ETH zürich

Institut für Technische Informatik und Kommunikationsnetze

Department ITET
August 2022

Prof. Dr. Laurent Vanbever
C. Busse-Grawitz, T. Bühler, A. Dietmüller, T. Schneider

Exam: Communication Networks

27 August 2022, 09:30–12:00, Room HIL G 41

Sample Solution

General remarks:

- ▷ Write your **name** and your **ETH student number** below on this front page and **sign it**.
- ▷ Put your **legitimation card** on the top right corner of your desk. Make sure that the side containing your name and **student number** is visible.
- ▷ Check that you have received **all task sheets** (Pages **1 – 39**).
- ▷ Do **not separate** the **task sheets** as we collect the exams **only after you left** the room.

- ▷ Write your answers directly on the task sheets.
- ▷ **All answers fit within the allocated space and often in much less.**
- ▷ If you need more space, use the three extra sheets at the **end of the exam**. Indicate the **task** in the corresponding field.

- ▷ **Read each task completely before you start solving it**.
- ▷ **For the best mark, it is not required to score all points.**

- ▷ Please answer either in **English or German**.
- ▷ **Write clearly** in blue or black ink (not red) using a **pen**, not a pencil.
- ▷ **Cancel** invalid parts of your solutions **clearly**.
- ▷ At the end of the exam, **place the exam face up on the top left corner** of your desk. Then collect all your belongings and **exit the room** according to the given instructions.

Special aids:

- ▷ All written materials (vocabulary books, lecture and lab scripts, exercises, etc.) are allowed.
- ▷ Using a calculator is allowed, but the use of electronic communication tools (mobile phone, computer, etc.) is strictly forbidden.

Family name: Student legi nr.:

First name: Signature:

Do not write in the table below (used by correctors only):

Task	Points
Ethernet & IP	/31
Intra-domain routing	/25
Inter-domain routing	/39
Reliable transport	/36
Applications	/19
Total	/150

leave blank

Task 1: Ethernet & IP — 31 Points

leave blank

a) Warm-Up — (6 Points)

(i) Name one scenario where a single packet may have two different IP headers. (1 Point)

Solution: This could be a VPN or something like 6in4 tunneling.

(ii) Explain why a router (a layer 3 device) also needs to be able to parse and modify MAC addresses (layer 2 header fields) while forwarding IP packets. (1 Point)

Solution: For example, a router needs to adapt MAC addresses to match link endpoints.

(iii) Can you use the `ping` command to ping a layer-2 switch, a layer-3 router, or both? (1 Point)

Solution: You can only ping a router.

(iv) Which protocol do layer-2 networks commonly use to avoid forwarding loops? (1 Point)

Solution: The spanning-tree protocol.

(v) What are the network address, the broadcast address, and the usable range of host IPs of the IP address `56.32.122.3/18`? (1 Point)

Solution: Network: `56.32.64.0` (also accepted: `56.32.64.0/18`)

Broadcast: `56.32.127.255`

Range: `56.32.64.1`–`56.32.127.254`

(vi) What does best-effort delivery mean? (1 Point)

Solution: One possible solution: packets are not guaranteed to arrive. Other solutions might also consider packet correctness, checksums, and so on.

leave blank

b) From the Bottom Up **(9 Points)**

Consider the ETH and SWITCH networks in Figure 1, connected via two routers. All circles represent layer-2 switches.

The DHCP server connected to the HG switch in the ETH network assigns unused IPs in the private ETH prefix `10.132.0.0/16` and also returns the default gateway: `10.132.0.1`.

The ETH router uses Network Address Translation (NAT) to translate IPs in the private ETH prefix to the public IP (`1.2.3.4`), and randomly assigns a port in the range `1000`–`2000`.

Throughout this question, you may abbreviate MAC addresses, e.g. write `11` instead of `11:11:11:11:11:11`.

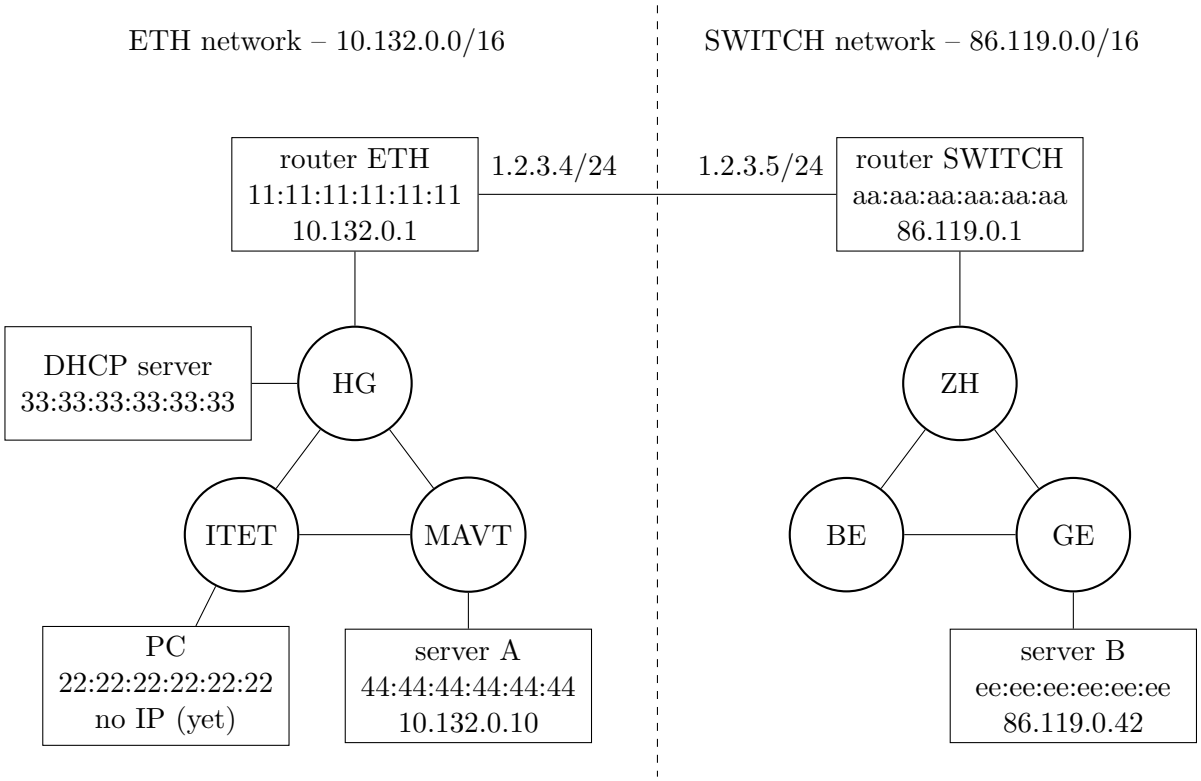

Figure 1: The ETH and SWITCH networks.

(i) You enable DHCP on your PC that you just connected in the ITET building. Answer the following questions: (2 Points)

Solution:
What does your PC's first DHCP request look like?

Src MAC	22
Dst MAC	ff
Request (summarize)	I want an IP address.

Table 1: DHCP request.

Which DHCP response will you receive (include all relevant information as precisely as possible)?

Src MAC	33
Dst MAC	22 (also accepted: ff)
Response (summarize)	Use IP address `10.132.0.2/16` (or any other IP in the ETH network) Use as default GW: `10.132.0.1`

Table 2: DHCP response.

(ii) Next, you want to send a packet to server A (`10.132.0.10`) and server B (`86.119.0.42`) using their (known) IP addresses. For each of these servers, your host first needs to send an ARP request. Fill in the tables for the two requests and explain the differences between them: (3 Points)

Solution:

Src MAC	22
Dst MAC	ff
Request (summarize)	Who has `10.132.0.10`?

Table 3: ARP request for server A

Src MAC	22
Dst MAC	ff
Request (summarize)	Who has `10.132.0.1`?

Table 4: ARP request for server B

The difference between the requests is the requested IP. Server A is in the same subnet as the PC, so it requests the MAC address for the IP address of server A. Server B is not in the subnet, so the PC instead needs to send the request to the default gateway, so it requests the MAC address for the IP address of the gateway.

(iii) Finally, your host sends a packet to server B. The table below shows all the devices on the path the packet takes. Complete the table by adding the missing MAC and IP addresses as well as the ports (the initial ports are already given to you). For each table row, write down the header values **just after the packet exits the indicated device**. Your PC still knows the MAC addresses from the previous question. (4 Points)

Solution:

	MAC		**IP**		**Port**	
Device	**src**	**dst**	**src**	**dst**	**src**	**dst**
ITET	22	11	`10.132.0.2`	`86.119.0.42`	5678	4321
HG	22	11	`10.132.0.2`	`86.119.0.42`	5678	4321
Router ETH	11	aa	`1.2.3.4`	`86.119.0.42`	1111	4321
Router SWITCH	aa	ee	`1.2.3.4`	`86.119.0.42`	1111	4321
ZH	aa	ee	`1.2.3.4`	`86.119.0.42`	1111	4321
GE	aa	ee	`1.2.3.4`	`86.119.0.42`	1111	4321

Table 5: The life of a packet

leave blank

c) Spanning Tree Protocol **(8 Points)**

Consider the layer-2 network with 8 switches (circles) in Figure 2. Each link in the network has a capacity of 1 Gbps. Attached to the switches are three senders and three receivers:

- Sender 1 (S_1) tries to send 1 Gbps of traffic to Receiver 1 (R_1).
- S_2 tries to send 1 Gbps of traffic to R_2.
- S_3 tries to send 1 Gbps of traffic to R_3.

The switches in the network run the Spanning Tree Protocol (STP). If there exist multiple shortest paths to the root node, a switch picks the next hop which has the lower switch ID.

Your goal is to maximize the aggregated throughput between the three senders and receivers.

(i) Apply the STP to the network in Figure 2. Cross out all links which the STP disables. What is the aggregated throughput between the three senders and receivers? (3 Points)

Solution:

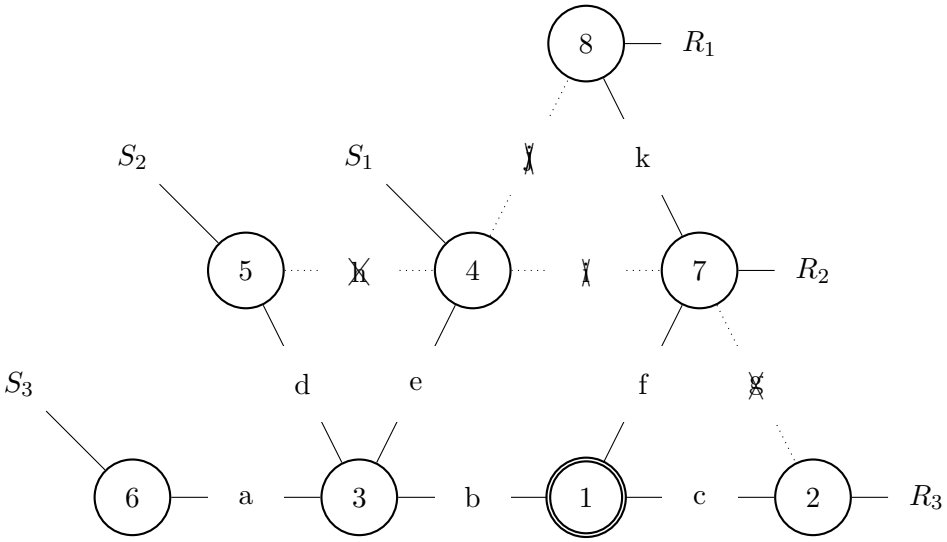

Figure 2: Solution.

Aggregated throughput: 1 Gbps

(ii) You can now exchange IDs of two switches to increase throughput, however, the root of the spanning tree should not change (i.e., the switch with ID 1 cannot move). Which two IDs do you exchange and what is the new aggregated throughput? (2 Points)

Solution: The key is to swap the IDs to result in more favorable tie-breaks. Exchange IDs 3 and 7. This way, the traffic between S_1 and R_1 can use a separate path. Alternative solutions: exchange IDs 2 and 7, or exchange IDs 3 and 8.

Aggregated throughput: 2 Gbps

(iii) You are still not happy with the throughput. You can now also physically remove *two* links $(a, b, \ldots, k)$ from the network in Figure 2. Which two links do you remove and what is the aggregated throughput once the spanning tree has adapted? Continue with the spanning tree from task **(ii)**. (3 Points)

Solution: Depending on the workload, the shortest paths may not always be the best. By removing links, we can better balance traffic for this particular workload. Remove links d and k. Aggregated throughput (assuming ID placement from (ii)): 3 Gbps.

Alternative solution: remove links f and g. Aggregated throughput 3 Gbps.

leave blank

d) Forwarding Tables **(8 Points)**

You receive the forwarding table in Table 6. Each /16 prefix has a next hop from 1 to 4. Leverage longest-prefix matching to reduce the forwarding table to as few entries as possible. All existing addresses in the table should still be forwarded to the same next hop. Additionally, you can add a single default route (`0.0.0.0/0`). Pick the next hop of the default route in such a way that you achieve the highest forwarding table compression. Write your solution directly into Table 7.

Prefix	Next hop
142.112.0.0/16	2
142.113.0.0/16	2
142.114.0.0/16	2
142.115.0.0/16	2
142.116.0.0/16	1
142.117.0.0/16	2
142.118.0.0/16	3
142.119.0.0/16	3
142.120.0.0/16	4
142.121.0.0/16	4
142.122.0.0/16	3
142.123.0.0/16	4
142.124.0.0/16	2
142.125.0.0/16	1
142.126.0.0/16	4
142.127.0.0/16	2

Table 6: Original IP forwarding table.

Prefix	Next hop
0.0.0.0/0	

Table 7: Optimized forwarding table with a default route. Note, you might not need all the rows.

Solution: The best solution results in 7 entries. See below.

Prefix	**Next hop**
0.0.0.0/0	2
142.116.0.0/16	1
142.118.0.0/15	3
142.120.0.0/14	4
142.122.0.0/16	3
142.125.0.0/16	1
142.126.0.0/16	4

Table 8: Solution

leave blank

Task 2: Intra-domain routing **25 Points**

leave blank

a) Warm-Up **(6 Points)**

(i) In one sentence, explain which problem "poisoned reverse" solves in a distance-vector protocol. (1 Point)

Solution: It solves the "count-to-infinity" problem, i.e., very slow convergence due to temporary routing loops.

(ii) How would you set the link weights in a network such that the computed shortest paths have the highest bandwidths? (1 Point)

Solution: A good approach (which works in most cases) would be to take the inverse of the bandwidth as link weights.

(iii) Explain why a global forwarding state is not valid if it only prevents dead ends. (1 Point)

Solution: Even without dead ends, traffic could still be stuck in a forwarding loop. Therefore, packets would never reach the intended destination. Clearly that is not a valid forwarding state

(iv) The Round-Trip Time (RTT) of traffic forwarded over a shortest path computed by Dijkstra is higher than the RTT of traffic which takes a different (non-shortest) path towards the same destination. How is that possible? (1 Point)

Solution: "Shortest" does not mean the fastest path, it rather depends on how the operator set the link weights. For example, the "shortest" path could have the highest bandwidth but might be longer (physically) than another path.

Another explanation would be that the "shortest" path is currently congested and therefore slower than the other path.

(v) Explain one possible problem if you apply the Dijkstra algorithm to a network which contains links with negative weights. (1 Point)

Solution: The Dijkstra algorithm could end up in a state where it continuously loops over links with negative weights in order to find an even shorter path.

(vi) Explain why routers normally load balance traffic between two shortest paths in such a way that all packets belonging to one flow follow the same path. (1 Point)

Solution: Distributing packets over both paths could lead to unnecessary packet reordering at the receiver (and therefore potential packet retransmissions) as one path might have a shorter delay than the other one.

b) Reverse Distance Vector **(10 Points)**

leave blank

Table 9 shows the evolution of shortest paths from node A towards all other nodes in the network (Figure 3) while running a distance vector algorithm. It takes one time step (i.e., one row in the table) for a distance vector message to be sent from one node to another on a link. A node can send distance vector messages to all direct neighbors at the same time.

step	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**
0	0	∞	∞	∞	∞	∞	∞	∞	∞
1	0	20	3	∞	∞	∞	∞	∞	∞
2	0	**18**	3	25	26	13	∞	∞	∞
3	0	18	3	**23**	**24**	13	27	16	∞
4	0	18	3	23	24	13	**24**	16	36
5	0	18	3	23	24	13	24	16	36

Table 9: Node A's evolution of shortest paths towards all other nodes while running a distance vector algorithm. ∞ indicates that A does not yet know a shortest path.

(i) Given the information in Table 9, use Figure 3 and fill in all identified links with their corresponding link weights. If the given information could lead to multiple solutions how the nodes are connected, only indicate one of them. (7 Points)

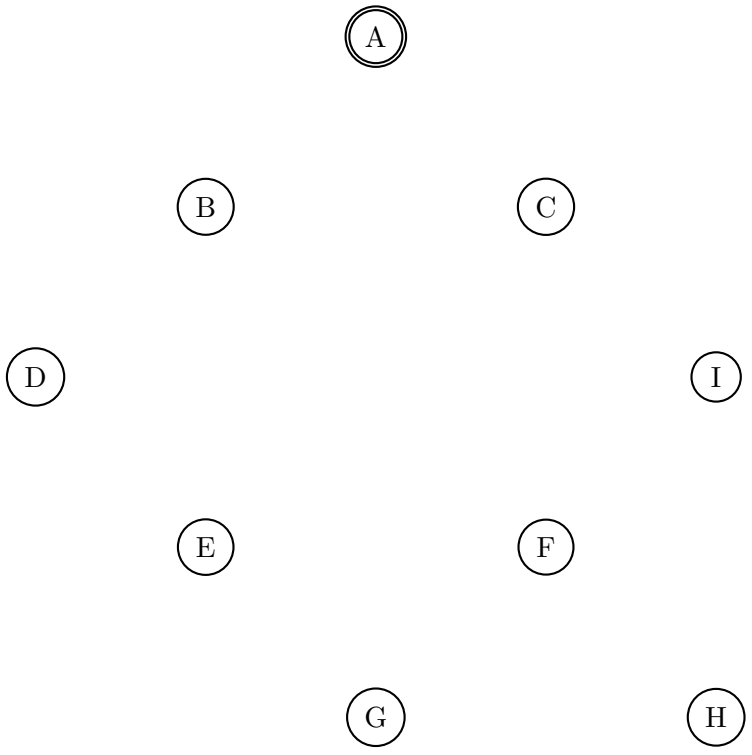

Figure 3: Use this figure to fill in all the detected links together with the corresponding weights according to the output of the distance vector algorithm in Table 9

Solution: Figure 4 and Figure 5 show two of many possible solutions. In order to construct the graph, it is helpful to go through the steps in order. It is important that a node is not connected "too early". For example, if node I would be connected to node F instead of H (as in the given solutions), its distance vector would already reach node A in step 3. That would contradict the information in Table 9. Similarly, for example

node I cannot be connected to node G instead, as we would otherwise see the shortest path reduction by 3 (step 3 to step 4 in G's column) also in I's column (from step 4 to step 5).

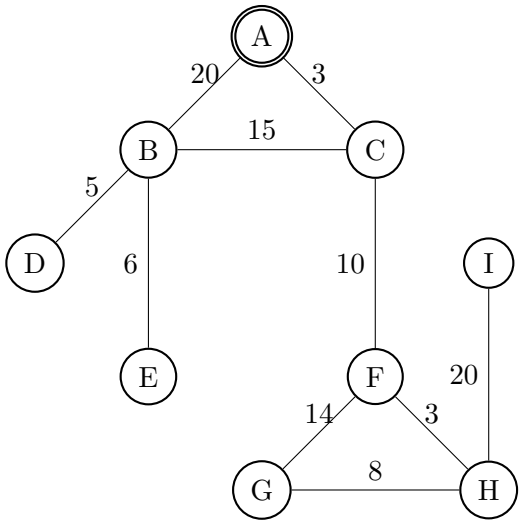

Figure 4: First possible solution

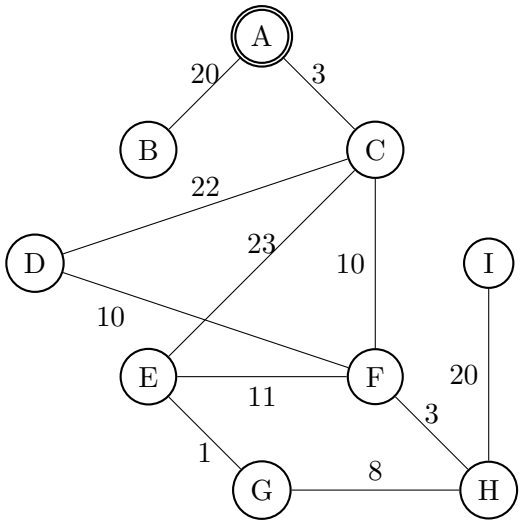

Figure 5: Second possible solution

(ii) As the previous question mentions, there might be multiple possible solutions given the information in Table 9. Assume now that the nodes instead run the Dijkstra algorithm to find all shortest paths and you receive the corresponding output for node A. Could the Dijkstra output of node A still lead to multiple solutions or would you only end up with one possible link and link weight assignment? Explain your answer.
Note: you do *not* have to figure out how the Dijkstra output or the link assignment would look like for the given network. (3 Points)

Solution: One can argue in two different ways.

Argument 1: For a Dijkstra algorithm with well-defined tie-breaking mechanism, we would always end up with the same, unique link and link weight assignment. That has two reasons: (i) we always know which node is currently evaluated (in a given time step). Therefore, it is also clear how the different nodes are connected with each other. And (ii) once Dijkstra finds a shortest path for a given node, it will never change later on.

Argument 2: If different Dijkstra implementations use different tie-breaking mechanisms, we could still end up with multiple solutions. For example, if two (not yet covered nodes) have the same shortest path, we could continue with either one, resulting in different solutions.

Note that in both cases (Dijkstra and Distance Vector), the network might contain additional links which we do not discover/see with the information given in the table. However, the question did not ask for that.

leave blank

c) Label-based Forwarding **(9 Points)**

This question compares destination-based forwarding introduced in the lecture with so-called "label-based" forwarding.

Figure 6 shows a simple example. The network contains four routers (A - D) and three hosts (H_1 - H_3) which are the destinations we want to reach.

The four tables at the **top** of the figure represent normal destination-based forwarding. Each router contains one forwarding entry for each destination ("dst" column) and knows how to forward packets such that they eventually reach the destination ("out" column).

The four tables at the **bottom** show label-based forwarding. The two edge routers (A and D), which are connected to the hosts, have a full forwarding table. For each packet, they add labels ("added label" column) to an extra header field. The two core routers (B and C) only forward based on these labels ("label" column).

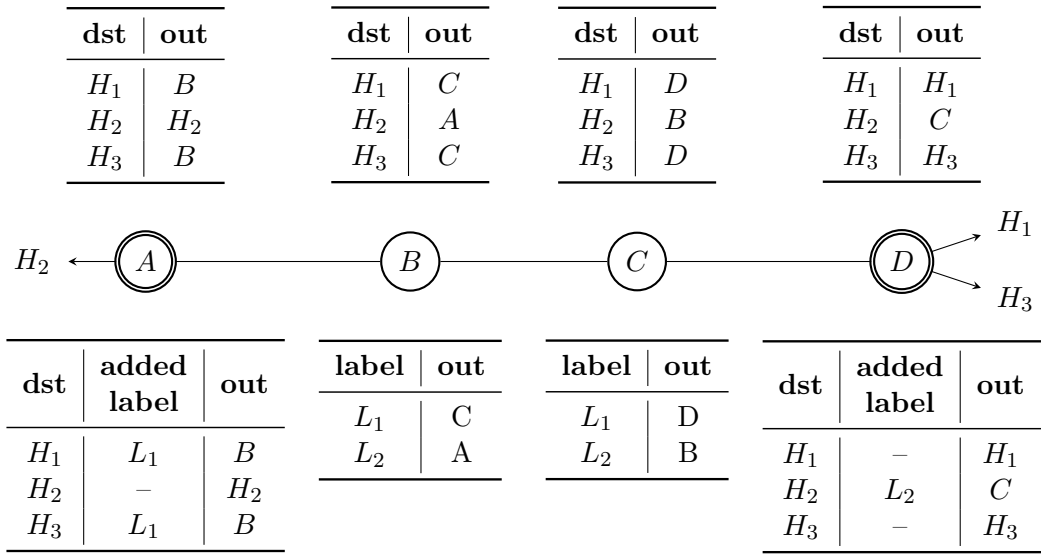

Figure 6: Dst-based (top) and label-based (bottom) forwarding.

(i) Find *two* advantages and *one* disadvantage of label-based forwarding compared to the destination-based forwarding we saw in the lecture. (3 Points)

Solution: Some **advantages** (two are enough): The core routers have a greatly reduced forwarding table size. More flexible routing options, e.g., source-based routing (see also answer to (iii)). Changes in the forwarding behavior can be much faster. For example, it might be enough that a single edge router changes the label for one table entry.

Some **disadvantages** (one is enough): The forwarding table of the edge routers is bigger (new label column). Also all packets might have larger headers (depending on where the label is placed). Finally, automatic forwarding re-computations in case of a failure might be more difficult for core routers, after all they cannot just fall back to IP-based computations (they do not know the full table).

(ii) We want to forward traffic for D destinations in an arbitrary network with N routers out of which M are edge routers such as router A in Figure 6. How many different labels do you need *at most* in order to achieve any forwarding state which is possible with the destination-based forwarding we saw in the lecture? Explain your answer. (2 Points)

Solution: As we have seen in the lecture, for destination-based routing, two paths towards the same destination will never separate again once they merge on a node/router. That means, we need at most M labels, one for each edge router.

One can also argue that in a real network, some of the core routers might also be connected to hosts (and therefore need to be reachable). In this case, we might need up to N labels.

Finally, another possible solution would be to say we need $\min(M, D)$ labels should not all the edge routers have at least one host.

(iii) Figure 8 contains a network with three edge routers (A, B, F) which are connected to one host each (H_1-H_3). Can you find labels which result in a forwarding behavior which is *not* achievable with destination-based forwarding? If you think such a forwarding behavior is possible, show one example by filling in corresponding entries in the "added label"/"label" (L_1, L_2, ...) and "out" (A, B, ...) columns. Make sure that packets entering an edge router are correctly forwarded to the three hosts.
If you think it is impossible, explain your reasons below. (4 Points)

Solution: It is indeed possible to find a forwarding behavior which is *not* possible with destination-based forwarding. In the lecture we saw the idea of source-based routing which is applicable in the label-based setting. Figure 7 shows one corresponding label-distribution. Router C forwards the packets towards host H_1 over both paths, depending on the label/source they are coming from.

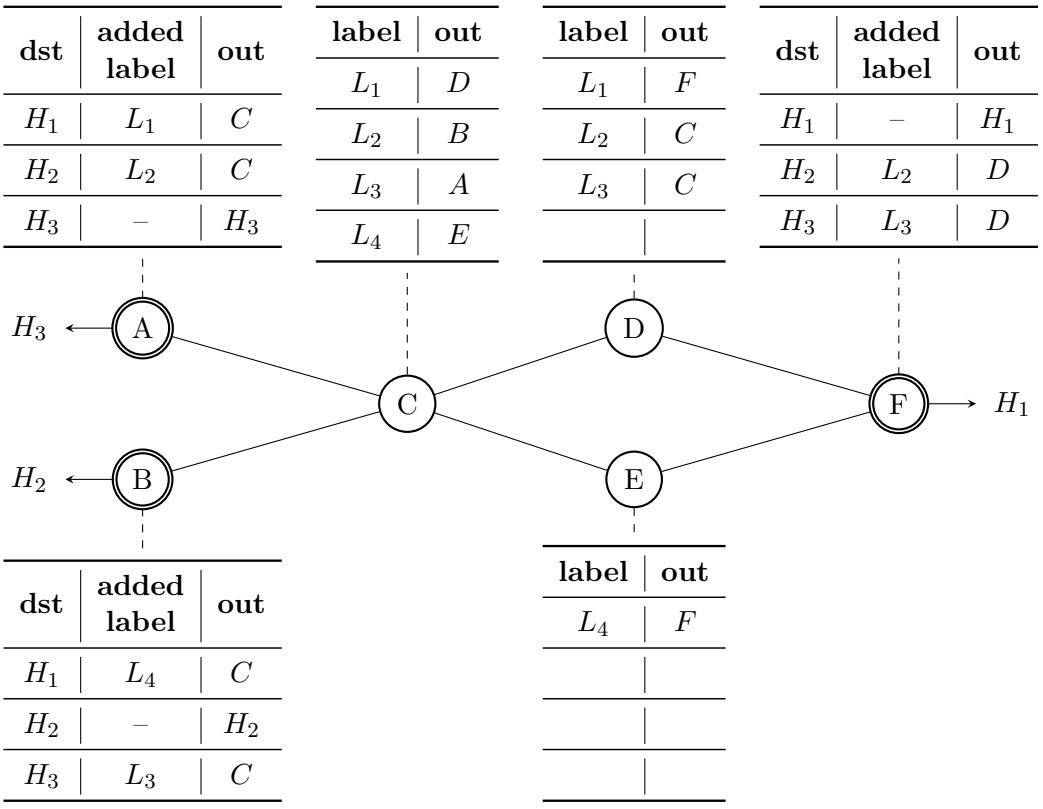

Figure 7: Show a forwarding behavior which is *not* possible with destination-based forwarding.

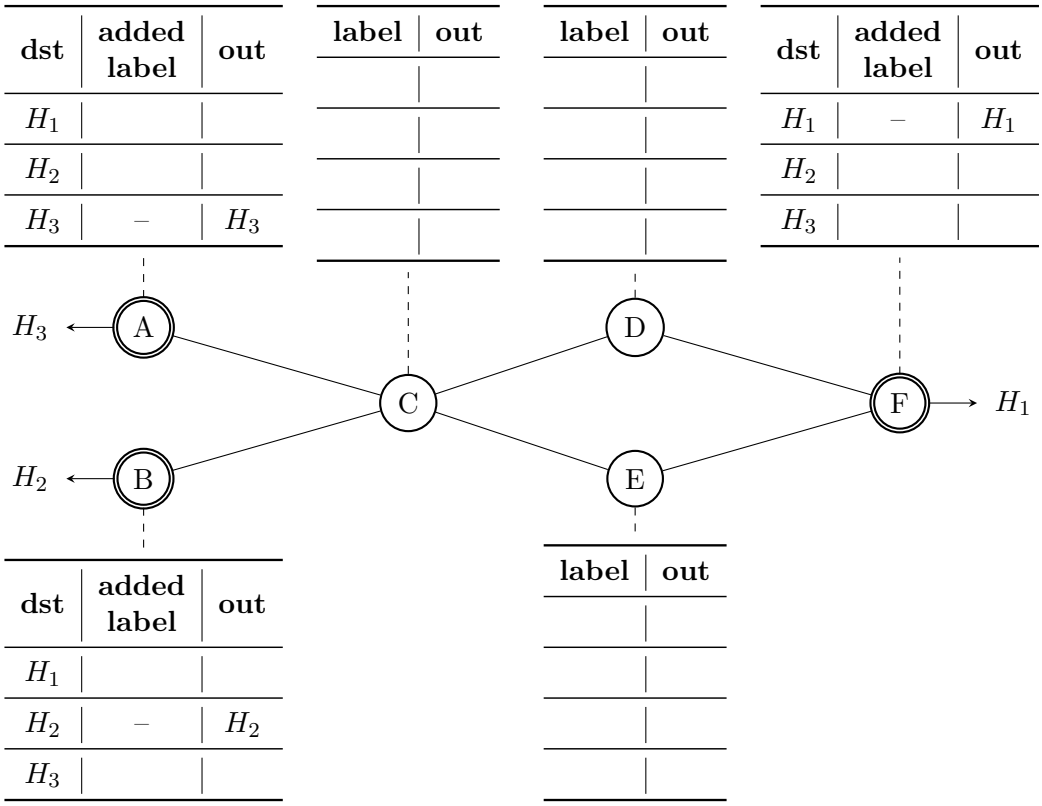

Figure 8: Show a forwarding behavior which is *not* possible with destination-based forwarding.

leave blank

Task 3: Inter-domain routing **39 Points**

leave blank

a) Warm-Up (8 Points)

You are an operator of an AS and you have decided to peer with two new networks, namely AS 195 and AS 466, at router `ZURI`. Before establishing the eBGP sessions, you receive the BGP routes that both of them would advertise to you. `ZURI` performs regular BGP path selection as presented in the lecture, breaking ties according to the **smaller next hop IP address**. Further, assume that `ZURI` does not know any additional routing information.

For each subtask, you are given a table with three routes, one from each AS (195 and 466), and the currently known route. Clearly indicate the selected route for the given destination **directly in the table**.

(i) Which route will be picked for the destination `5.21.2.35`? (1 Point)

	from	prefix	next hop	local pref.	MED	AS path	IGP Cost
☐	Current	`5.21.2.0/24`	`12.16.9.1`	50	200	260 590	4600
☑	AS 195	`5.21.2.0/24`	`20.79.3.2`	100	50	195 439 590	0
☐	AS 466	`5.21.2.0/24`	`13.8.47.25`	100	120	466 338 10 590	0

Solution: From AS 195. *Reason*: Consider the BGP decision rules:

- *Prefix length* (longer is better): All have the same prefix length.
- *Local preference* (higher is better): Routes Ignore local route with local pref 50.
- *AS Path length* (shorter is better): Route from AS 195 has a shorter AS path than the one from AS 466.

(ii) Which route will be picked for the destination `15.8.2.250`? (1 Point)

	from	prefix	next hop	local pref.	MED	AS path	IGP Cost
☑	Current	`15.8.2.0/24`	`12.16.9.2`	100	130	195 13 13 290	1250
☐	AS 195	`15.8.2.0/24`	`20.79.3.2`	100	140	195 13 13 290	0
☐	AS 466	`15.8.2.0/24`	`13.8.47.25`	100	80	466 20 6 13 290	0

Solution: Current route. *Reason*: Consider the BGP decision rules:

- *Prefix length* (longer is better): All have the same prefix length.
- *Local preference* (higher is better): All have the same local pref.
- *AS Path length* (shorter is better): Route from AS 466 has the longest path.
- *MED* (lower is better): MED is compared because the most recent AS in both paths are equal. The MED of the current route is lower.

(iii) Which route will be picked for the destination `68.7.5.6`? (1 Point)

	from	prefix	next hop	local pref.	MED	AS path	IGP Cost
☐	Current	`68.7.5.0/24`	`12.16.9.1`	100	120	42 91 20 590	4600
☐	AS 195	`68.7.5.0/24`	`20.79.3.2`	100	150	195 13 20 590	0
☑	AS 466	`68.7.5.0/24`	`13.8.47.25`	100	180	466 92 20 590	0

Solution: AS 466. *Reason*: Consider the BGP decision rules:

- *Prefix length* (longer is better): All have the same prefix length.
- *Local preference* (higher is better): All have the same local pref.
- *AS Path length* (shorter is better): All have equal AS path lengths.
- *MED* is **not compared** because the AS paths do not end in the same number.
- *IGP Cost* (lower is better): The current route has the highest IGP cost, both others have a cost of zero.
- *Next hop* (lower is better): The route from AS 466 has the lower next-hop IP address.

(iv) Which route will be picked for the destination `2.7.8.22`? (1 Point)

	from	prefix	next hop	local pref.	MED	AS path	IGP Cost
☐	Current	`2.7.8.0/23`	`56.22.219.29`	150	50	30 89 59 20	0
☑	AS 195	`2.7.8.0/24`	`20.79.3.2`	100	100	195 338 89 59 20	0
☐	AS 466	`2.7.8.0/23`	`13.8.47.25`	100	80	466 439 20	0

Solution: AS 195. *Reason*: The route from AS 195 has the longest prefix.

Consider the following table. There exists no single, most preferred route for `9.19.2.0/20`.

from	prefix	next hop	local pref.	MED	AS path	IGP Cost
Current	`9.19.2.0/20`	`12.16.9.1`	100	130	466 20 120	4600
AS 195	`9.19.2.0/20`	`20.79.3.2`	100	120	195 338 120	0
AS 466	`9.19.2.0/20`	`13.8.47.25`	100	150	466 20 120	0

(v) Compare the three routes for `9.19.2.0/20` individually with each other. Assume that only two routes are present at the same time. For each pair, indicate which route is preferred, and write down the deciding attribute. (2 Points)

Solution: All routes have equal prefix length, local pref, and AS path length.

1. *Current vs. AS 195*: The route from **AS 195** is preferred! MED is not compared because the most recent number in the AS path is not equal. The current route has a higher IGP cost, which means the route from AS 195 is preferred.
2. **Current vs. AS 466**: The **current** route is preferred because it has a lower MED attribute than the route from AS 466. The MED attribute is compared because the most recent number in the AS path is equal.
3. **AS 195 vs AS 466**: The route from **AS 466** is preferred! MED is not compared, and the IGP costs are equal. The route from AS 466 has a lower next-hop IP address.

(iv) Why is there no single, most preferred route for `9.19.2.0/24`, and what can you do as a network operator to resolve this issue? (2 Points)

Solution: There is no absolute order of the three routes (cyclic preference). The route preference is undefined. This issue can be solved either by denying a route, ignoring the MED attribute, or changing the local preference or AS path of any of the three routes.

b) Traffic Engineering and Load Balancing **(8 Points)**

leave blank

You are a network operator of AS 1, as depicted in Figure 9.

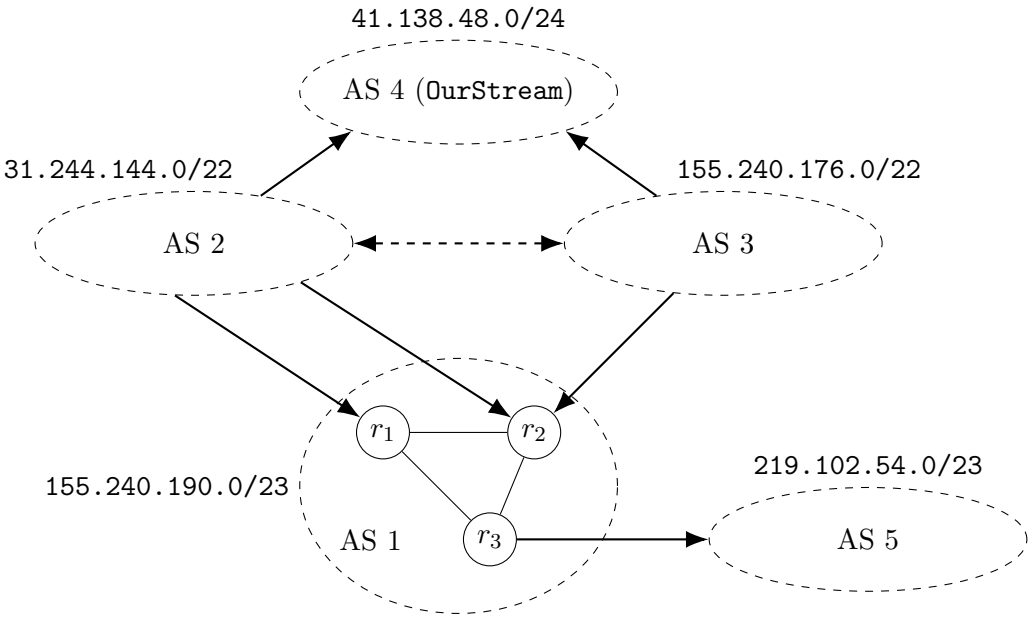

Figure 9: Network Topology

Single-headed plain arrows point from providers to their customers (AS 2 is the provider of AS 1), while double-headed dashed arrows connect peers (AS 2 and AS 3 are peers). Each AS applies the default selection and exportation BGP policies based on their customers, peers and providers.

(i) Consider the current topology. Which packets are routed over the direct peering session between AS 2 and AS 3? Give the prefixes of those packets. (2 Points)

Solution: `31.244.144.0/22` and `155.240.176.0/22`.

Explain why those prefixes will be routed directly, and why all others will not.

Solution: Since every AS prefers routes from their customers over those from their peers, and since both AS 2 and AS 3 have the same clients, only the prefixes originating in AS 2 and AS 3 will be routed over the peering link. Further, since an AS will never advertise the prefixes of a provider to another provider, AS 2 cannot reach AS 3 via AS 1 or AS 4 (or vice-versa).

(ii) Will router r_3 of AS 1 perform inter-domain load balancing for traffic towards destination `41.138.48.0/22` (AS 4)? What about r_2 of AS 1? Justify your answer. (2 Points)

Solution: Both r_3 and r_2 will not load-balance traffic, but either forward traffic to AS 2 or AS 3. BGP only selects a single best path, and this one is used to route traffic.

In the following, you will perform inter-domain load balancing for traffic from the very popular streaming platform `OurStream` hosted in AS 4 to your customer AS 5. You notice that traffic from `OurStream` **traverses AS 3** since AS 4 sets a *higher local preference* to routes received from AS 3 than AS 2. Consequently, the link from AS 3 to r_2 is congested.

(iii) You want to load balance the traffic from `OurStream` to AS 5 between AS 2 and AS 3. How do you change the routing announcements to load balance traffic regardless of the local preference configured by AS 4? Your solution should still provide reachability, even if any link from AS 1 to either AS 2 or AS 3 fails. (2 Points)

Solution: Announce `219.102.54.0/23` + `219.102.54.0/24` to AS 2, and `219.102.54.0/23` + `219.102.55.0/24` to AS 3.

(iv) Your load balancing works, and you now see traffic coming from both AS 2 and AS 3. However, AS 2 chooses to send the traffic towards r_2, and now, the link between r_2 and r_3 is congested. You now want to make AS 2 to send you traffic for AS 5 towards r_1. Mention two techniques (different from task **(iii)**) for influencing AS 2 to forward traffic with destination `219.120.54.0/23` towards r_1 instead of r_2. Both should not reduce the robustness against link failures. (2 Points)

Solution: Either append your own AS multiple times or set a **lower** MED value for the announcements from r_1 towards AS 2.

leave blank

c) BGP Hijack **(11 Points)**

Consider the small Internet topology depicted in Figure 10. A user, located in AS 4, communicates with a server in AS 5. You are owning and maintaining AS 6. AS 4 is advertising the prefix `244.41.174.0/23` towards AS 2, and AS 5 announces `78.35.58.0/23` to AS 3. We assume that no AS validates any BGP announcement (e.g., using RPKI).

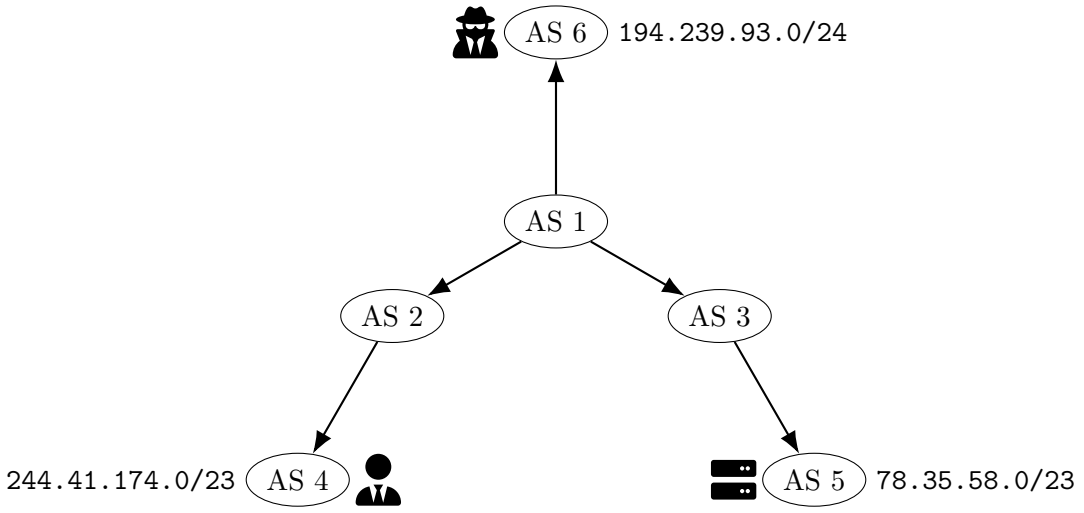

Figure 10: Network Topology

Single-headed plain arrows point from providers to their customers (AS 1 is the provider of AS 2). Unless stated otherwise, each AS applies the default selection and exportation BGP policies based on their customers, peers and providers.

In the following questions we consider different attacks that you can perform from AS 6. Your first goal is to **impersonate the server** (AS 5). You wish to attract traffic from AS 4 towards AS 5 and answer back to AS 4, without AS 5 ever receiving any traffic from AS 4.

(i) What BGP route do you advertise to AS 1, such that you can impersonate the server? We assume that no AS is manipulating the prefixes they receive. However, your attack should work no matter how AS 1, AS 2, and AS 3 are configured. (2 Points)

Solution: Advertise `78.35.58.0/24` and `78.35.59.0/24` to AS 1.

(ii) Could AS 5 detect the BGP hijack? Justify your answer. (1 Point)

Solution:

- Yes. They would receive an advertisement for their prefix as it will simply propagate to AS 5.
- No if we use path poisoning, such that AS 3 rejects the advertisement. In that case, the hijack should include AS 3 in the AS path.

You now wish to **eavesdrop** on the traffic **from the user to the server** (only in this direction). This means that you must attract traffic from AS 4 and forward it to AS 5.

(iii) What is the problem with the current topology? Why is it impossible to eavesdrop on traffic from the user to the server as an attacker located at AS 6? (2 Points)

Solution: To eavesdrop, you need to first attract traffic for the server to you, and then again forward traffic towards the destination. In the current setup, this is impossible as we only have a single session (i.e., link) from AS 6 to the rest of the internet.

(iv) To make it possible to eavesdrop on traffic from the user to the server, you decide to add a single new link (with an eBGP session) either to AS 2 or to AS 3. (6 Points)

With which AS would you establish a session, and which business relationship do you choose (i.e., are you a customer, a provider, or a peer)?

Explain which destinations you would advertise to AS 1 and over the new session, such that you can eavesdrop on traffic from the user to the server.

Does your attack work even if AS 1, AS 2, or AS 3 prepend their AS number multiple times to the AS path? Explain why or why not.

Explain how AS 6 handles incoming packets to perform the eavesdrop attack.

Solution: There are several different, valid solutions (others might be possible):

- *Peer with AS 2, or be a provider of AS 2.* In this case, we simply advertise the more specific prefixes to AS 2 and advertise nothing to AS 1 (at least no prefix from AS 5). Then, AS 2 will choose you as a destination, but it will not advertise the more specific route to AS 1. Hence, you can still use AS 1 to reach AS 5. This technique works no matter if special rules are implemented.
- *Be a customer of AS 2.* In this case, we cannot advertise a more specific prefix to AS 2, as it would also forward the route to AS 1. We have to advertise `78.35.58.0/23` towards AS 2, but with an AS path length of exactly 2. Then, AS 2 will prefer this route over the one learned from AS 1, while AS 1 will prefer the one from AS 3 due to a shorter AS path. This technique only works if AS 1 does not prepend its AS number multiple times to the path.
- *Peer with AS 3 (works with any relationship)* In this case, we cannot advertise a more specific prefix to AS 1, as AS 3 would also prefer that one. Instead, we have to advertise `78.35.58.0/23` towards AS 1, but with an AS path length of exactly 1. Then, AS 1 will prefer this route over the one learned from AS 3, while AS 3 will prefer the route from its customer. This technique works in any case unless AS 1 prefers routes from AS 3 over AS 6, no matter the AS path length.
- *Solution with AS path poisoning* Peer with any of AS 2 or AS 3 with any business relations At the *left-most* AS connected to AS 6 (either AS 2 or AS 1), we advertise the prefix with the `right-most` AS connected to AS 6 (either AS 1 or AS 3) in the path. This way, the hijack will not reach any AS between AS 6 and AS 5. This technique will work in any case.

d) Inferring AS Relationships **(12 Points)**

leave blank

Consider the following Internet with 11 different ASes in Figure 11. AS 15, AS 17, and AS 18 are Tier-1 ISPs without any provider, whereas all other ASes have at least one provider. Some BGP sessions between ASes are already shown in the figure, but not all of them.

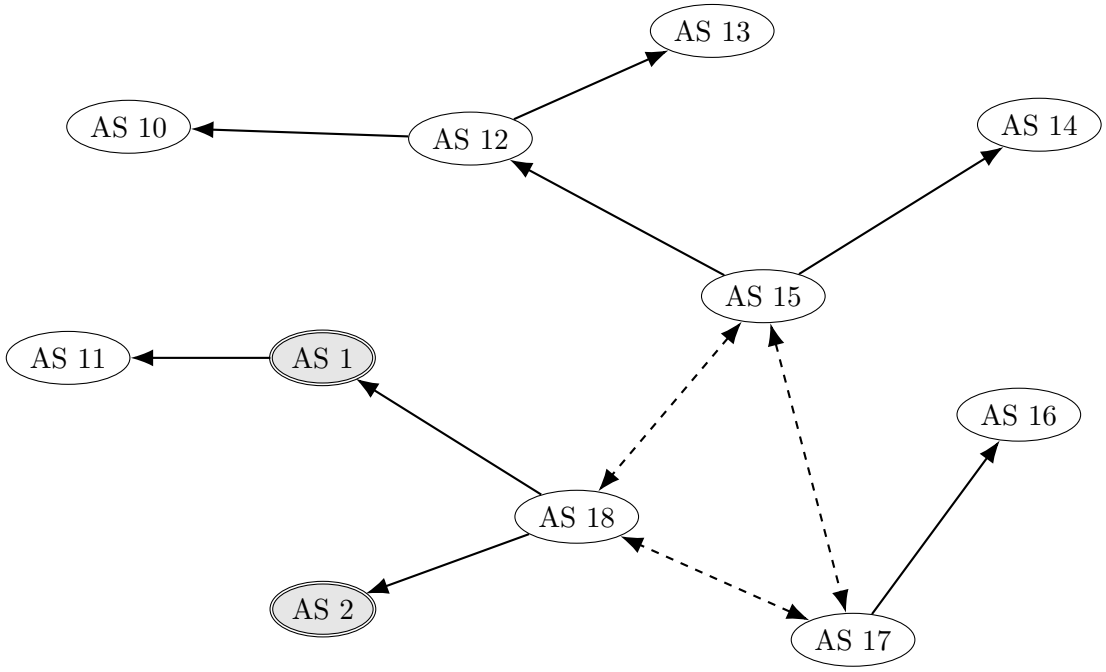

Figure 11: Network topology with a **subset** of existing BGP sessions.

Single-headed plain arrows point from providers to their customers, while double-headed dashed arrows connect peers. Each AS applies the default selection and exportation BGP policies based on their customers, peers and providers. ASes break ties by preferring the neighbor with the **lowest AS number**.

Each AS `XX` advertises its prefix `XX.0.0.0/24` to all of its neighbors. For instance, AS 12 advertises `12.0.0.0/24`. The Tier-1 ISPs (AS 15, 17, and 18) do not advertise any prefix. You are given a complete and sorted list of **all** incoming BGP messages of AS 1 and AS 2.

AS 1			
#	kind	prefix	AS path
1	U	`2.0.0.0/24`	1 18 2
2	U	`10.0.0.0/24`	1 11 10
3	U	`10.0.0.0/24`	1 12 10
4	W	`10.0.0.0/24`	1 11 10
5	U	`11.0.0.0/24`	1 11
6	U	`12.0.0.0/24`	1 12
7	U	`13.0.0.0/24`	1 12 13
8	U	`13.0.0.0/24`	1 18 15 12 13
9	U	`14.0.0.0/24`	1 18 15 14
10	U	`16.0.0.0/24`	1 18 17 16

AS 2			
#	kind	prefix	AS path
11	U	`1.0.0.0/24`	2 18 1
12	U	`10.0.0.0/24`	2 18 1 11 10
13	U	`10.0.0.0/24`	2 18 15 12 10
14	U	`11.0.0.0/24`	2 18 1 11
15	U	`12.0.0.0/24`	2 18 15 12
16	U	`13.0.0.0/24`	2 18 15 12 13
17	U	`14.0.0.0/24`	2 18 15 14
18	U	`16.0.0.0/24`	2 18 17 16

Figure 12: Stream of BGP messages received by AS 1 and AS 2. "U" abbreviates a BGP Update message, and "W" abbreviates a BGP Withdraw message.

(i) Consider only the BGP messages of AS 1 (the first table of Figure 12). What might have caused message number 4? (1 Point)

Solution: AS 11 and AS 12 may have terminated their business relationship or changed it from a customer (AS 10) to provider (AS 11) to a peering relationship (or a provider to customer). Another possibility is a link failure between AS 10 and AS 11, or some (mis-) configuration in AS 10 or AS 11.

Recall that only some BGP sessions are given, but not all of them. As an example, consider message 6 from Figure 12. AS 12 advertises its own prefix directly towards AS 1 with AS path $[1, 12]$. This indicates that there must exist a BGP session between AS 1 and AS 12.

(ii) Which kind of BGP session is possible between **AS 15** and **AS 16**? For each kind of business relationship, justify your answer. (3 Points)

Solution:

1. **YES** *Peer to Peer* Is possible, as path 18 15 16 is invalid. Therefore, we could never see the advertisement.
2. **NO** *Customer (AS 15) to Provider (AS 16)* is not possible because AS 15 is a Tier-1 ISP, but AS 16 is not.
3. **NO** *Provider (AS 15) to Customer (AS 16)* is not possible, as otherwise, AS 18 would prefer the path 18 15 16 over 18 17 16.

(iii) Which kind of BGP session is possible between **AS 13** and **AS 14**? For each kind of business relationship, justify your answer. (4 Points)

Solution:

1. **YES** *Peer to Peer* is possible, as neither AS 13 nor AS 14 would export routes learned from peers to providers.
2. **YES** *Customer (AS 13) to Provider (AS 14)* is possible, as AS 15 prefers the path 15 12 13 over 15 14 13.
3. **NO** *Provider (AS 13) to Customer (AS 14)* is not possible, as path 1 12 13 14 would appear in the BGP messages of AS 1, no matter which kind of session exists between AS 1 and AS 12.

(iv) Which kind of BGP session is possible between **AS 1** and **AS 12**? For each kind of business relationship, justify your answer. (4 Points)

Solution:

- **YES** *Peer to Peer* is the only solution, as we observe the path 1 12 10 and 1 12 13, but not 1 12 15 *.
- **NO** *Customer (AS 1) to Provider (AS 12)* is not possible, because we do not observe the path 1 12 15 *.
- **NO** *Provider (AS 1) to Customer (AS 12)* is not possible, because we do not observe the path 2 18 1 12 13, which should be preferred over 2 18 15 12 13, as AS 18 prefers customers over peers.

leave blank

Task 4: Reliable transport **36 Points**

leave blank

a) Warm-Up **(7 Points)**

(i) Explain what *the congestion collapse* refers to and why it almost happened. (1 Point)

Solution: Before we had a congestion control algorithm, the Internet almost "died" of congestion. At this point in time, the sending rate was only limited by flow control and clients always sent a full window of packets. Given a limited link bandwidth and increasing number of Internet users, the average throughput decreased from 32 Kbps to roughly 40 bps during the *congestion collapse*.

(ii) Why is the TCP sequence number no longer based on a timestamp? (1 Point)

Solution: A timestamp-based sequence number makes it feasible for an attacker to perform a targeted denial-of-service attack. The attacker could guess what the sequence number of new flows would be. The attacker then injects a lot of garbage packets which have the expected sequence number(s) with the hope that a TCP client would accept some of them. That could lead to a crash of the underlying application.

(iii) Explain with an example how cumulative ACKs can cause unnecessary retransmission. (1 Point)

Solution: Assume the sender transmits data packets 1, 2, 3, 4 which get reordered into 4, 3, 2, 1 before reaching the receiver. The receiver sends back ACK 1 three times which would lead to a retransmission of data packet 1, even though the original packet 1 was not lost.

(iv) Compute the max-min fair bandwidth allocation for the flows A, B, C, D in Figure 13. The bandwidth on each edge is shared for both directions. (4 Points)

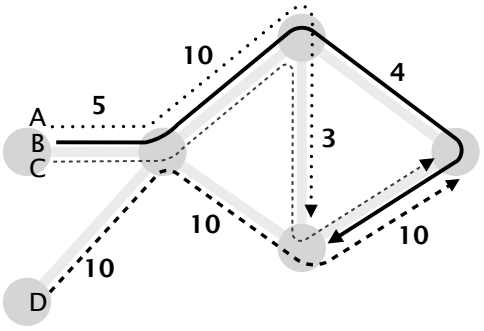

Figure 13: Graph with four concurrent flows. The edges are labeled with their bandwidth.

Solution: Flow A and C first saturate the vertical edge in the middle and get 1.5 each. Next, flow B saturates the starting edge of flows A, B, C and gets 2 (as 2×1.5 is already used by A and C). Finally, D saturates the last edge on its path and gets 6.5 ($10 - 1.5 - 2$).

leave blank

b) GBN protocol (13 Points)

In this task, you complete partial time-sequence diagrams of the Go-Back-N (GBN) protocol. *Attention: Each subtask presents a new situation, they do not relate to each other.*

Here is a **non-exhaustive** list of implementation choices made for the GBN sender and receiver. **Read them carefully.**

- The sender and receiver window have a size of 4 packets;
- The receiver's out-of-order buffer stays empty in this task and is therefore not shown;
- 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. For instance, if the sender gets ACKs [A9, A10, A10, A10], it will immediately retransmit the data segment D10;
- For each tick in the diagrams 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 (timeout) of 6 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 needs *two* ticks, and an ACK *one* tick to travel to the other end of the connection. See the given start in the diagrams.

Important notes for filling in the solutions:

- For all provided boxes and brackets, mark it clearly if they are empty (e.g. with a dash "—"). This can happen frequently.
- Indicate timeouts with an arrow. Draw the arrow start where the timeout counter starts, and draw the arrow end where the timeout stops (if it fits into the diagram).
- You do not need to add ticks to the time arrows or draw arrows that point into empty space (except for timeouts where the end does not fit into the diagram).

(i) Fill in the given boxes and brackets in Figure 14 and continue the arrows. Here, the sender intends to send D1 – D5. It has already sent D1 – D3 once, and just sent D4 for the first time (depicted). (2 Points)

Solution:

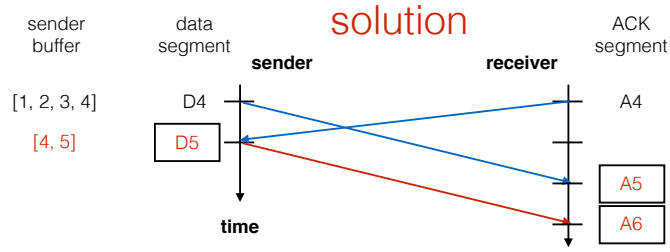

Figure 14: Solution for (i)

(ii) Fill in the given boxes and brackets in Figure 15 and continue the arrows. Here, the sender already sent D1 – D4. Currently, it fast retransmits D1 (depicted), but it is lost (marked with X). (4 Points)

Solution:

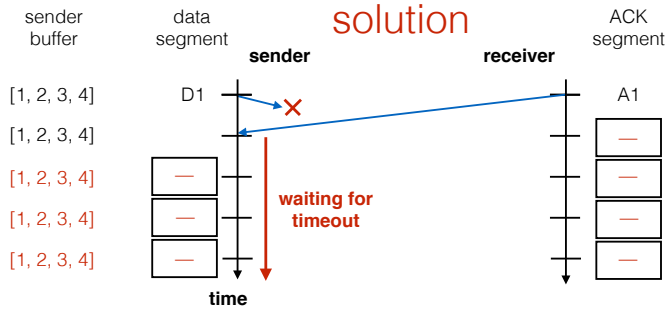

Figure 15: Solution for (ii)

The sender starts the retransmission timer immediately after receiving A1 (last time it sent a data segment or received an ACK).

(iii) Only fill in the ACKs in Figure 16. (All arrows are already given.) (3 Points)

Solution:

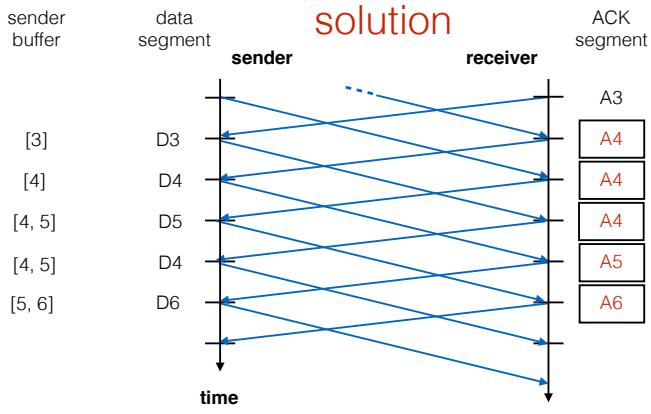

Figure 16: Solution for (iii)

Note that the first three ACKs need to be A4, otherwise the sender would not retransmit D4 in the second to last step.

(iv) Assume that we additionally have a congestion window. It does not change with the regular TCP rules, however. Instead, it is simply given that it starts with size 2 and shifts to size 1 at the dotted line in Figure 17. Fill in the given boxes and brackets in Figure 17 and continue the arrows. The sender is in the middle of sending D1 – D20.

(4 Points)

Solution:

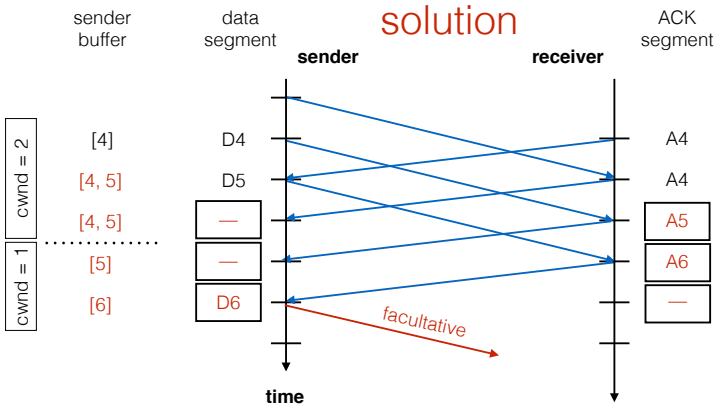

Figure 17: Solution for (iv)

The sender cannot send D6 in the third step even though the sender buffer/window is not yet full. The congestion window prevents it.

leave blank

c) Congestion Window Evolution (12 Points)

Figure 18 depicts an evolution of the TCP congestion window. In this task, you will complete the further evolution of the congestion window and compute how the RTT estimate and timeout value evolve alongside.

Assume that:

- RTT = 1 s for all packets, except for the lost ones
- the entire window is sent *simultaneously*, meaning that the `cwnd` directly jumps from e.g. 2 (at $t = 2\,\text{s}$) to 4 kB (at $t = 3\,\text{s}$)
- all sent packets have a size of 1 kB
- RTT_{est} means the current RTT estimate, RTO means the current timeout value
- in this task, you should only consider the propagation delay, all other delay types are negligible
- the slow-start phase does not continue in the *next* iteration if in the *next* iteration, the congestion window is bigger than the slow-start threshold `ssthresh`.

solution

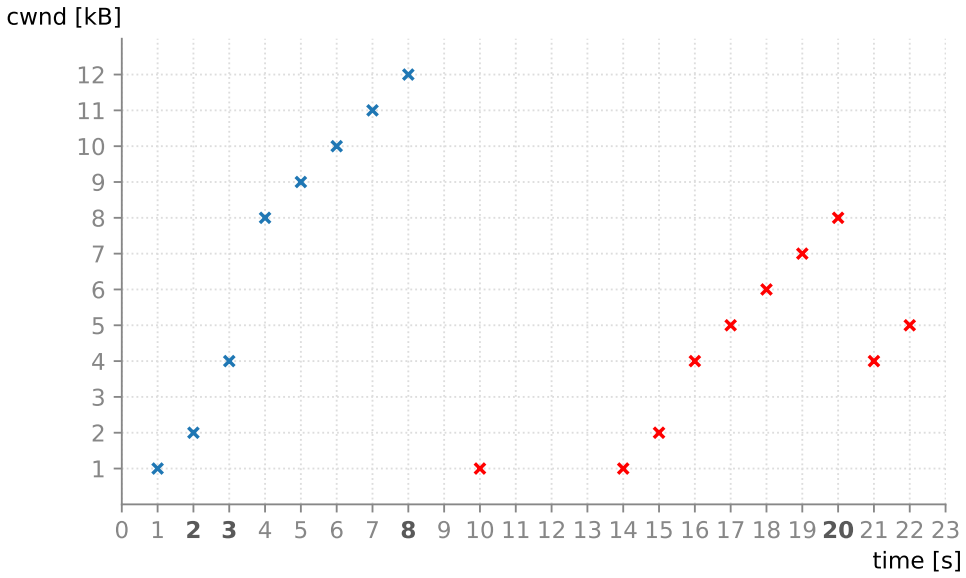

Figure 18: Congestion window evolution; y-axis: congestion window size, x-axis: time.

(i) Assume that at $t = 2\,\text{s}$ the RTT estimate is $\text{RTT}_{\text{est}} = 3\,\text{s}$. If $\alpha = 0.5$, what value does RTT_{est} have at $t = 3\,\text{s}$? Document your computation below. (3 Points)

Solution:

Apply the RTT estimation formula (known from the lecture) twice, since the current `cwnd` allows to send 2 packets. Use that both RTT samples are the same, namely 1 s (as introduced in the question). We end up with a RTT_{est} of 1.5 s. Computation without units:

$$
\begin{aligned}
\mathrm{RTT}^{\mathrm{B}}_{\mathrm{est}} &= \alpha \cdot (\alpha \cdot \mathrm{RTT}^{\mathrm{A}}_{\mathrm{est}} + (1-\alpha) \cdot \mathrm{RTT}_{\mathrm{sample}}) + (1-\alpha) \cdot \mathrm{RTT}_{\mathrm{sample}} \\
&\overset{\alpha=0.5}{=} \frac{1}{2} \cdot (\frac{1}{2} \cdot \mathrm{RTT}^{\mathrm{A}}_{\mathrm{est}} + \frac{1}{2}) + \frac{1}{2} \\
&= \frac{1}{2} \cdot (\frac{1}{2} \cdot 3 + \frac{1}{2}) + \frac{1}{2} \\
&= \frac{1}{4} \cdot 3 + \frac{1}{4} + \frac{1}{2} \\
&= 1.5
\end{aligned}
$$

(ii) The packets sent at $t = 8\,\mathrm{s}$ are all lost, resulting in a timeout. Assume that $\mathrm{RTT}_{\mathrm{est}} = 1\,\mathrm{s}$ at $t = 8\,\mathrm{s}$. If that is the case, how long will the timeout be? Draw (in Figure 18) the correct next `cwnd` at the point where the timeout expires. Document your computation below. (2 Points)

Solution: As we know from the lecture, the timeout is twice the current RTT estimate (which is 1 s). So we wait for 2 s. The `cwnd` is reset to 1 as shown in Figure 18.

(iii) After the first timeout expired, the timeout expires another time. Draw (in Figure 18) the next `cwnd` value at the point where the second timeout expires. Document your computation below. (1 Point)

Solution: We now experience the exponential backoff ($\mathrm{RTO}_{\mathrm{new}} = 2 \cdot \mathrm{RTO}_{\mathrm{old}}$). So we wait for 4 s. The `cwnd` is again (re)set to 1 as shown in Figure 18.

(iv) As soon as the second timeout expires, the sender sends the packets in the congestion window. From now, all packets are sent successfully. Draw (in Figure 18) the resulting `cwnd` values up to and including $t = 20\,\mathrm{s}$. *Assume that the second timeout does not affect the slow-start threshold `ssthresh`; also assume that there are no duplicate ACKs for this timespan.* Document your computation below. (3 Points)

Solution: Given that the second timeout does not influence the slow-start threshold, the current threshold is at 6 s. We perform slow start up to cwnd = 4. If we would continue with the slow start process, the `cwnd` in the next iteration would be bigger than the slow-start threshold (compare assumptions for this question). Therefore, we switch to additive increase as shown in Figure 18.

(v) At $t = 20\,\mathrm{s}$, the sender receives 3 duplicate ACKs. Explain why this can happen even though the data packets did arrive at the receiver. Furthermore, draw (in Figure 18) the resulting `cwnd` for the following two timesteps, assuming there are no more duplicate ACKs afterwards. (3 Points)

Solution: One reason could be that the data packets were reordered before arriving at the receiver (also compare Task 4 a) (iii)) which leads to the duplicated ACKs. Figure 18 shows the remaining `cwnd` steps.

leave blank

d) AIMD and Fairness **(4 Points)**

In this task, we call AIMD's coefficient for the additive increase $\alpha \in \mathbb{R}$, and the coefficient for the multiplicative decrease $\beta \in \mathbb{R}$. Concretely, we define AIMD as:

$$\texttt{cwnd}_{i+1} = \begin{cases} \texttt{cwnd}_i + \alpha & \text{no congestion signal} \\ \texttt{cwnd}_i \,/\, \beta & \text{congestion signal} \end{cases}$$

where $\texttt{cwnd}_{i+1}$ is the next iteration of $\texttt{cwnd}_i$, $i \in \mathbb{N}$. For this task, you can work without rounding, therefore $\texttt{cwnd}_i \in \mathbb{R} \;\; \forall i \in \mathbb{N}$.

Assume that two senders A and B run AIMD. Both use the same coefficients $\alpha, \beta \in \mathbb{R}$. What are the minimal conditions for α and β such that A and B converge to a fair bandwidth usage? If both oscillate around some values, a fair bandwidth usage means that they get the same amount of bandwidth *on average*.
Can it happen that A and B converge to a single point of bandwidth allocation (if so, under what circumstances?), or will they always "converge" to an oscillation? Explain.

Solution:
We have seen in the lecture that we need an actual additive increase and multiplicative decrease in order to reach a fair bandwidth usage. That gives us the following conditions. $\alpha > 0$ in order to achieve the additive increase and $\beta > 1$ for the multiplicative decrease.
A and B will *always* converge to an oscillation as we can see directly from the two conditions for α and β. α is strictly greater than 0, so the `cwnd` continuously increases (while more bandwidth is available). Similarly, β is strictly greater than 1, so the `cwnd` will always decrease by some amount once we reach the bottleneck bandwidth.

leave blank

Task 5: Applications **19 Points**

leave blank

a) Warm-Up **(4 Points)**

(i) Briefly explain why we need $n + 1$ RTTs to retrieve n small objects using persistent connections across multiple HTTP requests. (1 Point)

Solution: 1 RTT for the single TCP establishment the persistent connection requires; plus n RTTs for the subsequent HTTP request/responses (1 RTT for each object's HTTP request/response times n objects)

(ii) Briefly explain why web caching can reduce the delay in rendering a web page. (1 Point)

Solution: Rendering is faster as the client does not have to wait for the cached content to be re-transmitted by the Web server (possibly on slow/congested links). This is true as long as the cached content can be accessed either locally (the ideal case) or through a server which is closer than the actual Web server.

(iii) Your friend has hired a service to serve DNS records for her domain. She adds an A record, looks it up, and does not share it with anyone. A few days later, she modifies the A record but realizes that DNS lookups are still going to the old address, even when she tries on your computer. She suspects this is due to DNS caching but you have *never* visited her website before. Briefly explain her how that could be. (2 Points)

Solution: We are receiving the old IP address corresponding to the old A record because it is another DNS server, say S2 (and not the DNS server of the hired service, say S1) that has previously cached the old A record and now serves us the DNS response: At some point before the modification of the A record at S1, another computer has looked up the domain, which resulted in S2 retrieving from S1 the record and caching it until some expiration date which has not yet passed. So, this also means that the A record at S1 should not change until expiration date.

leave blank

b) Putting it All Together, Again **(7 Points)**

You just arrived at ETH Zürich. You boot your laptop, open your browser, and type in the following URL:

`https://isnt.routing.fun`

Assume that:

1. Your laptop has already received a public IP address from ETH's DHCP server.
2. Your laptop has already issued all the ARP requests needed. (Its ARP table is full).
3. Your laptop uses ETH's DNS resolver (which it also learned via DHCP).
4. No DNS entries are cached. (Everywhere).
5. The website is working correctly and fits entirely within a single segment/packet.
6. No packet is lost, ever.

Describe each packet sent by your host, the ETH's DNS resolver, and the webserver, in the order in which they are sent.

A short description of the packets suffices, meaning you do not need to write down all the packet headers. That said, for each packet exchange, make sure to specify: the source, the destination, the transport protocol used, together with a description of the content of the packet (DNS request for X, TCP SYN, HTTP GET, etc.).

Solution:

Packet	Source	Destination	Transport protocol	Content
1	laptop	ETH's DNS resolver	UDP	DNS request: Where is isnt.routing.fun?
2	ETH's DNS resolver	root DNS server	UDP	DNS request: Where is .fun?
3	root DNS server	ETH's DNS resolver	UDP	DNS reply: Here is the IP address of .fun DNS
4	ETH's DNS resolver	.fun DNS server	UDP	DNS request: Where is routing.fun?
5	.fun DNS server	ETH's DNS resolver	UDP	DNS reply: Here is the IP address of routing.fun
6	ETH's DNS resolver	routing.fun DNS server	UDP	DNS request: Where is isnt.routing.fun?
7	routing.fun DNS server	ETH's DNS resolver	UDP	DNS reply: Here is the IP address of isnt.routing.fun
8	ETH's DNS resolver	laptop	UDP	DNS reply: Here is the IP address of isnt.routing.fun
9	laptop	isnt.routing.fun server	TCP	TCP SYN
10	isnt.routing.fun server	laptop	TCP	TCP SYN ACK
11	laptop	isnt.routing.fun server	TCP	HTTPS GET request for isnt.routing.fun's base file
12	isnt.routing.fun server	laptop	TCP	HTTPS GET response 200 OK ... *base file*
13	laptop	isnt.routing.fun server	TCP	TCP FIN
14	isnt.routing.fun server	laptop	TCP	TCP FIN ACK
15	laptop	isnt.routing.fun server	TCP	TCP ACK

leave blank

c) Load, Balanced **(8 Points)**

The ETHZ networking team needs your help: the webserver hosting `www.ethz.ch` is frequently overloaded, leading to unhappy users.

They are currently considering acquiring three identical webservers for `www.ethz.ch` and would like to load balance the incoming user requests on them. The webservers would be hosted on `129.132.19.216`, `129.132.19.217`, and `129.132.19.218`.

In order to load-balance, the first technique they consider is a DNS-based round-robin, but they are not sure how to do it. This is where you come into action.

(i) Explain how DNS can be used to load-balance incoming requests. (1 Point)

Solution: By having the DNS server replying to subsequent DNS requests with different permutations of the three IP addresses that host www.ethz.ch. Since the clients receive

different permutations of the IP addresses, they attempt connections to different IP addresses (the ones appearing first in the sequence), thus equally balancing the load among the webservers.

(ii) What exact DNS resource records would you add to ETHZ's nameservers? Use the format (name, value, type, TTL). (2 Points)

Solution:
(www.ethz.ch, 129.132.19.216, A, 1)
(www.ethz.ch, 129.132.19.217, A, 1)
(www.ethz.ch, 129.132.19.218, A, 1)

(iii) Explain how the TTL affects the quality of the load-balancing. (1 Point)

Solution: Lower TTLs results in better load-balancing: When the TTL is low, clients/DNS servers don't cache the DNS mappings for too long and instead frequently ask the DNS server for www.ethz.ch's IP address. This enables the DNS server to frequently change the IP address clients receive, effectively quickly redistributing and reacting to changes in the overall load that could create imbalances among the webservers.

Another solution would involve relying on a load-balancing device. Here, the DNS record for `www.ethz.ch` would point to the IP address of the load balancer (`129.132.19.215`) which would then randomly load balance the incoming traffic onto the replicas by rewriting the destination IP of the incoming IP packets to the one of the chosen replica.

(iv) One issue is that they do not know how to guarantee that the load balancer ends up sending the IP packets belonging to the same TCP connection to the same replica. Describe a load-balancing technique the load balancer could use that does *not* require any state. (That is, your solution should not require the load balancer to remember where each TCP connection is going.) (3 Points)

Solution: We can use consistent hashing: for each packet, send it to the replica that results from hashing the immutable packet header fields (e.g., source IP, destination IP, source port, destination port, protocol) modulo the number of replicas. This technique ensures consistency (i.e., packets from the same TCP connection are sent to the same replica) because packets of the same TCP connection will have the same immutable packet header fields, thus hashing to the same replica. Also, the technique does not require storing any state per TCP connection: it only requires some constant state, i.e., what kind of hash to compute, which header fields to hash, and how many replicas exist.

(v) Besides consistent load-balancing decisions, you realize that their solution is missing another key element to work properly. Briefly explain. (1 Point)

Solution: It is missing load-awareness, which can result in imbalanced replica loads: if there are a few large TCP connections, hash collisions can cause all connections to be mapped to the same replica, overwhelming this replica and leaving the rest idle.

Alternative accepted solution: The replica needs to rewrite the source IP in reply packets (previously the destination IP) back to the IP of the load balancer (129.132.19.215) instead of using their own IP. Otherwise, the client sending the initial packets will not recognize the incoming replies.

Extra Sheet 1

In case you need more space, use the following pages. Make sure to always indicate the task to which the answer belongs (e.g., *3 d) (ii)*).

Task: _____

Task: _____

Extra Sheet 2

Task: _____

Task: _____

Extra Sheet 3

Task: _____

Task: _____