ETH zürich

Networked Systems
ETH Zürich — seit 2015

Communication Networks

Prof. Laurent Vanbever

Solution: Exercise 99 – Additional Practice

Link State and Distance Vector

99.1 Warm-up Questions (Exam Question 2016)

For the following statements, decide if they are *true* or *false*. Motivate your decision.

a) Consider a positively weighted graph G. Applying the Bellman-Ford (used by distance-vector protocols) or Dijkstra (used by link-state protocols) algorithm on G would lead to the same forwarding state.

Solution: False. In general, both solve the shortest-path problem. However, whenever multiple shortest paths exist, the two algorithms can decide differently based on their tie-breaking criteria.

b) Link-state protocols (such as OSPF) are guaranteed to compute loop-free forwarding state as long as the link-state databases are consistent on all routers.

Solution: True. However, they can experience transient loops while it isn't the case.

c) Link-state protocols (such as OSPF) require routers to maintain less state than distance-vector protocols (such as RIP).

Solution: False. Link-state protocols require routers to maintain the entire topology in memory (Link-State database). Distance-vector protocols only need to maintain the costs to reach each prefix.

d) Poisoned reverse solves the problem of count-to-infinity.

Solution: False. The problem is still there it is mitigated by having a small infinity value.

e) Consider a positively weighted graph G. Multiplying all link weights by 2 would change the all-pairs shortest paths computed by the Dijkstra algorithm on G.

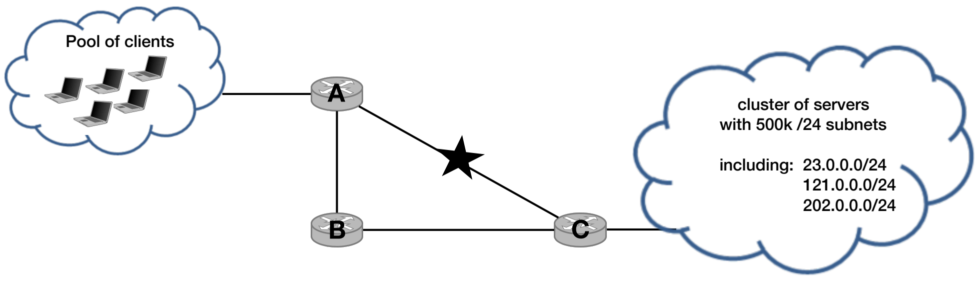
Solution: False. Multiplying by a constant factor keeps the ranking between the paths constant.

f) Consider a positively weighted graph G. Adding 1 to all link weights would change the all-pairs shortest paths computed by the Dijkstra algorithm on G.

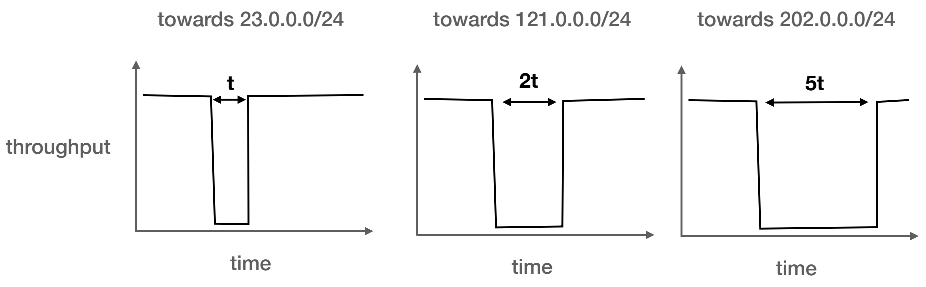
Solution: True. Longer paths will see a bigger increase than shorter ones.

99.2 A very long downtime (Exam Question 2019)

Consider the network in the Figure below which connects a large cluster of 500k servers (right) with a large pool of clients (left). The network uses OSPF internally. Each of the 500k servers is located in a different IP subnet, while all of the clients are located in the same /8 subnet.



Consider that the link between router A and C fails. As router A is adjacent to the failure, it immediately detects it and starts rerouting the traffic for the 500k prefixes to B. The Figure below reports the evolution of the throughput observed for 3 of the 500k server prefixes: 23.0.0.0/24, 121.0.0.0/24, and 202.0.0.0/24. One can see that the downtime experienced by each prefix is different: 23.0.0.0/24 experiences less downtime than 121.0.0.0/24, which itself experiences less downtime than 202.0.0.0/24.



a) Explain why the downtimes experienced by the three prefixes differ.

Solution: The router updates the entry in its forwarding one by one, and since there are more than 500k entries to update, it takes quite some time. Hence, it takes for each entry a different time until it is updated.

b) Describe two solutions to speed up the convergence for these three prefixes.

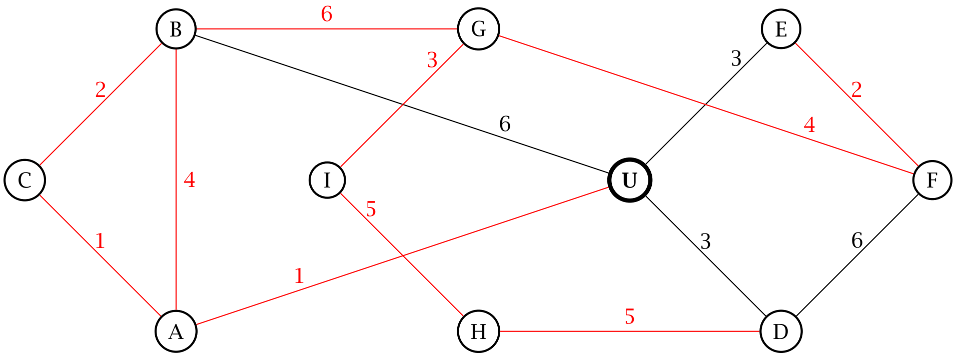
Solution: One option is to prioritize the three prefixes by updating the data plane for these three prefixes before all the others. Another option is to use a hierarchical forwarding table to speed up the overall convergence process. To this end, one would use two tables: a first table that maps all of the 500k prefixes to a much smaller number of virtual next-hops and then a second table that maps the virtual next-hops to the actual output port. In this case, one does not have to update all the entries for the 500k prefixes, but only change the output port of a few virtual next-hops.

99.3 Reverse Dijkstra (Exam Question 2024)

You are looking at a drawing of your network but it clearly seems some links are missing. You also have a table which shows the *partial* output of the Dijkstra's algorithm (some cells are empty) as performed in the lecture starting from the node **U**. For each iteration, the table indicates the weights of the shortest paths found so far. If after one iteration there are multiple nodes with an equally-shortest path, the algorithm continues with the node which comes first in the alphabet. There is at most one link between two nodes and each link has a strictly positive weight $[1, \infty]$.

You feel confident that, with the information from the network drawing and the partial output from Dijkstra's algorithm in the table below, you can find some of those missing links and link weights.

a) First, fill in the highlighted cells in the table below using the information available in the network graph. Write your answers directly in the table.



A network consisting of 10 nodes with missing links and link weights

Solution:

#	**U**	A	B	C	D	E	F	G	H	I
1	0	1	**6**	∞	**3**	**3**	∞	∞	∞	∞
2	0	1	5	2	3	3	∞	∞	∞	∞
3	0	1	4	2	3	3	∞	∞	∞	∞
4	0	1	4	2	3	3	**9**	∞	8	∞
5	0	1	4	2	3	3	5	∞	8	∞
6	0	1	4	2	3	3	5	10	8	∞
7	0	1	4	2	3	3	5	9	8	∞
8	0	1	4	2	3	3	5	9	8	13
9	0	1	4	2	3	3	5	9	8	12
10	0	1	4	2	3	3	5	9	8	12

For each iteration (1 to 10) the table shows the shortest path found by Dijkstra's algorithm performed on node **U** towards all other nodes.

b) Now that you have completed the table you can use this information to piece together the missing links in the network above. Draw all the links and corresponding weights you can identify directly.

Solution: See red links in figure above

c) You now wonder whether you actually found all the links or not. For all the following links, mark if they could exist or not.

Use the following format:

- If the link could exist, indicate the range of weights this link is allowed to have;
- If the link could not exist, write down the iteration in which the link would have showed up.

Link `U-C`: cannot exist, since C would need to show up in the first iteration.

Link `B-E`: can exist, weight $[1, \infty]$

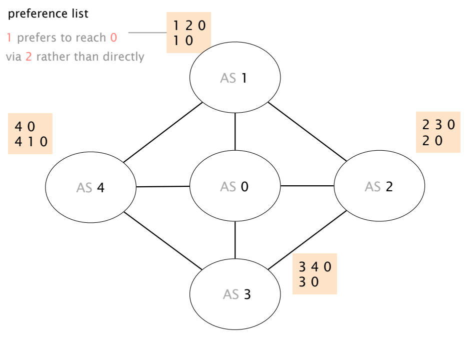
Link `U-I`: cannot exist, since I would need to show up in the first iteration.

Link `A-D`: can exist, weight $[2, \infty]$

Link `E-G`: cannot exist, since G would need to show up in the fifth iteration.

BGP

99.4 Convergence



Does this network ever converge?

Consider this BGP network composed of 5 ASes. Assume that each AS has configured its BGP policies in a way that leads to the preference lists shown in the figure. For example, AS 1 is configured to only accept an announcement for AS 0 if it has path [1,2,0] or [1,0]. In addition, AS 1 prefers the path [1,2,0] over the path [1,0].

Considering that only AS 0 originates prefixes, does that BGP network have a unique, stable solution?

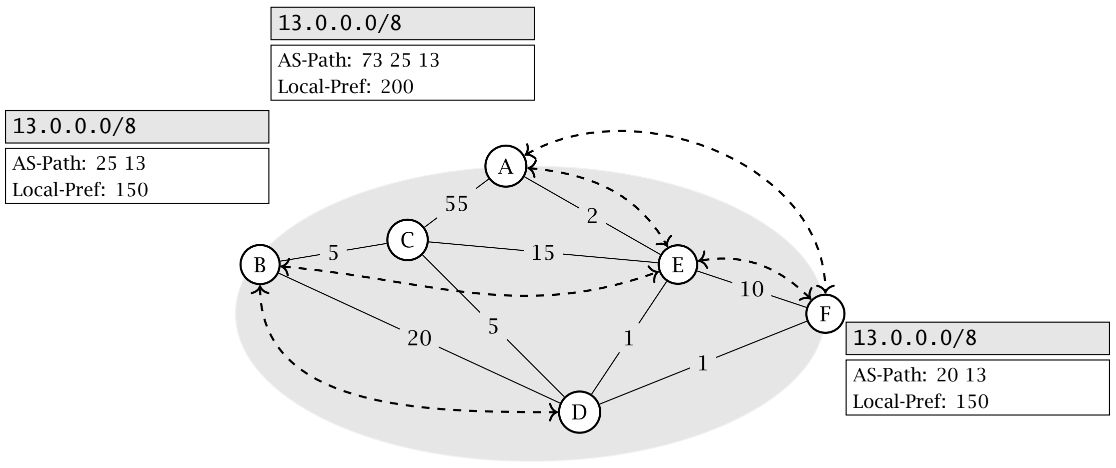
a) If yes, indicate the path that each AS selects in the stable solution.

b) If not, describe an example of oscillation. For instance, by describing a sequence of messages that repeats itself.

Solution: This BGP network does have a unique, stable solution in which:

- AS 1 selects [1,2,0] (preferred path);
- AS 2 selects [2, 0];
- AS 3 selects [3, 4, 0] (preferred path);
- AS 4 selects [4, 0] (preferred path).

99.5 BGP and IGP: Very creative! (Exam Question 2020)



A simple BGP network **not** forming an iBGP full-mesh.

Consider the AS above with three border routers (A, B, F) and three internal routers (C, D, E). All three border routers receive a route announcement for the prefix `13.0.0.0/8` from their eBGP neighbors (not depicted), which they distribute internally. The iBGP sessions are depicted by double-headed dashed arrows (e.g., router A and F maintain an iBGP session). All routers follow the standard BGP decision process. The three border routers have `next-hop-self` configured on all iBGP sessions.

a) For every router, list *(i)* the BGP next-hop, *(ii)* the path taken by the traffic and *(iii)* indicate whether the router's traffic can actually reach the destination. If the next-hop is external, put EXT. If there is no next-hop, put NO.

Solution:

Router	BGP next-hop	Path taken by the traffic	Reachable?
A	EXT	A → EXT	Yes
B	EXT	B → EXT	Yes
C	NO	C → ∅	No
D	B	D → C → ∅	No
E	A	E → A → EXT	Yes
F	A	F → D → C → ∅	No

b) Assume the eBGP session of router A fails and consequently, **the external route of A is not available anymore**. List for every router *(i)* the BGP next-hop, *(ii)* the path taken by the traffic and *(iii)* indicate whether the router's traffic can reach the destination. If the next-hop is external, put EXT. If there is no next-hop, put NO.

Solution:

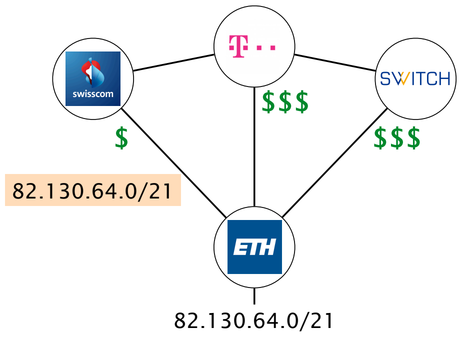
Router	BGP next-hop	Path taken by the traffic	Reachable?
A	F	A → E → D → C → ∅	No
B	EXT	B → EXT	Yes
C	NO	C → ∅	No
D	B	D → C → ∅	No
E	F	E → D → C → ∅	No
F	EXT	F → EXT	Yes

c) The network operator reacted and **added a new iBGP session between routers B and C**. The failure still persists, i.e., the external route of A is not available. List for every router *(i)* the BGP next-hop, *(ii)* the path taken by the traffic and *(iii)* indicate whether the router's traffic can reach the destination. If the next-hop is external, put EXT. If there is no next-hop, put NO.

Solution:

Router	BGP next-hop	Path taken by the traffic	Reachable?
A	F	A → E → D → C → B → EXT	Yes
B	EXT	B → EXT	Yes
C	B	C → B → EXT	Yes
D	B	D → C → B → EXT	Yes
E	F	E → D → C → B → EXT	Yes
F	EXT	F → EXT	Yes

99.6 Traffic Engineering



ETH is connected to three providers with different costs.

Assume that ETH has only one prefix: 82.130.64.0/21. As depicted on the left, the ETH network is connected to three providers (Swisscom, Deutsche Telekom and Switch) and the providers are interconnected with each other. The contract with Swisscom is the cheapest one (indicated by the dollar symbols). For this reason, ETH wants to receive all the incoming traffic over the Swisscom link and therefore announces its prefix only to Swisscom.

a) Do you think that is a good configuration? What happens if the link between ETH and Swisscom fails?

Solution:
Not a good solution. If the link fails, ETH will no longer receive any traffic. ETH is no longer reachable from other networks.

b) To improve the connectivity in case of a link failure between ETH and Swisscom, ETH wants to optimize its announcements. Write down the prefixes which ETH announces to Swisscom, Deutsche Telekom and Switch. During normal operation (no link failure) ETH should still receive all incoming traffic over the Swisscom link.

Solution:
To Swisscom: 82.130.64.0/22 and 82.130.68.0/22 (other splits are also possible)
To Deutsche Telekom: 82.130.64.0/21
To Switch: 82.130.64.0/21

c) After further investigations, ETH decides that only traffic towards 82.130.68.0/23 has to be received over the Swisscom link. All the other traffic can enter over any of the providers. Which prefixes do you have to announce to achieve this traffic distribution?

Solution:
To Swisscom: 82.130.68.0/23 and 82.130.64.0/21
To Deutsche Telekom: 82.130.64.0/21
To Switch: 82.130.64.0/21

Reliable Transport

99.7 TCP Warm-up (Exam Question 2019)

For the following questions answer either with true or false and give an explanation for your decision. Note that in the actual exam you only had to pick true, false or do not give an answer at all (as wrong answers result in point deductions).

a) In contrast to the GBN protocol used in the project, TCP's sequence number often increases by more than one between two consecutive data packets.

Solution: True. The TCP sequence number correlates to the current byte in the transmitted byte stream.

b) A client having an ongoing TCP connection to a server (IP 1.2.3.4, port 80) is not able to start a second TCP connection towards 1.2.3.4:80.

Solution: False. The source port of the two flows can be different to distinguish the two connections.

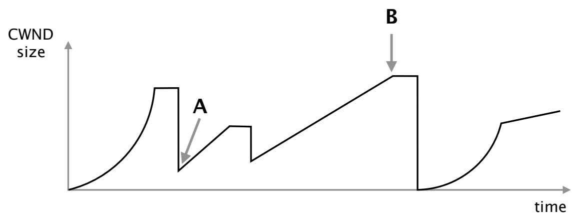
c) A TCP packet with a value of 9 in its header length field (HdrLen, sometimes also called data offset) indicates 16 bytes of TCP options.

Solution: True. HdrLen of 9 indicates 36 bytes. The default TCP header size is 20 bytes (no options) so we have 16 bytes for TCP options.

d) Consider an ongoing TCP flow. Whenever the congestion window increases, the sender can transmit additional data segments.

Solution: False. It also depends on the sender/receiver window sizes. It could e.g., be that the receiver currently cannot handle an additional data segment.

For the following four questions, we consider the congestion window (CWND) evolution observed for a flow f and depicted in the figure below.



e) Flow `f` was in the slow start phase exactly twice.

Solution: True. Once before point A and once after point B.

f) Flow `f` experiences at least one packet loss between time A and B.

Solution: False. The duplicated ACKs could also be due to packet reordering or delays (no packet loss required).

g) In the future, `f` will never be able to experience a higher CWND size than B.

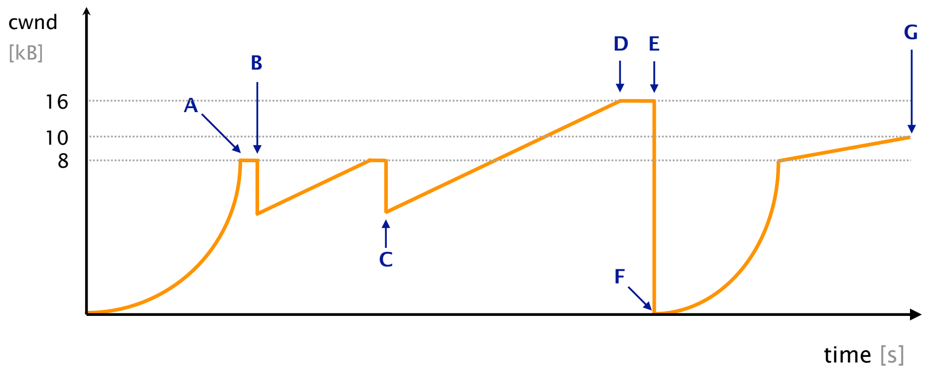
Solution: False. The CWND could reach higher values in the future, for example if other ongoing flows end (more bandwidth available).

h) Consider another flow `f2` starting at exactly the same time as `f` and traversing the exact same path, then `f2` would experience the exact same CWND evolution.

Solution: False. For example the packets of the second flow could be dropped/delayed at different points in time which would result in a different CWND evolution.

99.8 Congestion Window

Consider the following plot which depicts the evolution of the size of the TCP congestion window of the sender.



What kind of network conditions is this flow seeing?

Describe briefly:

a) What happens at point B?
Solution: Triple duplicate ACKs.

b) Does the event happening at point B require the network to discard packets? Why or why not?
Solution: No. This could be caused by packet reordering (e.g., due to queuing or asymmetric paths).

c) What happens at point E?
Solution: A timeout event which causes the sender to decrease its window.

d) Does the event happening at point E require the network to discard packets? Why or why not?
Solution: No. Congestion (queuing delay) in the forward or backward direction could cause the round-trip time to be higher than the retransmission timeout.

Consider that the Maximum Segment Size (MSS) of the connection is 1 kB and the Round-Trip Time (RTT) between the two end points is 100 milliseconds. The sender opens the connection at time $t = 0$. Transmission delay in this network is negligible, so you should only consider the propagation delay in the following.

e) How much time has elapsed at point A?
Solution: Total time: 1 RTT for the TCP handshake + 3 RTT in slow-start ($1 \rightarrow 2 \rightarrow 4 \rightarrow 8$ MSS) = 4 RTT, i.e. 400 ms.

f) How much time has elapsed *between* point C and D?
Solution: The congestion window grows from 4 MSS to 16 MSS linearly at a growth of 1 MSS per RTT, translating to 12 periods of RTT or 1.2 s.

g) How much time has elapsed *between* point F and point G?

Solution: From F, we start with slow start phase to an 8K window size which takes 3 RTT ($1 \rightarrow 2 \rightarrow 4 \rightarrow 8$ MSS). After that we flip to congestion avoidance and follow an additive increase from a 8 to a 10 MSS window size (8, 9, 10 MSS) which takes 2 RTTs. In total 5 RTTs, i.e. 500 ms, elapsed from point F to G.

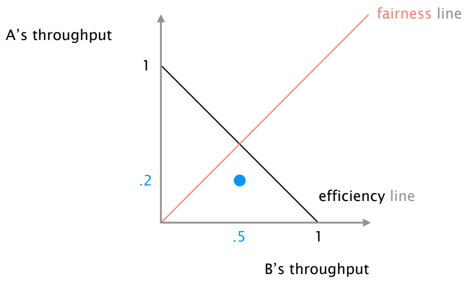
Briefly explain how come point D is higher than point B. Would you expect this to happen often?

Solution: D's height can vary as a consequence of other concurrent flows being sent along the same link. This is therefore likely to happen all the time.

99.9 Fairness

Consider the situation in which two hosts, A and B, are concurrently using a 1 Mbps link with a Maximum Segment Size (MSS) of 100 kb.

Assuming that B starts with 500 kbps and A with 200 kbps (see left picture). Describe the evolution of the throughput of the two hosts when:



Are you getting a fair share?

a) A and B rely on Additive Increase Multiplicate Decrease (AIMD).
Solution: By drawing on the left picture, you can show that AIMD eventually converges to the fairness state. For instance, assuming that both senders increase their CWND by 1 MSS when there is no congestion and divide it by 2 upon congestion, the following points can be drawn:

- (.2, .5)
- (.3, .6)
- (.4, .7) > congestion!
- (.2, .35)
- (.3, .45)
- (.4, .55)
- (.5, .65) > congestion!
- (.25, .325)
- ...

It can be seen that, because of its bigger share, B loses more than A because of the halving, eventually the system evolves along the fairness line.

b) A and B rely on Multiplicative Increase Additive Decrease (MIAD).

Solution: Again, by drawing on the left picture, you can show that MIAD does not converge to the fairness state at all, but rather to one in which A is completely shut down. For instance, assuming that both senders double their CWND when there is no congestion and decrease it by 1 MSS upon congestion, the following points can be drawn:

- (.2, .5)
- (.4, 1) > congestion!
- (.3, .9) > congestion!
- (.2, .8)
- (.4, 1.6) > congestion!
- (.3, 1.5) > congestion!
- (.2, 1.4) > congestion!
- (.1, 1.3) > congestion!
- (0, 1.2) > congestion!
- (0, 1.1) > congestion!
- (0, 1)
- ...

It can be seen that the sender which benefits from a bigger initial share will end up using the entire link.

Assume now that only A is malicious, and wants to cheat congestion control to get more throughput. Describe two distinct ways A could do so and what would be the net effect on B's throughput.

Solution: There are many ways A could cheat congestion control. Two simple ones would be: *i)* increase its congestion window faster than what is prescribed, e.g. instead of increasing its CWND by 1 per RTT, it could increase it by 2 or 3 per RTT; *ii)* opening up many connections and therefore profit from more overall throughput as TCP tends to allocate throughput equally across all connections.

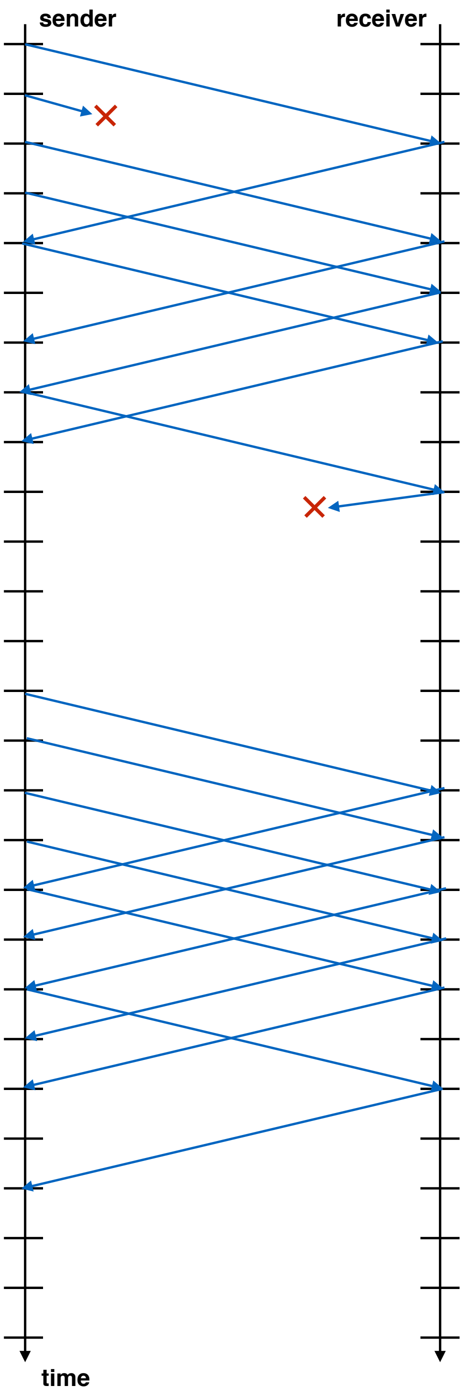
It is worth noting that detecting cheating is hard as controlling all end-points in the Internet is rather hopeless. Yet, the Internet continues to work and we haven't faced a new congestion collapse event since the 80s.

99.10 Go-Back-N (Exam Question 2017)

Consider a Go-Back-N (GBN) protocol with the following implementation choices for sender and receiver.

- The sender and receiver window have a size of 4;
- The receiver saves out-of-order segments in an (infinite) buffer and removes them as soon as the missing segment(s) arrive;
- The receiver uses cumulative ACKs which acknowledge all previous segments and point to the next expected data segment;
- The sender uses Fast Retransmit. After three duplicate ACKs, the sender immediately retransmits the corresponding data segment. For instance, if the sender gets the following ACKs [A1, A1, A1], it will immediately retransmit the data segment D1;
- For each tick in the diagram below, the sender can send one data segment and the receiver can send one ACK. Sender and receiver will first analyze the incoming packet and then send a data segment/ACK;
- The sender uses a retransmission timer of 5 ticks. Each time it sends a data segment or receives an ACK, the timer is reset. After a timeout, the sender retransmits all current segments in its sender buffer (in order, one segment per tick);
- A data segment or ACK needs two ticks to travel to the other end of the connection. See the given start in the diagram.

Your task: Use the diagram on the next page to draw the successful transmission of 6 data segments (D0 to D5) if the **first data segment** (D1, already indicated) **is lost** as well as **ACK A5 is lost the first time it is sent**. For each tick, indicate which data segment or ACK is transmitted (if any) as well as the content of the sender and out-of-order buffer.



sender buffer	data segment	ACK segment	out-of-order buffer
D0	D0	-	-
D0, D1	D1	-	-
D0, D1, D2	D2	A1	-
D0, D1, D2, D3	D3	-	-
D1, D2, D3, D4	D4	A1	D2
D1, D2, D3, D4	-	A1	D2, D3
D1, D2, D3, D4	-	A1	D2, D3, D4
D1, D2, D3, D4	D1	-	D2, D3, D4
D1, D2, D3, D4	-	-	D2, D3, D4
D1, D2, D3, D4	-	A5	-
D1, D2, D3, D4	-	-	-
D1, D2, D3, D4	-	-	-
D1, D2, D3, D4	-	-	-
D1, D2, D3, D4	D1	-	-
D1, D2, D3, D4	D2	-	-
D1, D2, D3, D4	D3	A5	-
D1, D2, D3, D4	D4	A5	-
D5	D5	A5	-
D5	-	A5	-
D5	D5	A6	-
D5	-	-	-
-	-	A6	-
-	-	-	-
-	-	-	-

99.11 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 infinitively 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 have 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.

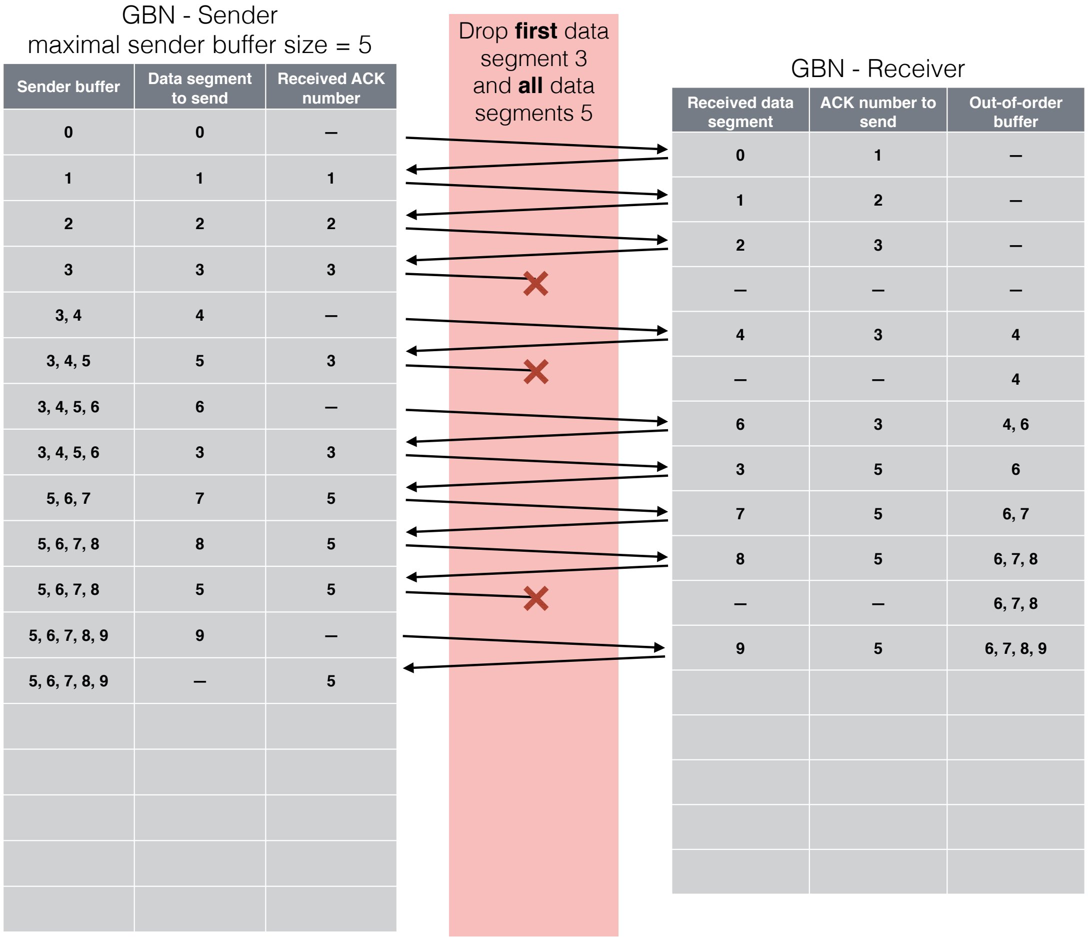
Assumptions:

- 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.

Solution: The acknowledgments always point to the next expected sequence number and not to one of the received segment. This means that, for example, the segment with sequence number 5 is acknowledged with 6.

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

Solution:



GBN - Sender
maximal sender buffer size = 5

Sender buffer	Data segment to send	Received ACK number
0	0	–
1	1	1
2	2	2
3	3	3
3, 4	4	–
3, 4, 5	5	3
3, 4, 5, 6	6	–
3, 4, 5, 6	3	3
5, 6, 7	7	5
5, 6, 7, 8	8	5
5, 6, 7, 8	5	5
5, 6, 7, 8, 9	9	–
5, 6, 7, 8, 9	–	5

GBN - Receiver

Received data segment	ACK number to send	Out-of-order buffer
0	1	–
1	2	–
2	3	–
–	–	–
4	3	4
–	–	4
6	3	4, 6
3	5	6
7	5	6, 7
8	5	6, 7, 8
–	–	6, 7, 8
9	5	6, 7, 8, 9

99.12 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.

IP

99.13 IPv6 Calculations

IPv6 addresses have a slightly different notation. Because the addresses are 128 bit long, we switch from a decimal notation to a hexadecimal one. In general, IPv6 addresses are represented by eight colon-separated blocks of up to four hexadecimal digits each. In a block, leading zeros can be omitted. Furthermore, we can use the "::" symbol to compress one or more consecutive zero blocks. However, the "::" symbol can only be used once in a single IPv6 address. As an example, the IPv6 address: 2001:0db8:0000:0000:0000:ff00:0042:8329 can be simplified to 2001:db8::ff00:42:8329.

a) You are the operator of an enterprise network. Your ISP is giving you a /96 subnet 2001::/96. How many addresses do you have available? Is that a reasonable subnet size compared with the currently available IPv4 addresses?

Solution: You have $2^{128-96} - 1 = 4'294'967'295$ addresses available (no broadcast addresses in IPv6). This is as if you had all IPv4 addresses available exclusively for your enterprise network.

b) In your enterprise network, each host machine is identified by a unique ID starting from 1. Now that you have a lot of IPv6 addresses available, you decide to give each host a unique IP address. The host with ID 1 gets the first IPv6 address in your subnet (2001::1), the host with ID 2 the second IP address and so on. Complete the following table:

Solution:

IPv6 address	**host ID**
2001::5	5
2001::E	14
2001::3A5	933
2001::FFFF:FFFF	4 294 967 295
2001::3:0	196 608

99.14 Detective work

You just started your first job as a network operator of a small network. To get more familiar with the network, you look at a packet trace captured at a switch. The trace contains packets from multiple hosts and one router connected by a (layer 2) switch. The router acts as default gateway, providing access to the Internet and is assigned the first IP address in the subnet. Each row in the following table represents one packet observed at the switch.

SRC MAC Address	DST MAC Address	SRC IP Address	DST IP Address
6a:00:02:49:a1:a0	11:05:ab:59:bb:02	65.222.11.1	65.222.8.2
6a:00:02:49:a1:a0	da:15:00:00:01:11	65.222.11.1	65.222.16.1
da:15:00:00:01:11	11:05:ab:59:bb:02	129.132.103.40	65.222.8.2
11:05:ab:59:bb:02	40:34:00:7a:00:01	65.222.8.2	65.222.15.254
11:05:ab:59:bb:02	ac:00:0a:aa:10:05	65.222.8.2	65.222.9.99
ac:00:0a:aa:10:05	01:05:3c:34:00:02	65.222.9.99	65.222.13.255
6a:00:02:49:a1:a0	da:15:00:00:01:11	65.222.11.1	65.222.8.1

a) Can you identify all the hosts that are part of the local network?

Solution: The local hosts are all the sources and destinations that do not have to go through the default gateway (e.g., their MAC address is not replaced by the MAC address of the router):

- 65.222.11.1
- 65.222.8.2
- 65.222.9.99
- 65.222.15.254
- 65.222.13.255

b) Can you reconstruct the IP subnet used to address the hosts within that local network?

Solution: First, we should note, that the router MAC address is only used for IP sources or destinations outside the local subnet (router is used as gateway) or for packets from/towards the router. With this in mind, we can identify the lowest subnet address from the packets (65.222.11.1 -> 65.222.8.1) and (65.222.11.1 -> 65.222.8.2) as 65.222.8.1. Furthermore, we can infer that 65.222.15.254 still belongs to the local subnet (65.222.8.2 -> 65.222.15.254) but 65.222.16.1 is a destination outside of the network (65.222.11.1 -> 65.222.16.1). We can therefore identify the used subnet as 65.222.8.0/21.

DNS

99.15 Curious students

Consider that ITET has a local DNS server serving the DNS requests for all students' devices connected in the department. How could you determine if an external website has been visited recently by a fellow colleague of yours? Explain.

Solution: You can simply use `dig` to issue a query for any external website you're interested in and observe the response time. If the external website has been visited recently, the response time should be close to immediate (as the local DNS server is located close by). If not, the response time will be slower as the local DNS server would have to initiate a new remote query.

You could also look at the TTL value returned by the local server and compare it to the TTL you get when querying directly the authoritative DNS server for that domain. If the TTL returned by the local server is lower than the one from the authoritative DNS server, you know that the entry has been cached by the local server and hence, someone has visited the website recently.

99.16 Name it or Route it: pick one

In the course, we saw two ways to replicate and load-balance content: *(i)* using Anycast routing; or *(ii)* using DNS.

Any clear winner?

a) List and briefly justify three pros and cons of each.

Solution: DNS

(i) Advantages

i. Simplicity: DNS-based load-balancing can be implemented by any CDN.

ii. (potentially) Fine-grained: Decisions can be particularized on a per-source IP basis.

iii. Near real-time: Assuming small TTL values (modulo the problem of load, see below), DNS load-balancing enables frequent load adaption.

(ii) Disadvantages

i. Infrastructure cost: Good load-balancing requires small TTL values which induce a high load at the DNS server level, which in turn requires to dimension the DNS infrastructure accordingly.

ii. Location: The source IP seen by the server and on which the load-balancing decision is made is the one from the resolver (e.g. Swisscom's one), not the direct client, meaning the resolver and the client can actually be far away from each other (think of Google's open DNS resolver).

iii. (potentially) Coarse-grained: The DNS resolver source IP can actually serve a huge amount of clients which will therefore share the same load-balancing decisions.

Solution: (part 2) Anycast routing

(i) Advantages
   i. Simplicity: The required infrastructure is simple as the load-balancing is done by the network directly.
   ii. Reliability: Whenever a route fails, the network will simply converge to another replica (route).
   iii. Expandablility: It is easy to add an additional replica, as simply an additional location needs to start advertising the respective prefix.

(ii) Disadvantages
   i. Broken Connections: As routing changes can happen at anytime, packets of the same TCP flow might arrive at different replicas effectively breaking the connection.
   ii. Content and performance: The load-balancing is not aware of the utilization of the replicas. Hence, it is not possible to offload work from heavily utilized replicas.
   iii. Location: Packets may not use the nearest replica due to routing policies.

b) We saw that CDNs often rely on DNS for distributing their load. Could they also use Anycast routing instead? Explain why or why not.

Solution: Yes, you can use Anycast routing instead to distribute the load. However, the load-balancing is neither content nor performance aware. You have to make sure that the same content is available on all replicas and none of the replicas is fully utilized.

99.17 Multiple answers

Whenever a client (e.g., your computer) receives multiple IP addresses as answer to a DNS lookup, it picks the very first one. Only if that one does not work, it tries the next one in order.

When you run `dig yahoo.com`, you receive multiple IP addresses as an answer compared to, for example, `dig google.com`.

Can you think of a reason for providing multiple IP addresses? Run the lookup for `yahoo.com` multiple times.

Solution: (i) Resiliency: Should the first IP addresses not be reachable, you can try with the second one without having to do another DNS request.

(ii) Load of the authoritative DNS server: By giving multiple answers, you have a higher chance that one of the answers works for the entire lifetime of the reply (defined TTL value). Your DNS server will therefore get less frequent requests compared to a DNS server which only answers with one entry and a low TTL value.

(iii) Load balancing: You can use it as one way to do load balancing.