ETH zürich

Networked Systems
ETH Zürich — seit 2015

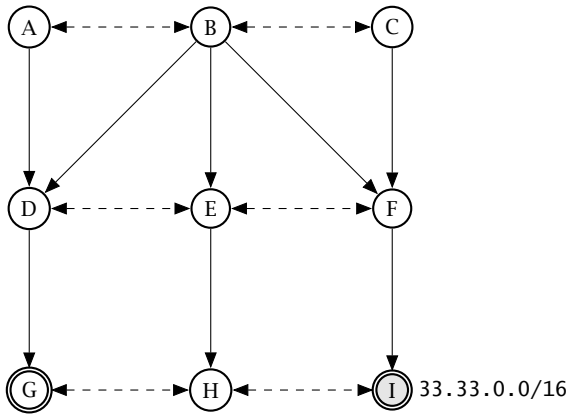
Communication Networks

Prof. Laurent Vanbever

Solution: Exercise 9 – BGP Hijack and Reliable Transport Concepts

Return of the BGP Hijack

9.1 BGP Security (Exam Question 2020)



An Internet topology of 9 ASes in which AS I announces a prefix and AS G tries to hijack it.

Consider the Internet topology consisting of 9 Autonomous Systems (ASes) in the Figure above. Single-headed plain arrows point from providers to their customers (AS A is the provider of AS D) while double-headed dashed arrows connect peers (AS A and AS B are peers). Each AS is made up of a single BGP router and applies the default selection and exportation BGP policies based on their customers, peers and providers.

In this task, the routers break ties using the AS number of the neighbor: in case multiple routes are equally good, the router selects the route of the neighbor with the lowest AS number (in alphabetical order; e.g., a route from AS A is preferred over AS B in case of a tie).

AS I is the origin of prefix `33.33.0.0/16` and advertises it to its neighbors. Independently of what the external advertisements are, AS I ***always*** prefers its internal route to reach any IP destination in `33.33.0.0/16`.

a) AS G wants to hijack the traffic going to AS I for `33.33.0.0/16`. It starts advertising the exact same prefix with itself, AS G, as origin. From which ASes is it able to hijack the traffic?

Solution: We can hijack traffic from the following ASes: A, B, D, E, H.

b) The ASes notice the hijack and, as a counter-measure, deploy Resource Public Key Infrastructure (RPKI) Internet-wide. After that, from which ASes is the attacker able to hijack the traffic by still advertising the exact same prefix with itself as origin?

Solution: AS G is not able to attract any traffic anymore as all the ASes can tell that the route is invalid.

c) RPKI has a flaw. What is the problem of RPKI? How can AS G hijack the prefix `33.33.0.0/16` despite RPKI? From which ASes is AS G able to hijack the traffic?

Solution: RPKI only checks whether the origin AS indeed owns the advertised prefix, but it has no way of validating that the announcement actually originated at the correct AS and has not just been added to the beginning of the AS path.

AS G can simply add a fake origin to its announcements and advertise `33.33.0.0/16` with an AS path of G, I.

By doing so, it can only attract the traffic from ASes A, D. Since AS G prepends the true origin to the AS path, the AS path length of its announcements increases by one. Hence, some ASes prefer the true announcement over the hijack because of that (e.g., AS B).

d) In response, the ASes switch to BGPSec (Secure BGP). Explain what security it provides and how AS E can detect that the announcement from AS G has a forged AS path.

Solution: BGPSec provides origin and path authentication through cryptographic signatures. It allows to verify the entire path.

In BGPSec, every AS signs the message with its private key and includes the AS number of the next AS on the path. When a router receives an announcement, it can check each signature on the path using the public keys of the ASes. It then quickly sees whether the path is valid or not. In this case, AS E could detect that the origin (i.e., the very first hop on the AS path) is not correctly signed.

Reliable Transport Concepts

9.2 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.

9.3 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?

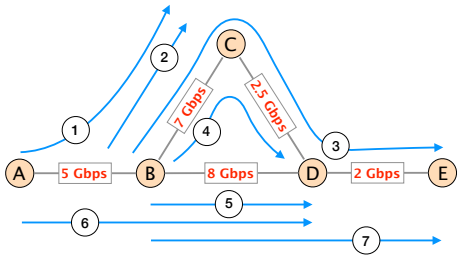
Solution: Reordered packets will immediately trigger a NAK. The receiver could e.g. wait for a certain amount of time before sending the NAK.

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.

9.4 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**					

9.5 Go-Back-N Warm-Up Questions

Sender and receiver keep separate windows and buffers for sent and received segments.

a) Compare how the sender and the receiver advance their respective windows.

b) Which segments does the *sender* buffer? When can segments be removed from the buffer?

c) A *receiver* typically buffers out-of-order segments. What is the advantage of such a buffer?

Solution:

a) The sender can advance its window when an in-order acknowledgment arrives. The receiver can advance its window with each in-order segment.

b) The sender needs to buffer all sent, but unacknowledged, segments. When the segment with the lowest sequence number is acknowledged, it can be removed.

c) An out-of-order buffer helps to reduce unnecessary retransmission of already received segments.

Cumulative ACKs acknowledge that all segments *up to* the acknowledged segment have been received.

d) Why are cumulative ACKs used? Do they help with lost data segments, lost ACKs, or both?

Solution:

d) Cumulative ACKs improve the behaviour for lost ACKs, as each ACK covers all previous ones, which can help to avoid retransmitting already received segments due to lost ACKs.

When *Fast Retransmit* is used, the sender retransmits a segment after duplicate ACKs.

e) How does Fast Retransmit improve performance?

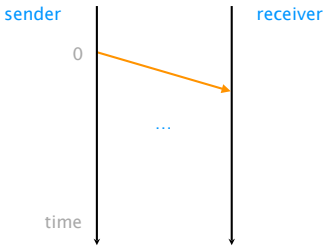
f) Compare Fast Retransmit in the case of mild congestion (some segment losses) and heavy congestion (nearly all segments lost).

Solution:

e) With duplicate ACKs, the sender can detect isolated segment loss and can quickly resend segments, without waiting for timeouts, thus improving performance.

f) Fast Retransmit works best for mild congestion, as explained above. In the case of heavy congestion, not enough segments may arrive to even receive a significant number of duplicate ACKs. In this case, the sender needs to wait for the timeout.

9.6 Reliable Transport (Exam Style Question)



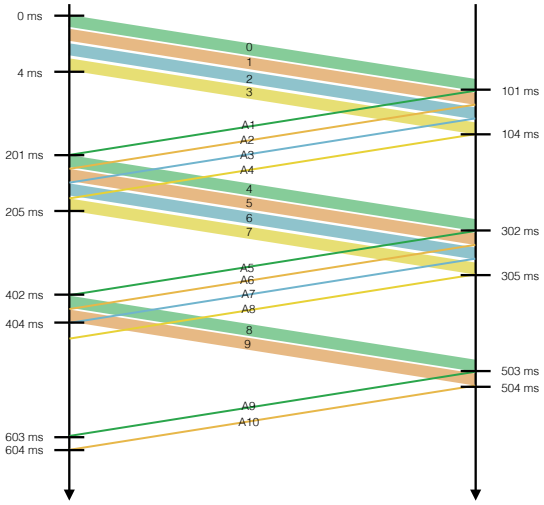
How long would a transfer take?

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 a 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: