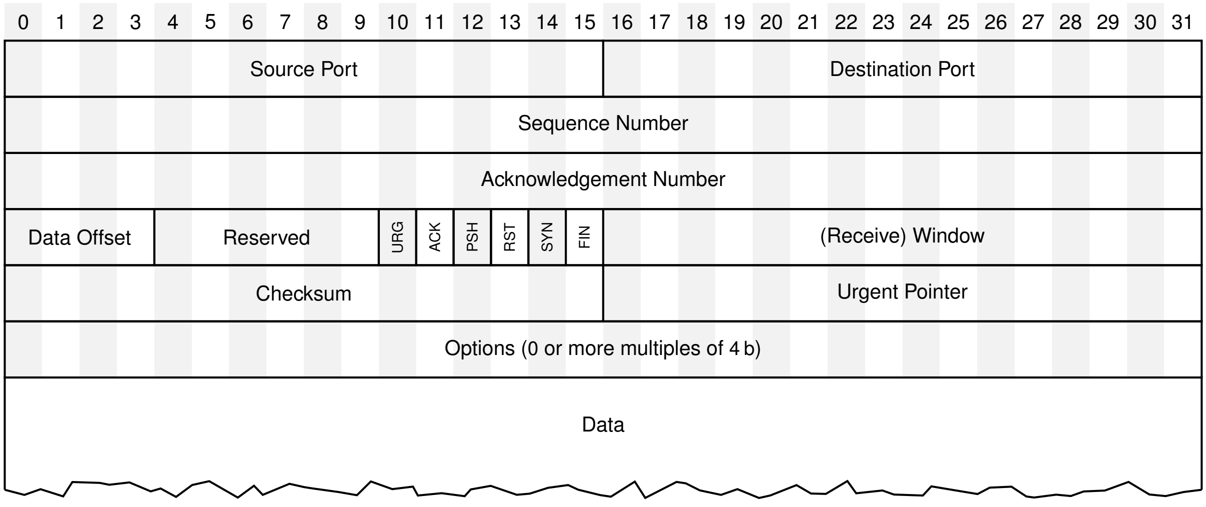
e) Determine the minimum value for the `shift.cnt` field of the TCP window scaling option.

f) Specify the header of the first TCP SYN packet that establishes the connection. To do this, use the concrete numerical values from the specification. A TCP header is shown again in Figure 3.1 as a reminder. There you will also find two forms for the solution.
Note: It is not necessary to fill the header in binary. However, please make it clear whether the numbers are represented in hexadecimal, decimal, or binary.

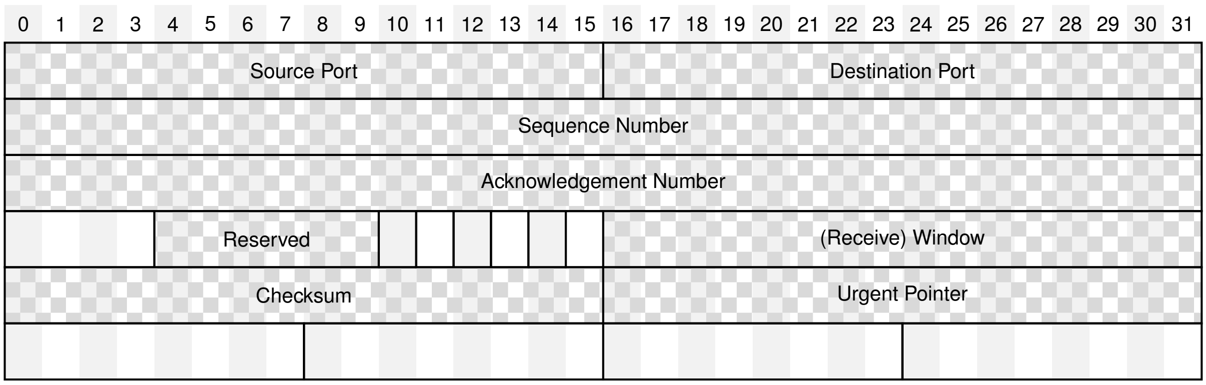
Assume that the size of the traffic control window is currently half of the value calculated in subproblem b). The MSS is 1200 B and the TCP connection is currently in the congestion-avoidance phase.

g) How long does it take for the window to fully utilize the line?
Note: The congestion control window is not directly affected by TCP window scaling.

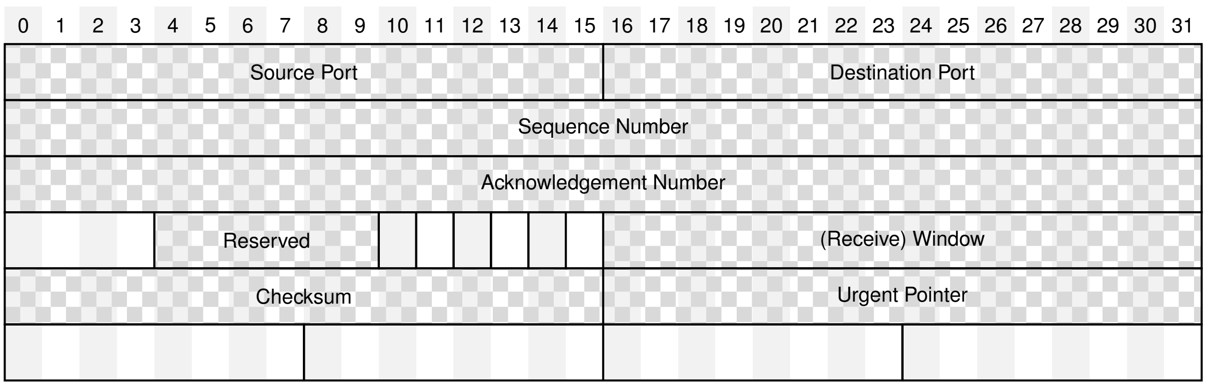
h) Does the result of subproblem g) result in a problem?



(a) TCP-Header



(b) Preprint



(c) Another preprint

Figure 3.1: TCP headers for solving task 3

2. TCP WINDOW SCALE OPTION

2.1 Introduction

The window scale extension expands the definition of the TCP window to 32 bits and then uses a scale factor to carry this 32- bit value in the 16-bit Window field of the TCP header (SEG.WND in RFC-793). The scale factor is carried in a new TCP option, Window Scale. This option is sent only in a SYN segment (a segment with the SYN bit on), hence the window scale is fixed in each direction when a connection is opened. (Another design choice would be to specify the window scale in every TCP segment. It would be incorrect to send a window scale option only when the scale factor changed, since a TCP option in an acknowledgement segment will not be delivered reliably (unless the ACK happens to be piggy-backed on data in the other direction). Fixing the scale when the connection is opened has the advantage of lower overhead but the disadvantage that the scale factor cannot be changed during the connection.)

The maximum receive window, and therefore the scale factor, is determined by the maximum receive buffer space. In a typical modern implementation, this maximum buffer space is set by default but can be overridden by a user program before a TCP connection is opened. This determines the scale factor, and therefore no new user interface is needed for window scaling.

2.2 Window Scale Option

The three-byte Window Scale option may be sent in a SYN segment by a TCP. It has two purposes: (1) indicate that the TCP is prepared to do both send and receive window scaling, and (2) communicate a scale factor to be applied to its receive window. Thus, a TCP that is prepared to scale windows should send the option, even if its own scale factor is 1. The scale factor is limited to a power of two and encoded logarithmically, so it may be implemented by binary shift operations.

```
      TCP Window Scale Option (WSopt):
         Kind: 3 Length: 3 bytes
                +---------+---------+---------+
                | Kind=3  |Length=3 |shift.cnt|
                +---------+---------+---------+
```

This option is an offer, not a promise; both sides must send Window Scale options in their SYN segments to enable window scaling in either direction. If window scaling is enabled, then the TCP that sent this option will right-shift its true receive-window values by 'shift.cnt' bits for transmission in SEG.WND. The value 'shift.cnt' may be zero (offering to scale, while applying a scale factor of 1 to the receive window).

This option may be sent in an initial <SYN> segment (i.e., a segment with the SYN bit on and the ACK bit off). It may also be sent in a <SYN,ACK> segment, but only if a Window Scale op- tion was received in the initial <SYN> segment. A Window Scale option in a segment without a SYN bit should be ignored.

The Window field in a SYN (i.e., a <SYN> or <SYN,ACK>) segment itself is never scaled.