Communication Networks

Prof. Laurent Vanbever

Exercises week 4 – Reliable Transport

Reliable versus Unreliable Transport

In the lecture, you have learned how a reliable transport protocol can be built on top of a best-effort delivery network. However, some applications still use an unreliable transport protocol.
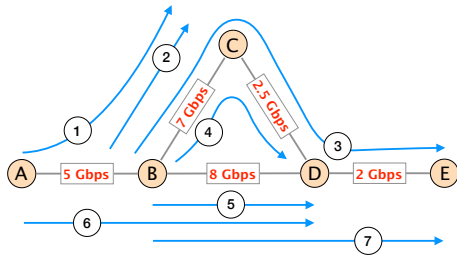
a) What are the characteristics of best-effort and of reliable transport?

b) What could be advantages of using an unreliable transport protocol?

c) What type of applications are suitable to use unreliable transport protocols?

d) As we will later see, the User Datagram Protocol (UDP) only provides unreliable transport. Assume you are forced to use a network which only supports UDP as a transport protocol. You must transmit an important document which eventually should be correctly transmitted. Do you see a way to implement some of the reliable transport mechanisms despite using UDP?

Negative Acknowledgments

In the lecture, we have mainly looked at transport protocols using (positive) Acknowledgments (ACKs). However, we could also use so called Negative Acknowledgments (NAKs or NACKs). In this case, the receiver is sending a NAK for every packet that it *did not* receive. To detect lost packets, the receiver looks at the sequence numbers of all the received packets and sends NAKs for every missing sequence number. After receiving a NAK, the sender will retransmit the corresponding packet.

a) Assuming a network with nearly no packet loss, what could be the main advantage of using NAKs?

b) Assume now that the receiver will immediately send a NAK as soon as it detects a gap in the received packet numbers. E.g. for the following packet number sequence [4, 5, 7] the receiver would immediately send a NAK for packet 6. Can you see a problem with this implementation? How could you (partially) mitigate the problem?

c) So far, NAKs look like a good alternative to (positive) ACKs. Nonetheless, TCP – the currently most-widely used transport protocol – is *not* using NAKs. There has to be a problem. Assume that the sender is transmitting 5 packets (with sequence number 1 to 5). Find at least two sequences of packet or NAK losses such that the **sender** wrongly assumes that the 5 packets were correctly received.
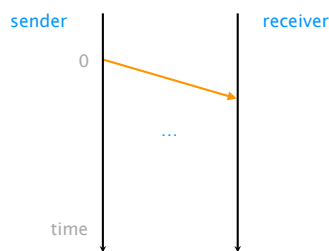
Fairness

Consider the network on the left consisting of 5 nodes (A to E). Each link has a maximal bandwidth indicated in red. 7 flows (1 to 7) are using the network at the same time. You can assume that they have to send a lot of traffic and will use whatever bandwidth they will get. Apply the max-min fair allocation algorithm discussed in the lecture to find a fair bandwidth allocation for each flow. You can use the table below. In the top row, indicate which link is the current bottleneck. The other rows contain the corresponding bandwidth distribution for each flow.

A network with shared links and 7 flows.

Bottleneck link					
Flow 1 A - B - C					
Flow 2 B - C					
Flow 3 B - C - D - E					
Flow 4 B - C - D					
Flow 5 B - D					
Flow 6 A - B - D					
Flow 7 B - D - E					

Reliable Transport (Exam Style Question)

How long would a transfer take?

Consider a Go-Back-N sender and receiver directly connected by a 10 Mbps link with a propagation delay of 100 milliseconds. The retransmission timer is set to 3 seconds and the window has a length of 4 segments.

Draw a time-sequence diagram (see left) showing the transmission of 10 segments (each segment contains 10 000 bits). An ACK is transmitted as soon as the last bit of the corresponding data segment is received. The size of an ACK is very small, that means they have an negligible transmission delay.

a) Draw the time-sequence diagram for the case where there are no losses.

b) Draw the time-sequence diagram for the case where the 3rd and the last segment are lost once.

Reliable Transport[2] (Exam Style Question)

On the next page you see the beginning of a communication between two end-points using the Go-Back-N protocol with Selective Repeat. Consider that the sender has infinitely many data segments to send and they are immediately available.

We ask you to fill in the missing values in the two tables. Stop if you either reach the bottom of the tables or the sender is no longer able to send new data segments because its buffer is full. Start with the blue row indicated on the left.

Note: Please read the entire question carefully!

Set-up:

- Every table row corresponds to one time-slot. The sender and receiver can send one data segment respectively ACK segment in every time-slot;

- Consider that the *Sender buffer* contains all the sent but not yet acknowledged segments, while the *Out-of-order buffer* contains all the messages which has been received... out-of-order;

- If the sender receives an ACK in one time-slot, it first processes the ACK (e.g. removes segments from the sender buffer) and then sends the data segment for this time-slot. Similarly, the receiver will first analyse the received data segment and then send a corresponding ACK;

- The link between the sender and receiver is not reliable. The first data segment with a sequence number of 3 and all data segments with a sequence number of 5 are dropped and do not reach the receiver.

Sender behavior:

- The sender uses Selective Repeat after receiving 3 duplicate ACKs. That means as soon as the sender receives an ACK with the same sequence number for the third time, it will retransmit the missing segment in the same time-slot (instead of a new data segment);

- The sender can store at most 5 unacknowledged segments in its sender buffer.

Assumptions:

- You will not reach the maximal sequence number. No overflow;

- The timeout value is very long and will not occur;

- The receiver out-of-order buffer can store an unlimited number of segments.

Fill the following table starting from the blue row on the left.

GBN - Sender
maximal sender buffer size = 5

Sender buffer	Data segment to send	Received ACK number
[0]	0	—
[1]	1	1

Drop **first** data segment 3 and **all** data segments 5

GBN - Receiver

Received data segment	ACK number to send	Out-of-order buffer
0	1	[-]
1	2	[-]