Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Networked Systems
ETH Zürich — seit 2015

Communication Networks

Prof. Laurent Vanbever

Solution: Exercise 3 – Transport Concepts

3.1 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.

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?

Solution: Yes, the reliable transport mechanisms could be implemented by the application/in the application layer.

3.2 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.

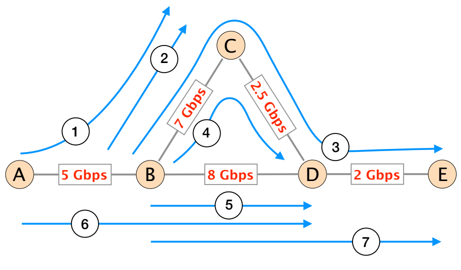
c) So far, NAKs look like a good alternative to (positive) ACKs. Nonetheless, TCP – the currently most-widely used transport protocol – is *not* using NAKs. There has to be a problem. Assume that the sender is transmitting 5 packets (with sequence number 1 to 5). Find at least two sequences of packet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

3.3 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.

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**					