ETH zürich

Networked Systems
ETH Zürich — seit 2015

D-ITET | Prof. Dr. Laurent Vanbever

February 2023 | R. Jacob, E. Kirci, A. Gran Alcoz, R. Schmid, E. Costa Molero

Exam: Advanced Topics in Communication Networks

14 February 2023, 15:30–17:30, Room ETF C 1

- ▷ Write your name and your ETH student number below on this front page and **sign it**.
- ▷ Put your **legitimation card** (legi) on the most accessible corner of your desk. Make sure that the side containing your name and **student number is visible**.
- ▷ Verify that you have received all task sheets (Pages 1 - 25).
- ▷ **Do not separate** the task sheets. We will collect the exams after you left the room.

- ▷ Write your answers directly on the task sheets.
- ▷ All answers fit within the allocated space—often in much less.
- ▷ If you need more space, use the **extra sheets** at the end of the exam. **Indicate the task** in the corresponding field, and add a **"see Extra Sheet X"** note in the original task space.
- ▷ **Read each task completely before you start solving it.**
- ▷ It is not required to score all points to get the best mark.

- ▷ Answer in **English**.
- ▷ **Write clearly** in blue or black ink (not red) using a **pen**, not a pencil.
- ▷ **Cancel** invalid parts of your solutions **clearly** (e.g., by crossing them out).
- ▷ At the end of the exam, **place the exam face up** on the most accessible corner of your desk. Then collect all your belongings and **exit the room** according to the given instructions.

- ▷ No written material nor calculator are allowed.

Family name:	Student legi nr.:
First name:	Signature:

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

Task	Points
Programmable data planes	/23
Managing network traffic	/20
Optimizing network performance	/37
Design question	/40
Total	/120

leave blank

leave blank

Task 1: Programmable data planes 23 Points

a) General Questions (8 Points)

For each of the following statements, indicate whether they are *true* or *false*. There is always one correct answer. This block of questions grants up to 4 points: 4 points for four correct answers, 2 points for two correct answers, and 0 points otherwise.

(i) P4 language, architecture, and programs (4 Points)

true ☐ false ☐ In a P4 program, all emitted headers must have been parsed before.

true ☐ false ☐ In P4, every header can be parsed at most once.

true ☐ false ☐ P4 can only update, but not compute checksums over an entire packet.

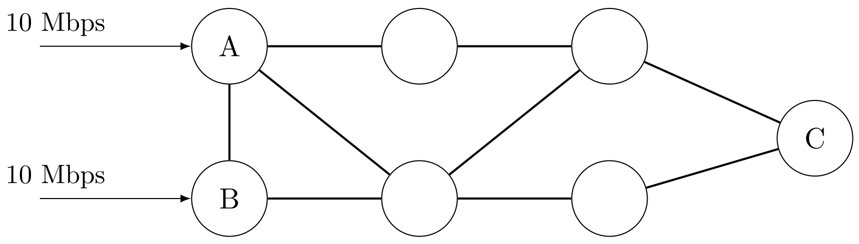
true ☐ false ☐ In P4, limiting a traffic flow to a fixed rate requires stateful objects.

(ii) Load balancing (2 Points)

Consider the network below:

- All links have a capacity of 10 Mbps, equal delay, and equal OSPF costs.
- 10 Mbps of traffic toward C enters at A and B, respectively.
- All routers run per-packet ECMP.

Annotate each link with its expected traffic volume.



(iii) Forwarding (2 Points)

This block grants up to 2 points: 2 points for two correct answers, and 0 points otherwise.

true ☐ false ☐ Consider the network depicted under "(ii) Load balancing."
If A and B are the network's label edge routers (LER) for a BGP-free core network, then all packets destined for C's IP address belong to the same forwarding equivalence class (FEC).

true ☐ false ☐ Flowlet-based load balancing improves over per-packet ECMP by reducing the chances of packet reordering.

b) Probabilistic data structures (15 Points)

leave blank

In this task, we consider the implementation of a Bloom filter on real hardware.

On this hardware, you can **only read once and modify once** the value of a single register cell, for each register. You can do basic comparisons and computations between the read and modify. To work around this single-access-per-packet limitation, you implemented a bloom filter where each hash function $h_i(x)$ is mapped to a different register R_i. The hash functions for this task are defined as follows:

$$h_1(x) = 2x + 3 \mod 10$$
$$h_2(x) = 5x - 1 \mod 10$$

where x is a numerical input value. In this task, we will input words and use the sum of the letters' ASCII values as x. The following tables contain all the numerical values you need.

	x	$2x+3$	$5x-1$
AdvNet	578	1159	2889
is	220	443	1099
a	97	197	484
great	531	1065	2654

	x	$2x+3$	$5x-1$
boring	641	1285	3204
lecture	756	1515	3779
by	219	441	1094
NSG	232	467	1159

(i) Update the Bloom filter by inserting the words *'AdvNet'*, *'is'*, and *'great'*. Use the registers below to indicate the state of the Bloom filter after each word. (3 Points)

	$\mathbf{R_1}$	$\mathbf{R_2}$		$\mathbf{R_1}$	$\mathbf{R_2}$		$\mathbf{R_1}$	$\mathbf{R_2}$		$\mathbf{R_1}$	$\mathbf{R_2}$
0	0	0									
1	0	0									
2	0	0									
3	0	0									
4	0	0	$\rightarrow$			$\rightarrow$			$\rightarrow$		
5	0	0									
6	0	0									
7	0	0									
8	0	0									
9	0	0									

(ii) After inserting *'AdvNet'*, *'is'*, and *'great'* in the Bloom filter, which of the other words (from the tables above) would cause a "false positive" answer by the filter? What does that mean for these words, and why can this happen? (4 Points)

False positives: ______________________________

Explanation: ______________________________

____

____

(iii) Describe one advantage and one disadvantage of this workaround compared to a Bloom filter implementation with a single register. (2 Points)

Advantage: ____

Disadvantage: ____

(iv) Your hardware provides a CountMin sketch with the following API: (6 Points)

`read_min(x)`: returns the value stored for `x`
`increment(x)`: increments the counter `x`
`decrement(x)`: decrements the counter `x`

We want to use these functions to implement a Bloom filter with the following API:

`contains(x)`: check whether the set contains `x` or not
`insert(x)`: inserts `x` into the set
`delete(x)`: deletes `x` from the set

Since the hardware supports a single "read-and-modify" access to registers, you are limited to one call of `read_min(x)`, some basic computation, and then either `increment(x)` or `decrement(x)` to implement each function of your Bloom filter.
Write pseudo-code for the Bloom filter functions using only the CountMin sketch API, branching logic ("`if (...) { ... } else { ... }`") and "`return`" statements.

`contains(x)`: ____

`insert(x)`: ____

`delete(x)`: ____

leave blank

leave blank

Task 2: Managing network traffic 20 Points

a) Label switching theory (4 Points)

For each of the following statements, indicate whether they are *true* or *false*. There is always one correct answer. Each block of questions grants up to 4 points: 4 points for four correct answers, 2 points for three correct answers, and 0 points otherwise.

true	false	
☐	☐	Penultimate hop popping is a method of reducing label lookups on the egress router.
☐	☐	All routers can perform SWAP operations in an MPLS network.
☐	☐	MPLS does not support Quality of Service (QoS) mechanisms.
☐	☐	RSVP takes advantage of distance-vector algorithms to distribute the state of available bandwidth.

b) Label switching application **(16 Points)**

leave blank

In this task, you go through an RSVP example, compare packet and circuit switching in general, and strict/loose routes in RSVP-TE.

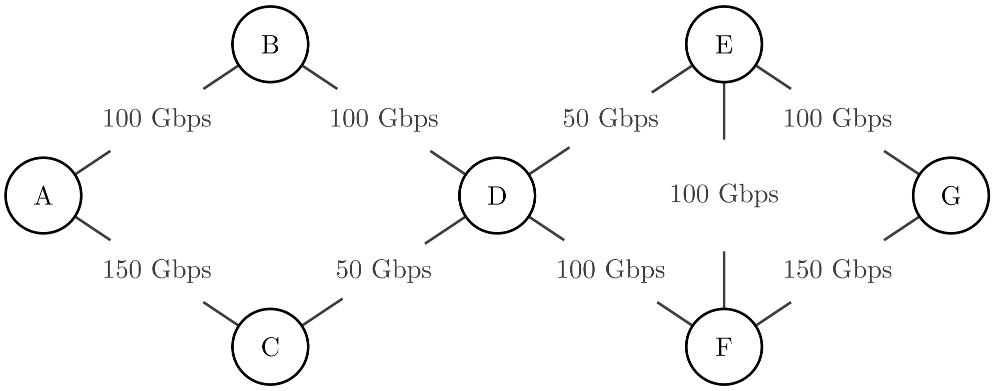


Figure 1: The topology of the network. Link annotations indicate the link's bidirectional capacity. All links have the same delay and unary cost.

(i) Consider the network topology in Fig. 1. Link annotations indicate the link's bidirectional capacity. All links have the same delay and unary cost. Find the shortest constrained path(s) for an aggregated traffic demand of 75 Gbps from Router A to Router G. Describe the steps of your reasoning. (3 Points)

Path: ____________________

Algorithm steps: ____________________

(ii) Explain two benefits of packet switching and circuit switching each. (4 Points)

Packet switching benefit 1: ____________________

Packet switching benefit 2: ____________________

Circuit switching benefit 1:

Circuit switching benefit 2:

(iii) Give and explain three distinct arguments to support the following statement: "Virtual circuit switching (e.g., MPLS) gives the best of both worlds (packet and circuit switching)." (6 Points)

Argument 1:

Argument 2:

Argument 3:

(iv) Explain what strict and loose routes are in RSVP-TE. Propose one scenario where you would pick strict route over the loose route and explain your reasoning. (3 Points)

Strict routes:

Loose routes:

Scenario:

leave blank

leave blank

Task 3: Optimizing network performance **37 Points**

a) Quality of Service (QoS) **(24 Points)**

(i) Fig. 2 depicts the structure of a QoS-enabled router. Name the different components of the router, marked with identifiers 1 to 4. Describe their function briefly. (8 Points)

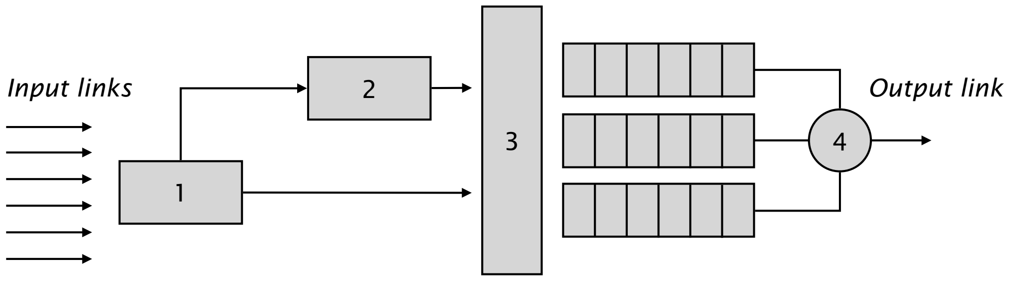


Figure 2: Building blocks of a quality-of-service (QoS) implementation in a network switch.

(1) Name: __________

(1) Function: __________

(2) Name: __________

(2) Function: __________

(3) Name: __________

(3) Function: __________

(4) Name: __________

(4) Function: __________

(ii) For each scenario below, QoS can be helpful. Describe how you would configure the QoS-enabled router to achieve the described goal. You can assume that all components from subtask (i) are fully programmable. (12 Points)

- The router gets attacked with a flood of TCP SYN packets (at its input). As a result, the output link is congested and some legitimate packets get dropped. How would you configure the QoS-enabled router to mitigate the SYN-flood attack?

- The router serves a heavy load of bursty TCP traffic. A significant number of packets get dropped and, yet, the output link is under-utilized. You suspect that it is due to TCP global synchronization. How would you configure the QoS-enabled router to break the synchronization of TCP flows and increase the output link utilization?

- The router is located in a data center and serves traffic from multiple tenants. You want to make sure that each tenant gets an equal portion of the output-link resources. How would you configure the QoS-enabled router to satisfy this objective?

- The router is located in a data center and serves TCP flows of various sizes. Packets of each flow are tagged with the total flow size. You want to maximize the number of completed flows per time interval. How would you configure the QoS-enabled router to achieve this objective?

(iii) In the setup of Fig. 2, you want to implement a token-bucket algorithm. Give a high-level explanation of how the algorithm works. (4 Points)

Explanation:

b) Fast convergence **(13 Points)**

leave blank

Consider the network topology in Fig. 3. Each link is annotated with its IGP weight. The network runs MPLS, where each router builds a label switched path (LSP) towards the others, following the IGP shortest-paths.

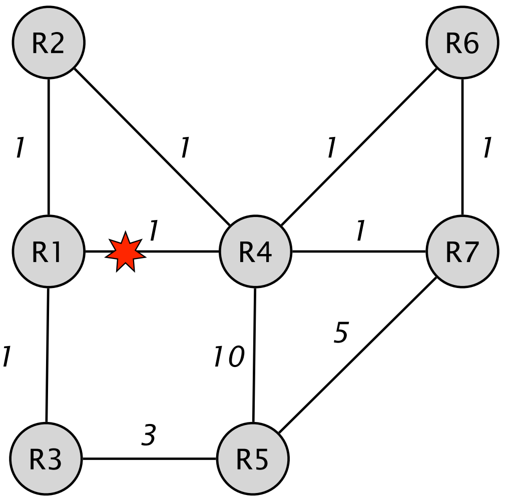


Figure 3: Network topology.

(i) Compute the shortest path from each of the different routers towards R7, and their cost. For now, do not consider any link failure. (2 Points)

Path between R1→R7: ______

Cost of R1→R7: ______

Path between R2→R7: ______

Cost of R2→R7: ______

Path between R3→R7: ______

Cost of R3→R7: ______

Path between R4→R7: ______

Cost of R4→R7: ______

Path between R5→R7: ______

Cost of R5→R7: ______

Path between R6→R7: ______

Cost of R6→R7: ______

(ii) Which router(s), if any, can be considered a LFA of R1 to reach R7, for the failure between R1 and R4? Justify your answer. (3 Points)

(iii) Does R1 have any *remote* LFA (RLFA) towards R7, for the failure between R1 and R4? Justify your answer. (3 Points)

(iv) We want to provide *router-disjoint* end-to-end LSP protection between R1 and R7. Can we achieve it with the current topology? If yes, provide (all) the possible *router-disjoint* end-to-end LSP protection path(s) between R1 and R7. If not, justify why. (3 Points)

(v) Name one advantage and one drawback of using *per-link* LSP protection with respect to LFAs. (2 Points)

Advantage:

Drawback:

leave blank

Task 4: IP Routing meets Ethernet Switching 40 Points

In this task, we ask you to design an alternative way to perform IP routing, inspired by Ethernet switching.

Traditional IP routers (operating at layer 3) learn how to forward IP packets using distributed routing protocols (e.g. OSPF or BGP). In contrast, Ethernet switches (operating at layer 2) learn how to forward Ethernet frames based *on the traffic* they observe: When they receive an Ethernet frame on port `X` sourced by a MAC address `M`, they remember to send frames to `M` onto port `X`. If they need to forward a frame for an unknown destination MAC, they simply flood it.

Your goal is to implement an IP router in P4 that learns which next hops to use like an Ethernet switch: It dynamically learns how to forward IP packets by observing the traffic and flooding IP packets when necessary.

Throughout this task, we will consider the network in Fig. 4. It contains three P4 switches (`S1`, `S2`, `S3`) connected to five IP hosts (`h1`, . . . , `h5`) located in three different subnets: `10.0.0.0/24`, `20.0.0.0/24`, and `30.0.0.0/24`. We assume that the hosts have been pre-configured with a default IP gateway (for `0.0.0.0/0`) and that their ARP tables have been pre-configured statically.

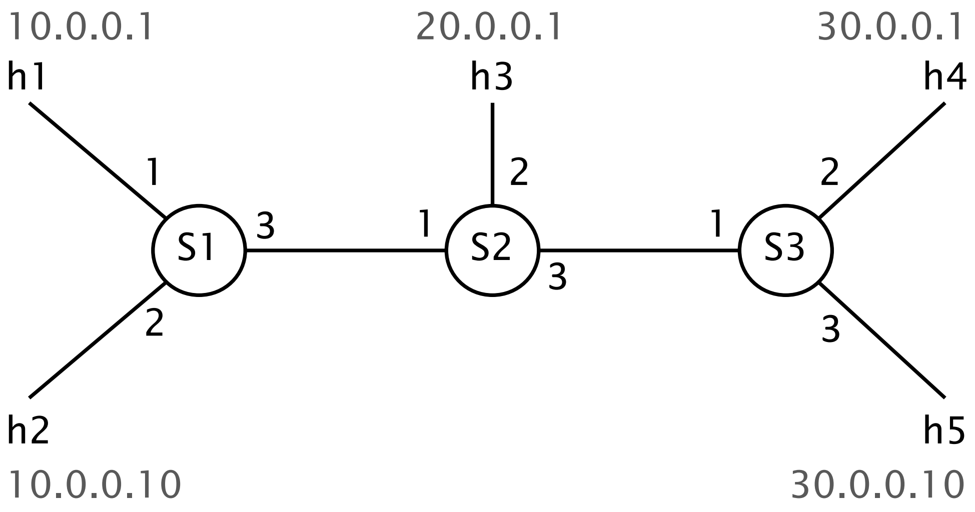


Figure 4: Network topology

a) Flooding **(5 Points)**

leave blank

Your colleague has already correctly implemented the flooding part which makes a P4 switch flood the IP packets it receives on all ports *but* the one they came on (Fig. 5). With this code in place, all hosts can ping each other.

The P4 code uses multicast groups, as seen in the exercises for L2 flooding. Your colleague also defined the content of the `select_mcast_grp` table on all switches (Fig. 6).

```
/* -*- P4_16 -*- */
#include <core.p4>
#include <v1model.p4>

const bit<16> TYPE_IPV4 = 0x800;

/********************** H E A D E R S  ********************************/

typedef bit<9>  egressSpec_t;
typedef bit<48> macAddr_t;
typedef bit<32> ip4Addr_t;

header ethernet_t {
    macAddr_t dstAddr;
    macAddr_t srcAddr;
    bit<16>   etherType;
}

header ipv4_t {
    bit<4>    version;
    bit<4>    ihl;
    bit<6>    dscp;
    bit<2>    ecn;
    bit<16>   totalLen;
    bit<16>   identification;
    bit<3>    flags;
    bit<13>   fragOffset;
    bit<8>    ttl;
    bit<8>    protocol;
    bit<16>   hdrChecksum;
    ip4Addr_t srcAddr;
    ip4Addr_t dstAddr;
}

struct headers {
    ethernet_t   ethernet;
    ipv4_t    ipv4;
}

/********************** P A R S E R  **********************************/

parser MyParser(packet_in packet,
                out headers hdr,
                inout metadata meta,
                inout standard_metadata_t standard_metadata) {

    state start {
        packet.extract(hdr.ethernet);
        transition select(hdr.ethernet.etherType){
            TYPE_IPV4: parse_ipv4;
            default: accept;
        }
    }

    state parse_ipv4 {
        packet.extract(hdr.ipv4);
        transition accept;
    }
}


/*************  I N G R E S S   P R O C E S S I N G   *****************/

control MyIngress(inout headers hdr,
                  inout metadata meta,
                  inout standard_metadata_t standard_metadata) {

    action set_mcast_grp(bit<16> mcast_grp) {
        standard_metadata.mcast_grp = mcast_grp;
    }

    table select_mcast_grp {
        key = {
            standard_metadata.ingress_port : exact;
        }
        actions = {
            set_mcast_grp;
            NoAction;
        }
        size = 32;
        default_action =  NoAction;
    }

    apply {
        select_mcast_grp.apply();
    }
}

// [...] Rest of the p4 program is trimmed (not relevant)
```

Figure 5: P4 code skeleton

```
mc_node_create 0 2 3
mc_node_create 1 1 3
mc_node_create 2 1 2

mc_mgrp_create 1
mc_node_associate 1 0

mc_mgrp_create 2
mc_node_associate 2 1

mc_mgrp_create 3
mc_node_associate 3 2

table_add select_mcast_grp set_mcast_grp 1 => 1
table_add select_mcast_grp set_mcast_grp 2 => 2
table_add select_mcast_grp set_mcast_grp 3 => 3
```

Figure 6: Configuration of the Multicast group and initial table content

Explain in English what the `select_mcast_grp` table does, what the rules in the table mean, and why they lead to traffic being multicasted on all ports but the incoming one.

(5 Points)

Role of the `select_mcast_grp` table: ____________________

Semantic of the rules: ____________________

Why do the rules work: ____________________

b) Learning *without* collision (15 Points)

leave blank

Now that you have a way to flood IP packets, you turn your attention to IP learning.

As for L2 learning, the idea is to have the P4 switches remember the incoming port onto which they receive an IP packet sourced by `X`, and use it as the outgoing port to use when they receive an IP packet destined to `X`. If the P4 switch does not know how to reach `X`, it should flood the packet on all the ports but the incoming one.

Unlike normal L2 learning (the one we have seen in the exercises), we ask you to implement a solution that does ***not*** rely on a controller that modifies match-action tables. Instead, the P4 switch should remember how to reach each source IP *using data plane state* only.

You think of using a hash table which will store the output port to use for each IP address. You define a register `ipv4_forwarding_table` (line 7 in the code below). As you can see, port numbers are stored on 9 bits. For simplicity, you can assume (for now) that hash collisions *never* happen. You can also assume that all the 8192 entries are initialized to 0.

Complete the ingress pipeline processing logic in the P4 code below (Fig. 7) to implement the IP learning as described above.

Hint: We provide you with the documentation to access registers (`read`, `write`) as well as how to calculate `hash` in Fig. 8.

```
 1 control MyIngress(inout headers hdr, inout metadata meta,
 2                   inout standard_metadata_t standard_metadata) {
 3
 4     register<bit<9>>(8192) ipv4_forwarding_table;
 5
 6     action set_mcast_grp(bit<16> mcast_grp) {
 7         standard_metadata.mcast_grp = mcast_grp;
 8     }
 9
10     table select_mcast_grp {
11         key = {
12             standard_metadata.ingress_port : exact;
13         }
14         actions = {
15             set_mcast_grp;
16             NoAction;
17         }
18         size = 32;
19         default_action =  NoAction;
20     }
21
22     apply { // FILL ME IN
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
```

```











    }
}
```

Figure 7: P4 code skeleton

```
extern register<T>;
void read(out T result, in I index);
void write(in I index, in T value);

/***
 * A register object is created by calling its constructor.  This
 * creates an array of 'size' identical elements, each with type
 * T.  The array indices are in the range [0, size-1].  For
 * example, this constructor call:
 *
 *     register<bit<32>>(512) my_reg;

 * allocates storage for 512 values, each with type bit<32>.
 */

extern void hash<O, T, D, M>(out O result, in HashAlgorithm algo, in T base,
     in D data, in M max);

/***
 * Calculate a hash function of the value specified by the data
 * parameter.  The value written to the out parameter named result
 * will always be in the range [base, base+max-1] inclusive, if max >=
 * 1.  If max=0, the value written to result will always be base.
 *
 * @param O          Must be a type bit<W>
 * @param D          Must be a tuple type where all the fields are bit-
   fields (type bit<W> or int<W>) or varbits.
 * @param T          Must be a type bit<W>
 * @param M          Must be a type bit<W>
 *
 * For example:
 *
 * hash(output, HashAlgorithm.crc16, (bit<32>)0, {hdr.ipv4.srcAddr,
 *                                                 hdr.ipv4.dstAddr,
 *                                                 hdr.tcp.srcPort,
 *                                                 hdr.tcp.dstPort,
 *                                                 hdr.ipv4.protocol},
 *                                                 (bit<32>)8192);
 *
 * computes a hash value of the 5-tuple between 0 and 8192 and store the
   results in the variable output.
 */
```

Figure 8: P4 documentation on how to use the `hash` and `register` functions

leave blank

c) Learning *with* (a bounded amount of) collisions **(10 Points)**

We now relax the assumption that hash collisions never happen. We ask you to devise a scheme that allows the P4 switches to ensure correct forwarding *even* if there is a hash collision. For simplicity though, we assume that at most two distinct IP addresses may collide on the same hash index.

In the following, we ask you to explain in English and in details how you would modify your solution to: (i) *detect* collisions (for up to 2 IP addresses); **and** (ii) *handle* them so that forwarding is still performed correctly.

Be specific in your answers: if you adapt the data structures, explain how.

(i) Your proposed modifications to *detect* collisions: (4 Points)

(ii) Your proposed modifications to *handle* collisions: (6 Points)

leave blank

d) Learning leveraging the IP hierarchy **(10 Points)**

We finally want to improve the routing scheme by reducing the amount of flooding required.

To do so, we will leverage that IP addresses are hierarchical: They can be grouped by their longest common prefix (contrary to MAC addresses). For instance, if a P4 switch observes traffic for `10.0.0.1` *and* `10.0.0.200` arriving on the same output port, it could infer that traffic for any other thus-far-unseen IP addresses in `10.0.0.0/24` should leave via this port as well (e.g., for `10.0.0.10`) instead of flooding it.

In practice, this inference may sometimes be wrong; e.g., it could be that the traffic for `10.0.0.10` should go through another port than the traffic for `10.0.0.1` and `10.0.0.200`. For simplicity though, we assume here that `/24`-prefix-based inference is *always* correct.

We ask you to implement a feature that periodically learns the corresponding `/24`-prefix entries and download them into the switch. This feature should work alongside the solution from the previous questions. If you have not answered the previous questions, you may assume that IP learning is working as specified.

In this sub-task, you *must* use the controller and add an extra match-action table.

(i) Explain in English and *in details* how you would implement this feature using the controller and a new table. (5 Points)

(ii) Write down the table definition in P4. (2 Points)

(iii) Explain the new table control flow; that is, explain in details how you would chain tables together. (3 Points)