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.

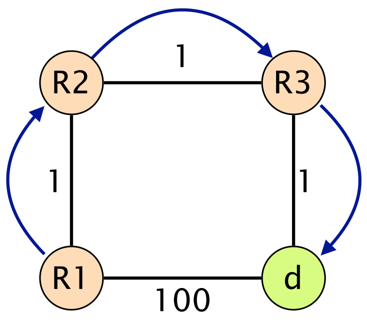
Solution: True. Longer paths will see a bigger increase than shorter ones.

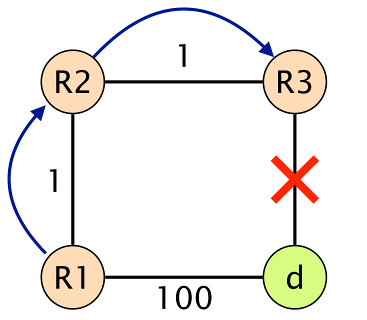
99.2 Dijkstra's Algorithm with Link Failure

The routers in the network on the top left use Dijkstra's Algorithm to find the shortest path towards destination d. You can assume that every router knows the entire network graph. In case of a failure, the routers directly affected by it inform the other routers by flooding this information in the network.

The blue arrows indicate how the routers forward the traffic: $R2$, for example, sends packets for destination d to router $R3$.

Now the link between router $R3$ and d fails (network at the bottom left) and $R3$ can no longer send packets towards destination d. As $R3$ is directly connected to the failed link, it detects the failure immediately. It starts to flood this information, such that all the routers can update their network view and recompute Dijkstra's algorithm.





Dijkstra's algorithm with a link failure

a) What is the new shortest-path from $R3$ towards destination d?

Solution: $R3, R2, R1, d$

b) Assume now that the computation of the new shortest-path is *very* fast and finishes before $R3$ starts the flooding of the messages announcing the link failure. $R3$ sends a packet towards d using the new shortest-path. Will the packet reach its destination? Which path will it take?

Solution: No, $R2$ does not yet know about the link failure and did not update its shortest-path towards d. It will send the packet back towards $R3$. The packet is therefore stuck in a forwarding loop.

c) Can you find a sequence of link failure messages and shortest-path computations such that the problem discovered in the previous task is observed between $R1$ and $R2$? The *only* link failure is still between router $R3$ and d.

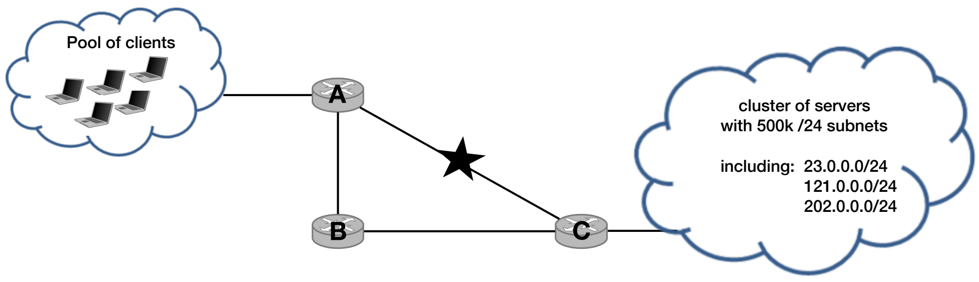
Solution: $R2$ did receive a link failure message and updated its shortest-path. $R2$ then sends a packet towards d to the next-hop $R1$ before $R1$ can update its shortest-path.

d) As we have discovered, the order in which the link failure messages are processed and the new shortest-paths are computed is crucial for a correct forwarding behavior in the network. In which order should the router update their forwarding tables, such that the previously observed problems will not occur? Can you find a more general "rule" for a save ordering of forwarding rule updates?

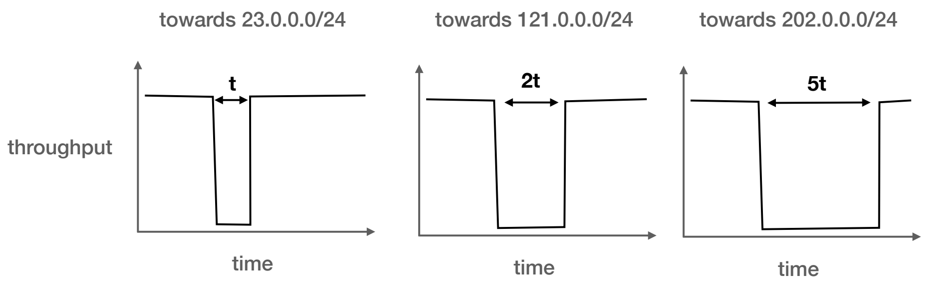
Solution: Good order: $R1, R2, R3$. To prevent forwarding loops, the routers should update their forwarding tables based on their distance to the destination. The router nearest to the destination should update its forwarding table first.

99.3 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.

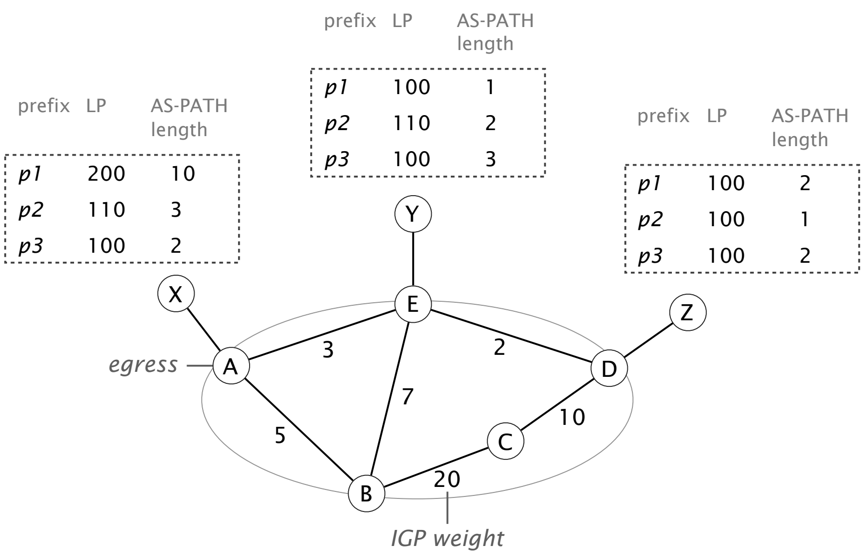
b) Describe two solutions to speed up the convergence for these three prefixes.

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.

99.4 Putting Everything Together (Exam Question 2016)

Consider the ISP network composed of 5 routers (A, B, C, D, E) depicted in the Figure below. Three of these routers, A, E and D, are connected to routers located in neighboring ASes via eBGP. These neighboring routers are indicated by X, Y and Z. Each of them advertises the same three distinct IP prefixes $p1$, $p2$ and $p3$.

The three tables in the Figure indicate the Local-Preference (LP) associated to each external prefix by A, E and D along with their corresponding AS-PATH length. For instance, A learns a route to $p1$ from X with an AS-PATH length of 10 to which it associates a LP of 200. Internally, the ISP uses an iBGP full-mesh to distribute the BGP routes and OSPF as intra-domain routing protocol. The weight of each internal link is indicated next to it.



prefix	LP	AS-PATH length
p1	200	10
p2	110	3
p3	100	2

prefix	LP	AS-PATH length
p1	100	1
p2	110	2
p3	100	3

prefix	LP	AS-PATH length
p1	100	2
p2	100	1
p3	100	2

An ISP network which receives BGP routes for 3 external prefixes ($p1$, $p2$, $p3$) from 3 routers (X, Y, Z) in neighboring ASes.

For each router in the ISP, indicate the router ID of the selected egress (A, E, D) along with the router ID of the internal next-hop (A, B, C, D, E or *direct*) used to reach it. For that you can use the tables on the next page. You can assume that A, E and D use the next-hop-self configuration.

Solution:

To solve this question, follow the BGP decision algorithm for the given values. For the same prefix, BGP will prefer the route with the higher local preference. If the preferences are equal, BPG picks the route with the lower AS-PATH length. Should the path length be equal as well (e.g., **p3** route from X and Z), the routers will pick the route with the shortest path to the next-hop. This means, the routers will pick the best route based on the IGP (OSPF) path with the lowest cost.

The internal next hop is found by figuring out how the packets are forwarded (shortest path) to reach the corresponding egress point.

A

prefix	egress	internal NH
p1	A	direct
p2	E	E
p3	A	direct

B

prefix	egress	internal NH
p1	A	A
p2	E	E
p3	A	A

C

prefix	egress	internal NH
p1	A	D
p2	E	D
p3	D	D

D

prefix	egress	internal NH
p1	A	E
p2	E	E
p3	D	direct

E

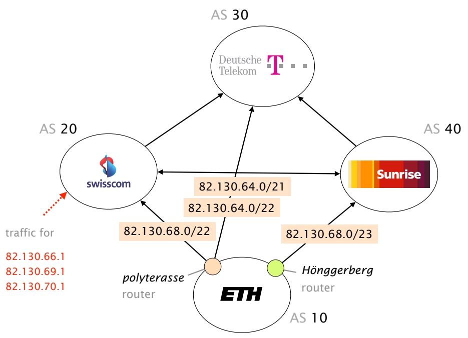
prefix	egress	internal NH
p1	A	A
p2	E	direct
p3	D	D

Solution to task 3 c).

99.5 Traffic (not so much) Engineered

After passing the Communication Networks exam with flying colors, ETH hires you as a junior network engineer. *Congrats!*

Your first mission is to analyze their BGP configuration. They indeed suspect that something might be wrong, especially since they installed this box from Sisco Systems that automatically configure BGP announcements according to Traffic Engineering objectives. For the sake of simplicity, assume again that ETH has only one prefix: 82.130.64.0/21 and three providers: Swisscom, Deutsche Telekom and Sunrise. The actual announcements are depicted on the left. Customers are drawn below their providers (Swisscom is a customer of Deutsche Telekom), while peers are drawn next to each other (Swisccom is a peer of Sunrise).



Where are my packets going?

Consider the incoming traffic from Swisscom. What path is taken for packets destined to:

a) 82.130.66.1? **Solution:** [20, 30, 10]

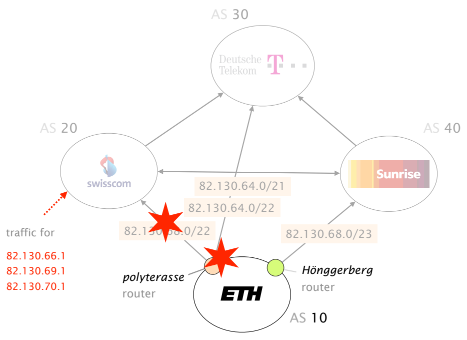
b) 82.130.69.1? **Solution:** [20, 40, 10]

c) 82.130.70.1? **Solution:** [20, 10]

Are Swisscom, Deutsche Telekom and Sunrise happy about these announcements? Can they do anything about that? Explain briefly.

Solution: Swisscom and Sunrise are not happy. Should ETH advertise them its full prefix, they could direct all traffic to it directly and earn money instead of paying money to Deutsche Telekom or use their peering (revenue neutral) link. Deutsche Telekom is happy though as it receives extra traffic (and therefore, revenues) from Swisscom and Sunrise for traffic pertaining to the sub-prefixes.

As routing and forwarding in the Internet is done according to the longest destination prefix, there is nothing that Swisscom or Sunrise can do besides convincing ETH to change its announcements.



Is this network as redundant as it looks?

As the *polyterasse* router is getting older, its reliability starts to suffer. In theory, this should not be a big problem as ETH is *triple*-homed! Yet, in practice, the ETH engineers observe regular network connectivity upon failures.

Assuming the same announcements, what path ends up being taken by the packets destined to the above three IP addresses when:

a) the link between *polyterasse* and Swisscom goes down?
   (i) 82.130.66.1? **Solution:** [20, 30, 10]
   (ii) 82.130.69.1? **Solution:** [20, 40, 10]
   (iii) 82.130.70.1? **Solution:** [20, 30, 10]

b) the *polyterasse* router dies?
   (i) 82.130.66.1? **Solution:** *drop*
   (ii) 82.130.69.1? **Solution:** [20, 40, 10]
   (iii) 82.130.70.1? **Solution:** *drop*

What would you change in the ETH announcements to improve reliability, without disturbing the inbound Traffic Engineering performed by the Sisco box in the steady case (without failures)? Explain briefly.

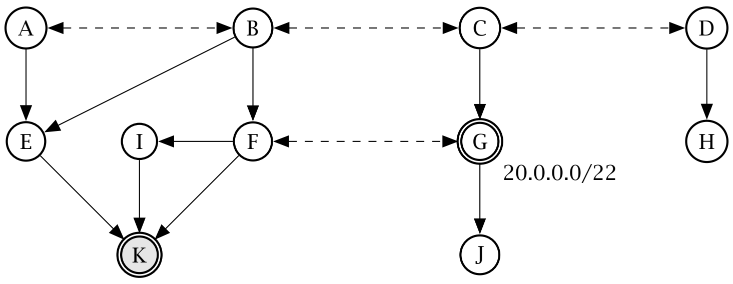
Solution: A simple solution would be to advertise the full prefix 82.130.64.0/21 to all providers *along* with the more-specific ones. Since traffic is based on the longest-prefix match, announcing additional less-specific prefix will not change the forwarding in the steady state but will enable any of its three providers to continue providing full transit in case of failures.

99.6 BGP Hijack (Exam Question 2018)

Consider the Internet topology consisting of 11 Autonomous Systems (ASes) in the Figure below. Single-headed plain arrows point from providers to their customers (AS *A* is the provider of AS *E*), 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 (alphabetical order).

AS *G* is the origin of prefix 20.0.0.0/22 and advertises it to its neighbors. Independently of what the external advertisements are, AS *G* ***always*** prefers its internal route to reach any IP destination in 20.0.0.0/22.



An Internet topology with 11 ASes. AS *K* aims at hijacking traffic destined to AS *G*.

a) AS *K* wants to hijack all the traffic going to AS *G* for 20.0.0.0/22. It starts advertising the exact same prefix. From which ASes is it able to attract the traffic?

Solution:
A, *B*, *E*, *F*, *I*.

b) AS *K* is not satisfied by the result. What can it do to attract traffic destined to AS *G* from more of the ASes? List the ASes from which it is able to attract the traffic and explain why this works.

Solution:
AS *K* can break the prefix into more specific prefixes, for example 20.0.0.0/23 and 20.0.2.0/23. We can consider the two prefix lengths (/23 and /22) separately since the traffic will always follow the route for the most specific prefix independent of the other less specific routes:

First, we look at the two /23 prefixes and which ASes receive the route from *K* for them: *A*, *B*, *C*, *E*, *F*, and *I*. Given that these ASes now know routes for the two /23 prefixes, they will forward their traffic according to the /23 routes as the forwarding decisions are based on the longest-prefix match. Hence, the traffic of *A*, *B*, *C*, *E*, *F*, and *I* is hijacked by *K* through the more specific routes. Note, *C* will not advertise the /23 routes to its peer *D* as it already received the route from its other peer *B*.

Second, we can look at the original /22 prefix advertised by *G*. AS *C* receives an announcement for the /22 route from its customer *G*. It will advertise this route to both of its peers: *B* and *D*. For *B*, the announcement has no effect as it knows a more specific route. However, *D* does not have any other route for the address space of *G* and hence will use the /22 route as its best path. In addition, it will advertise the route to its customer *H*. *D* thinks that its traffic goes directly via *C* to *G*. In reality, the traffic from *D* goes to *C* where it will follow the more specific route and end up at hijacker *K*. Therefore, *K* is able to hijack the traffic of *A*, *B*, *C*, *D*, *E*, *F*, *H*, and *I*. It only fails to hijack the traffic of *G* and *J*.

c) The ASes from which AS *K* manages to attract the traffic realize what is happening as all their traffic to 20.0.0.0/22 goes to a dead-end (AS *K*).

Show how AS *K* could still deliver the traffic to the real destination (AS *G*) by poisoning the AS path while attracting as much traffic as possible. In addition, list the ASes from which it can attract the traffic.

Solution:
BGP has a built-in loop detection mechanism. Every router checks the AS path when it receives a route announcement. If the AS path contains its own AS number, the router rejects the route as a routing loop could be created. One can “abuse” this mechanism by adding specific AS numbers to the AS path. This is called poisoning.

The hijacker could therefore add the AS number of certain ASes to the AS path of its hijack advertisements. This will trigger the BGP loop detection of those ASes where the hijack advertisements are immediately dropped. How can that help the hijacker? By doing so, certain ASes will still prefer the route from the original origin. These ASes can then be used to keep a path open to the destination and to forward the hijacked traffic to its original destination.

Applying that to the question, AS K can poison the AS path of the two /23 prefixes with AS F, i.e. prepending F once. Therefore, F will drop the /23 prefixes as it detects a loop in the AS path. F still prefers the original route towards G. K can now simply forward the hijacked traffic to F using the direct link and a static route, for example. Note that all the other ASes still handle the /23 prefixes as before. Therefore, K will be able to hijack traffic from A, B, C, D, E, H, I. All the hijacked traffic is then forwarded to G making the ongoing hijack much more difficult to detect.

d) Can you think of a different way for AS K to achieve similar results as in *c)* without poisoning the AS path? Explain.

Solution:
Instead of poisoning, AS K could advertise the **/22 prefix** only to a subset of its neighbors ensuring that at least one path remains open towards the original destination. AS K could advertise the prefix only to E attracting the traffic from A, B and E, while keeping a path via F to G. It cannot advertise the prefix to I as F would always prefer the route from I over the one from G therefore dismissing the return path. Also, AS K *cannot* advertise the more specific /23 prefixes as AS F would then send its traffic via B to K.

In general, please note that in this question the hijacker has complete knowledge of the entire "Internet" as well as all the details about business relationships between the ASes. Clearly that is not the case in the real Internet making such targeted attacks more difficult.

Reliable Transport

99.7 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.

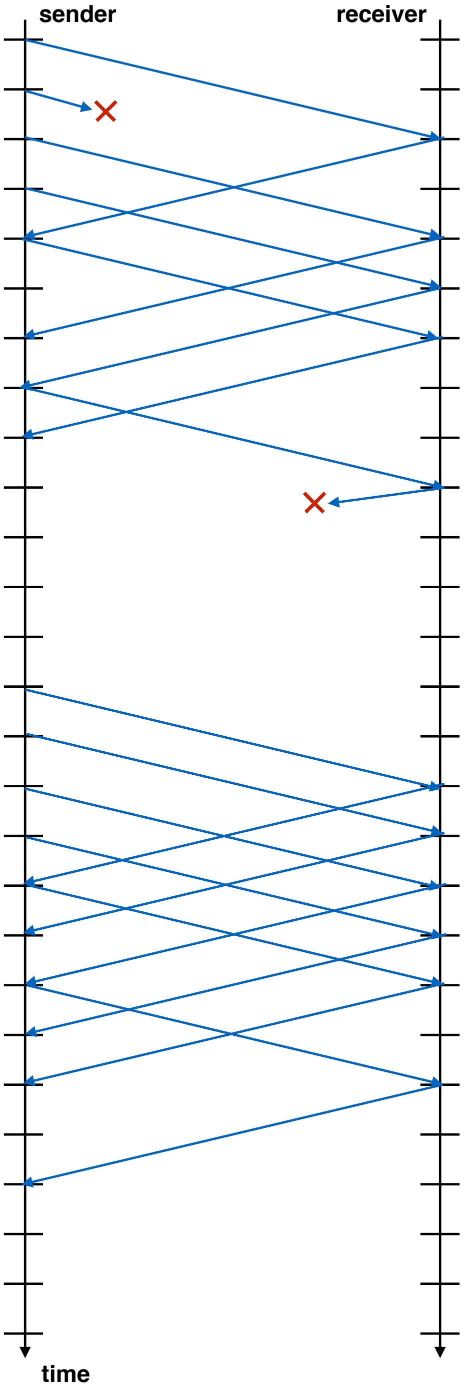
99.8 Go-Back-N (Exam Question 2017)

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.

Solution:



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	-
-	-	-	-
-	-	-	-

IP, DHCP, ARP

99.9 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:

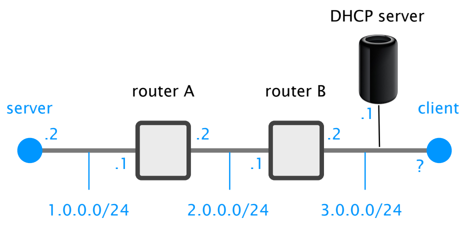
Solution:

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

99.10 Putting everything together

Consider the network on the left composed of three Ethernet segments separated by two intermediate routers (A and B). In this network, the server's interface along with the routers' interfaces are configured with static IP addresses. While clients connected to the 3.0.0.0/24 Ethernet segment obtain an IP address via DHCP.

Assuming that the client has just started, with a perfectly empty state, precisely describe all packets that are generated when the command "ping 1.0.0.2" is issued (until the server answers back). Among others, your answer *must* include the content of the Layer 2 and Layer 3 headers.



Describe everything that happens to the packets sent between the client and the server

Solution:

- src_mac: *client_mac*, dst_mac: *broadcast*
  src_ip: *0.0.0.0*, dst_ip: *255.255.255.255*
  payload: *DHCP discovery*
- src_mac: *dhcp_server_mac*, dst_mac: *broadcast*
  src_ip: *3.0.0.1*, dst_ip: *255.255.255.255*
  payload: *DHCP offer for 3.0.0.3*

  Note: In addition to the IP address of the client (arbitrarily picked by the DHCP server), the DHCP offer also contains the gateway (3.0.0.2) to use. The DHCP offer can be sent either in broadcast or unicast (both answers accepted). Broadcast is often used as some network stacks will not accept "unicasted" frames unless an IP address is configured first.
- src_mac: *client_mac*, dst_mac: *broadcast*
  payload: *ARP request: Who has 3.0.0.2? Tell 3.0.0.3.*
- src_mac: *routerB_right_mac*, dst_mac: *client_mac*
  payload: *ARP reply: 3.0.0.2 is at routerB_right_mac*
- src_mac: *client_mac*, dst_mac: *routerB_right_mac*
  src_ip: *3.0.0.3*, dst_ip: *1.0.0.2*
  payload: *ICMP echo request*

... (rest of the packets are omitted)

- src_mac: *server_mac*, dst_mac: *routerA_left_mac*
  src_ip: *1.0.0.2*, dst_ip: *3.0.0.3*
  payload: *ICMP echo reply*

99.11 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):

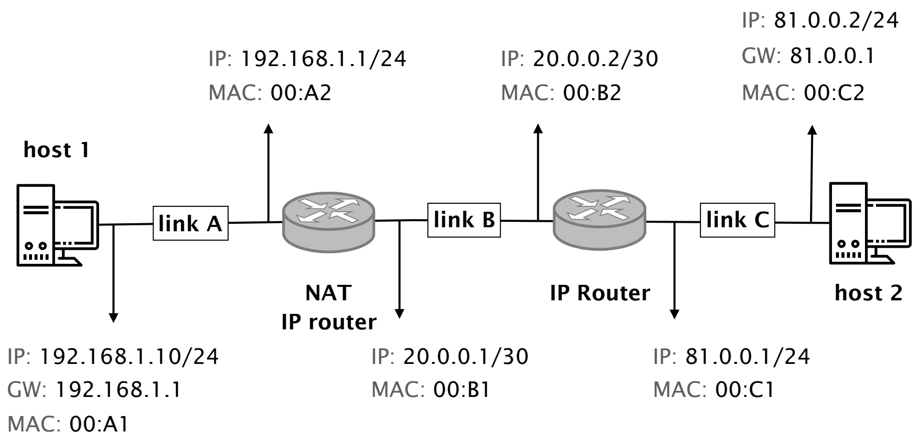
- 65.222.11.1
- 65.222.8.2
- 65.222.9.99
- 65.222.15.254
- 65.222.13.255

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.

99.12 Changing addresses (Exam Question 2019)

Consider the network depicted in the Figure below which is composed of two hosts along with two routers, one of which acts as Network Address Translator (NAT). Host 1 is located in a private subnet (192.168.1.0/24) and uses 192.168.1.1 as gateway, while host 2 is located in a public subnet (81.0.0.0/24) and uses 81.0.0.1 as gateway. The Figure below also depicts the MAC address of each of the 6 interfaces connected at either end of the three links. The NAT/router performs address translation between the private and the public subnets, translating traffic originating from private IPs to its public one (here, 20.0.0.1), and vice-versa.



A network topology relying on Network Address Translation.

a) Consider that host 1 tries to open a TCP connection with host 2 on port 80 using 1337 as (random) source port. Write down a possible sequence of packet headers observed at each link for the first two packets (i.e., the SYN sent by host 1, and the SYN/ACK sent by host 2). Fill in the table below to answer. Assume that hosts and routers have the required MAC addresses in their ARP table.

Solution:

	src MAC	dst MAC	src IP	dst IP	src TCP port	dst TCP port
link A	00:A1	00:A2	192.168.1.10	81.0.0.2	1337	80
link B	00:B1	00:B2	20.0.0.1	81.0.0.2	rand	80
link C	00:C1	00:C2	20.0.0.1	81.0.0.2	rand	80
link C	00:C2	00:C1	81.0.0.2	20.0.0.1	80	rand
link B	00:B2	00:B1	81.0.0.2	20.0.0.1	80	rand
link A	00:A2	00:A1	81.0.0.2	192.168.1.10	80	1337

b) Could host 2 initiate a TCP connection to host 1? Briefly explain why/why not.

Solution: That is not possible as host 1 is behind a NAT. Packets that reach the public IP of the NAT will not be forwarded to host 1 as there is no corresponding NAT rule available.

DNS

99.13 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.

99.14 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.

99.15 Local vs. authoritative DNS server

Perform a DNS query for `uzh.ch` using first the authoritative DNS server (`ns1.uzh.ch`) and then your local server.

Note: When using `nslookup` on Windows, you need to specify the `-debug` flag to get the relevant information for this task. For example:

```
nslookup -debug <Domain Name> <DNS-Server>
```

- Compare the `ANSWER SECTION` of the responses. Can you see differences between the answers from your local DNS server and the authoritative server? Run the query to your local server multiple times to make the differences more obvious.

  Solution: The answers differ in the time to live (TTL). While the TTL is constant in the replies from the authoritative DNS server, it varies in the replies from the local server.

- What is the reason for this difference?

  Solution: The local DNS server caches replies to requests. To ensure that it does not keep outdated information in its cache, each authoritative name server attaches a TTL to its replies. The TTL tells the local DNS server how long it can store the reply in the cache and use it to reply to requests.

- As you have seen in the lecture, DNS can be used to balance the incoming load. What are the considerations one has to make when using DNS load balancing with respect to the TTL?

  Solution: With low TTLs we can ensure that we can shift the load quickly. However, low TTLs also mean that our authoritative DNS server will get many more requests.

HTTP (and Web)

99.16 Loading a website (Exam Question 2017)

The website `www.your-shop.ch` consists of the following elements:

- HTML `www.your-shop.ch/index.html`
- Stylesheet `www.your-shop.ch/style.css`
- image `www.your-shop.ch/logo.png`
- image `images.your-shop.ch/product.jpg`
- Facebook like button `cdn.facebook.com/like.png`
- Facebook “tracking” code `www.facebook.com/track.js`
- Google “tracking” code `www.google.com/track.js`

a) Assuming that your host is configured to use a local recursive DNS server in your network and all caches are empty. List all the DNS queries that your host sends to this DNS server when you open up `https://www.your-shop.ch/` in your favorite browser.

Solution: The host will send one query per domain:

- `www.your-shop.ch`
- `images.your-shop.ch`
- `cdn.facebook.com`
- `www.facebook.com`
- `www.google.com`

b) After loading the website, you send an email to `contact@your-shop.ch` via a mail server that uses the same DNS server as your host. Does the local recursive DNS server need to run additional queries to other DNS servers if it has all the replies from the queries in the previous task in its cache? Explain why or why not.

Solution: Yes. The queries in the previous task asked for A-entries. The mail server needs an MX-entry for `your-shop.ch`, which was not queried before and is therefore not in the cache.

c) How many TCP connections would an unoptimized browser (also referred to as “naive” in the lecture) open to load `https://www.your-shop.ch/`? Briefly explain your answer.

Solution: It will require 7 TCP connections: one connection per object.

d) During your holidays in Australia, you realize that the Facebook like button loads much faster than the logo of the shop even though both images have the same size. Can you explain the reason for this and why you do not observe this behavior in Switzerland?

Solution: The Facebook button is hosted in a CDN and can be loaded fast from the Australian server of the CDN. The shop logo is not loaded from a CDN but from the Shop’s webserver (which could be in Switzerland).

e) One hour later (still in Australia), you open the shop’s website again. This time, the logo of the shop and the Facebook button appear at the same time. Explain **two distinct** reasons that would justify this behavior.

Solution: The logo could have been moved to a CDN or it is cached by the browser.