ETH zürich

Networked Systems
ETH Zürich — seit 2015

Communication Networks

Prof. Laurent Vanbever

Solution: Exercise 9 – Reliable transport concepts and GBN

Concepts

9.1 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?

Solution:

- Best-effort delivery: There is no guarantee for packets to arrive in the correct order, correctly (bit corruption) or even arrive at all.
- Reliable transport: It provides all the above guarantees by making use of sequence numbers, checksums and acknowledgements.

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

Solution:

- Better performance/less overhead since you don't have to wait for ACKs to arrive;
- Lightweight implementation;
- As no connection setup is required (e.g., TCP three-way handshake), you can immediately start sending.

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

Solution: Applications for which it is more important to have "live" data than to have "complete" data. In voice/video-calls, for example, lost packets lead to lower quality, but delayed packets lead to distorted conversations. Some VPN applications also use UDP to reduce the overhead of a full TCP connection.

d) 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?

Solution: Yes, the reliable transport mechanisms could be implemented by the application/in the application layer. As an example, the relatively new QUIC protocol (initially designed by Google) works on top of UDP and provides e.g., a much faster session setup time compared to normal TCP sessions.

9.2 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?

Solution: The number of NAKs will be much smaller than the number of ACKs in a normal case. Fewer packets in the network could have a positive influence on the delay, bandwidth, ...

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?

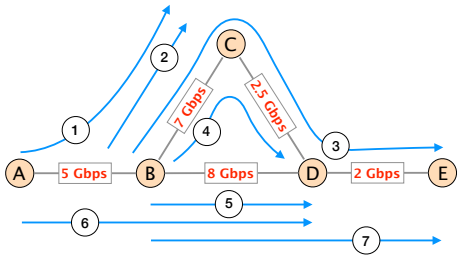
Solution: Reordered packets will immediately trigger a NAK. The receiver could e.g. wait for a certain amount of time before sending the NAK.

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.

Solution:

- [1, 2, 3] correctly received. Packet 4 and 5 were lost.
- [1, 2, 3, 5] correctly received. The NAK for packet 4 was lost.

9.3 Fairness



A network with shared links and 7 flows.

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.

Solution:

Bottleneck link	D-E	C-D	B-C	A-B	B-D	
Flow 1 A - B - C	1	1.5	**2.25**			
Flow 2 B - C	1	1.5	**2.25**			
Flow 3 B - C - D - E	**1**					
Flow 4 B - C - D	1	**1.5**				
Flow 5 B - D	1	1.5	2.25	2.75	**4.25**	
Flow 6 A - B - D	1	1.5	2.25	**2.75**		
Flow 7 B - D - E	**1**					

Go-Back-N (GBN)

9.4 Understanding Go-Back-N's Behavior (Exam Style Question)

Assume you have a Go-Back-N (GBN) sender and receiver. The receiver acknowledges *each* data segment with a cumulative ACK which indicates the next expected data segment. Furthermore, it saves out-of-order segments in a buffer. The sender and receiver buffer can contain four segments each. The timeout period is much larger than the time required for the sender to transmit four segments in a row.

a) The sender wants to transmit 10 data segments (0,...,9) to the receiver. Assume that *exactly* one segment is lost. How many segments has the sender to transmit in the best (resp. worst) case? For each case, indicate which segment was lost.

Solution:

- Best case: 11 segments, the last segment is dropped.
- Worst case: 14 segments, e.g., the second segment is dropped. GBN will retransmit all packets in the current window.

b) Once again, the sender wants to transmit 10 data segments (0,..., 9) to the receiver. This time, assume that exactly one ACK is lost. How many segments does the sender have to transmit in the best (resp. worst) case and which ACK was lost?

Solution:

- Best case: 10 segments, e.g., the ACK for segment 5 is lost. Since GBN uses cumulative ACKs, the ACK for segment 6 implicitly also acknowledges segment 5.
- Worst case: 11 segments, the ACK for the very last segment is lost.

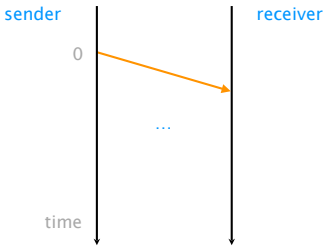
c) Assume the sender just transmitted segments 4, 5, 6 and 7 and is now waiting for ACKs from the receiver. It receives three times an ACK with number 4. Therefore, it cannot remove segments from its buffer and eventually the timeout is reached. Following the GBN protocol, the sender will retransmit all four segments.

A friend of yours explains that she improved her GBN algorithm so that, in the case above, the sender would just retransmit data segment 4 (instead of all four segments). She tells you that, quite often, she would then get an ACK with number 8 back (all four packets were successfully transmitted).

Can you explain why your friend believes that only data segment 4 was missing? Under which network conditions would the proposed improvement *not* work (assuming you still get three times an ACK with number 4)?

Solution: To send three ACKs with number 4, the receiver also has to receive three data segments. That is for example possible if it received segments 5, 6, 7, but not segment 4 (otherwise, not all ACKs would have number 4). As out-of-order segments are saved in a buffer, retransmitting segment 4 is enough. However, if data segments or ACKs are duplicated in the network, we can no longer be sure that data segment 4 was the only missing segment.

9.5 Reliable Transport (Exam Style Question)



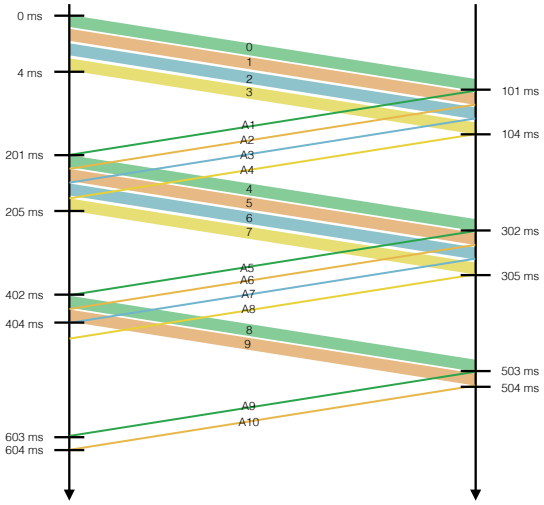
How long would a transfer take?

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.

Solution: The acknowledgments always point to the next expected sequence number and not to the sequence number of the last received segment. This means that, for example, the segment with sequence number 5 is acknowledged with A6.



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

Solution: