Chair of Network Architectures and Services
Department of Informatics
Technical University of Munich

Note:
- During the attendance check a sticker containing a unique code will be put on this exam.
- This code contains a unique number that associates this exam with your registration number.
- This number is printed both next to the code and to the signature field in the attendance check list.

Advanced Computer Networking

Exam:	IN2097 / Retake	**Date:**	Wednesday 4th April, 2018
Examiner:	Prof. Dr.-Ing. Georg Carle	**Time:**	11:30 – 12:30

	P 1	P 2	P 3	P 4	P 5	P 6
I						
II						

Working instructions

- This exam consists of
  - **16 pages** with a total of **6 problems** and
  - a two-sided printed **cheat sheet**.

  Please make sure now that you received a complete copy of the exam.
- Detaching pages from the exam is prohibited.
- Subproblems marked by * can be solved without results of previous subproblems.
- **Answers are only accepted if the solution approach is documented.** Give a reason for each answer unless explicitly stated otherwise in the respective subproblem.
- Do not write with red or green colors nor use pencils.
- The total amount of achievable credits in this exam is 60 credits.
- Allowed resources:
  - one **analog dictionary** English ↔ native language
- Physically turn off all electronic devices, put them into your bag and close the bag.

Left room from ________ to ________ / Early submission at ________

Problem 1 **Quiz (6.5 credits)**

The following questions cover multiple topics and can be solved independently of each other.

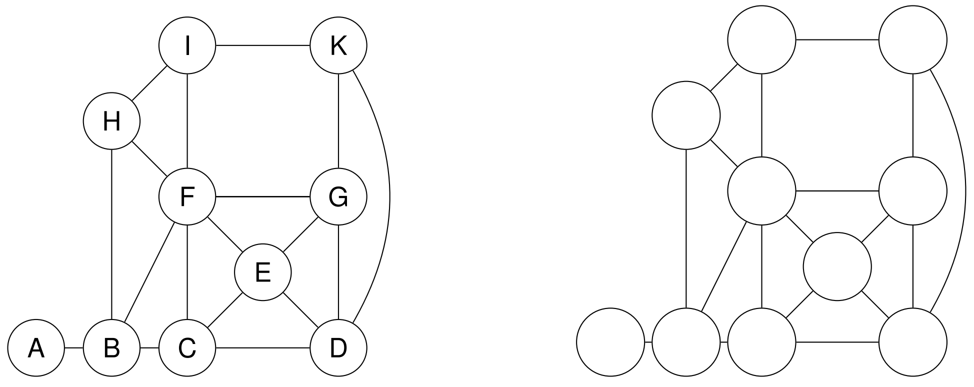
a)* There exist multiple technologies to create virtual networks. Name **two** important differences between networks using VLAN (IEEE 802.1q) and networks using IPsec.

b)* Name three implementation or architectural differences between the measurement tools nmap and ZMap.

c)* The TCP congestion algorithm BBR achieves lower round trip times compared to Cubic. How does BBR achieve this?

d)* Perform the *k*-core algorithm for the topology shown in the solution box.

1. List the *k* value for every node in the graph, i. e. the current *k* value when the node is removed.
2. Mark the core of the network after applying the *k*-core algorithm.



Problem 2 Poor man’s wireshark (8.5 credits)

This problem investigates the initial octetts of an hexdump of an Ethernet frame given in Figure 2.1.

```
0x0000  8a  dd  ee  2a  67  ee  a2  f5    af  79  34  2b  81  00  03  fc

0x0010  08  00  45  c0  00  58  32  a6    00  00  40  01  c5  eb  0A  09

0x0020  00  02  0A  09  00  01  03  03    6a  84  00  00  00  00  ..  ..
```

Figure 2.1: Hexdump of an incomplete Ethernet frame excluding FCS

a)* Mark and name the fields of the Ethernet frame present in Figure 2.1.

0 1 2

b) Mark the address of the L3 sender and the L3 receiver. Report them in their typical address format.

0 1 2

c) Mark the start of the payload.

0 ½

d) Reason what kind of payload it is.

0 1

e) Give a possible reason why this specific kind of message could be created.

0 1 2

Problem 3 Routing (13.5 credits)

Table 3.1 shows the entries of a routing table:

Subnet/Prefix	Next hop interface	Next hop IP address (simplified)
`2001:db80::/128`	0	`A`
`2001:db81::/96`	1	`B`
`2001:db82::/64`	2	`C`
`2001:db82::/32`	3	`D`
`::/0`	4	`E`

Table 3.1: Routing table entries

a)* Name and describe the algorithm used to determine the next hop for a given IP address.

b)* What is a network mask?

The lookup algorithm in Listing 1 determines the next hop for an IP address (`matchIP`).

Listing 1: Routing algorithm (pseudo code)

```
def routes = [
    # contains the entries of Table 2.1 in the given order as 128 bit integers
    {subnet = 0x2001 db80 0000 0000 0000 0000 0000 0000,
     ntmask = 0xFFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF,
     nxthop = {nxtHopInterface =  0, nextHopAddr = A}}, ...
]

def matchWrong(subntAdr, matchAdr):
    return (subntAdr == matchAdr)

def lookup(matchIP):
    for route in routes:
        if matchWrong(route.subnet, matchIP):
            return route.nxthop
    return None
```

c)* Have a look at Listing 1, perform a lookup according to the given algorithm, and write down the return values.

Address to look up	Result
`2001:db80::`	
`2001:db82::1`	

d) Subproblem c) may have results differing from what is expected from a lookup algorithm. Explain both results of Subproblem c).

0 1 2

Listing 1 uses a wrong matching algorithm `matchWrong`.

e) Create the `matchCorrect()` function which detects if an IP address (`matchAdr`) is part of a subnet (`subntAdr`). The function should return `True` if `matchAdr` is part of `subntAdr` and false in any other case. **Remark:** Assume that `subntAdr` and `matchAdr` are 128 bit integers.

0 1 2 3 4

f) Create an updated lookup function utilizing `matchCorrect()`.

0 1 2

g) Even with the correct matching function the code in Listing 1 can only work correctly if the routes are sorted according to their prefix length. Explain why.

0 1

Problem 4 IPFIX Software Developer (8 credits)

You are the developer of a software which exports IPFIX traffic from observed network packets. The software does not yet behave as it should. Consequently, you start to investigate the source code to find the root cause of this strange behavior.

a)* In the source code you find some lines tackling flow expiration. Describe when an active timeout is triggered and when an inactive timeout is triggered.

Active timeout is triggered when

Inactive timeout is triggered when

In the code you find two constants describing the active and inactive timeouts in seconds:

```
ACTIVE_TIMEOUT = 120
INACTIVE_TIMEOUT = 600
```

Listing 2: Timeout values in seconds

b) Are the chosen values in Listing 2 sensible? Explain your answer.

c)* You dive deeper into the code and find some lines dealing with biflow aggregation. Briefly describe what biflow aggregation is.

The following code is responsible for creating a unique flow identifier based on the IP-5-tuple when biflow aggregation is active.

```python
def flow_id_biflow(flow):
    ip_forward_direction = min(flow.src_ip, flow.dst_ip)
    ip_backward_direction = max(flow.src_ip, flow.dst_ip)
    proto = flow.proto
    port_forward_direction = min(flow.src_port, flow.dst_port)
    port_backward_direction = max(flow.src_port, flow.dst_port)

    return str(ip_forward_direction) + str(ip_backward_direction) + \
           str(proto) + str(port_forward_direction) + \
           str(port_backward_direction)
```

Listing 3: Biflow aggregation code snippet

d) You immediately identify multiple conceptual flaws in the code in Listing 3. Explain the **two** reasons why biflow aggregation does not work as intended.

0
1
2
3
4

Problem 5 Performance measurements (11 credits)

0
1
2

a)* What is the packet rate (packets per second) on a 10 Gbit/s Ethernet connection if the all packets have a size of 105 Byte. The packet size of 105 Byte already includes the frame check sequence (FCS). **Remark:** Preamble + SFD + IPG = 20 Byte.

Packet processing costs, such as cycles needed for the processing of a packet on a CPU, can roughly be divided in two components:

1. per-packet costs: costs occur per packet, independent of packet length,
2. per-byte costs: costs occur per byte in the packet.

0
1
2

b)* Consider two packet processing tasks, routing and IPsec tunneling. Argue which component dominates for which task.

0
1
2

c)* Consider a router which is promoted by the vendor of being capable of a throughput of 10 Gbit/s. Why could this promise of the vendor be misleading and what would be a sensible performance metric.

Figure 5.1 shows a latency histogram of a software switch. The latency of 100 packets was measured between the ingress and the egress interface. Assume that the shape of the latency histogram does not change even for longer experiments or for higher load, i.e., the distribution for 1 million packets looks the same as given in Figure 5.1 (apart from higher numbers as more events were measured) .

0
1

d)* Reason what may be the cause that the switch has a minimum latency of 40 µs?

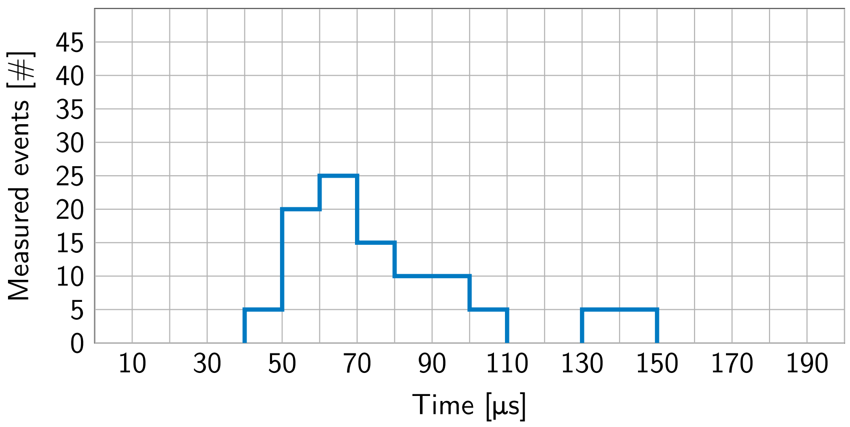


Figure 5.1: Latency histogram for 100 packets

e) Create a CDF (in Figure 5.2) from the latency histogram given in Figure 5.1.

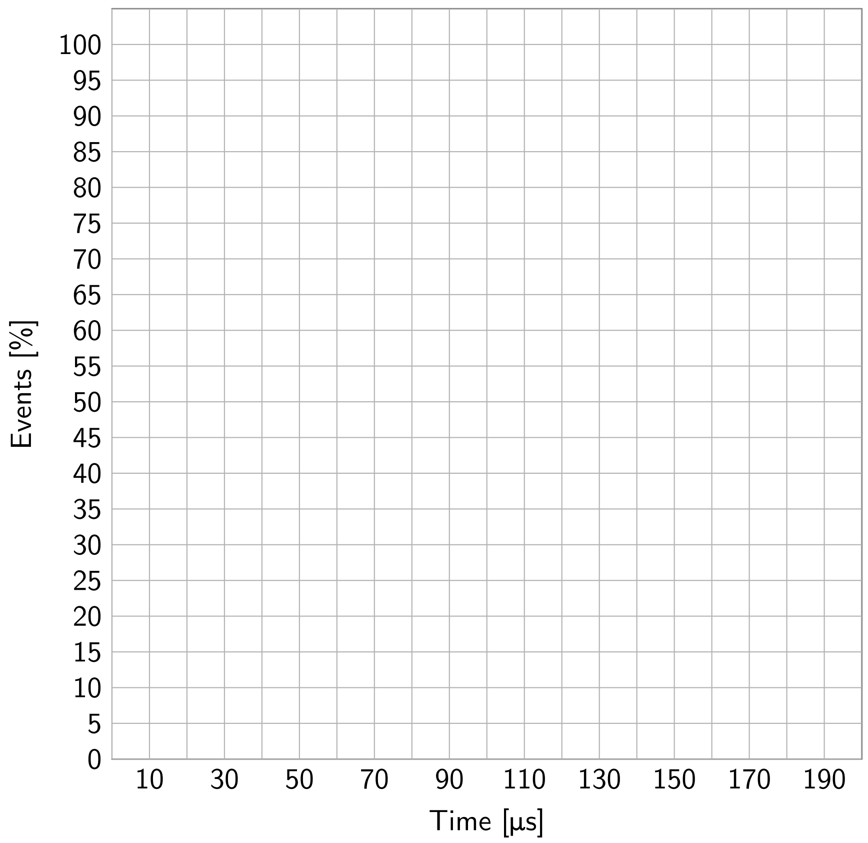


Figure 5.2: CDF

f) Determine the probability of no packet having a latency of more than 120 µs for a sequence of 3 packets sent. **Note:** Document your approach and calculate the value.

Problem 6 Traceroute (12.5 credits)

For this problem the network topology given in Figure6.1 is investigated.

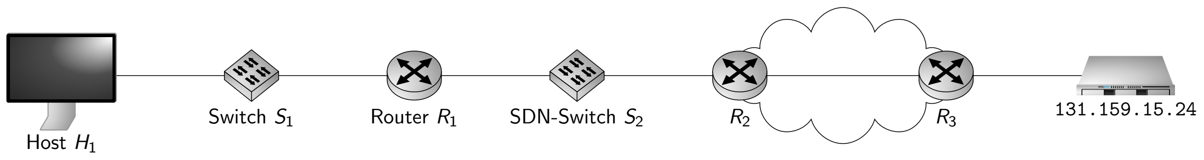


Figure 6.1: Network topology

Switch S_1 is a regular switch without SDN capability. Switch S_2 is an SDN switch where the default rules were deleted. After that the rules specified in Listing 4 were installed. If no rules are available for a packet, the S_2 is configured to drop the packet.

```
ovs-ofctl add-flow S2 dl_type=0x0806,actions=flood
ovs-ofctl add-flow S2 dl_type=0x0800,nw_proto=6,actions=flood
```

Listing 4: OpenFlow rules installed on S_2

Remark: `nw_proto` specifies the entry of the protocol field on the network layer protocol. The cheatsheet contains the values for different protocols.

0 1 2

a)* Explain the rules specified in Listing 4.

- `add-flow S2`:
- `dl_type=0x0806`:
- `dl_type=0x0800`:
- `nw_proto=6`:
- `actions=flood`:

Host H_1 successfully loaded a website located on the server with the address `131.159.15.24`. To investigate the path the packet travels through the network, the tool `traceroute` is used.

0 1

b)* Traceroute can utilize different protocols, e.g., ICMP and TCP. Explain one advantage using TCP over ICMP for `traceroute`.

Hints for traceroute:

- IP addresses can be abbreviated by `devicename.ip`, e.g. H_1.ip (Routers can have several interfaces ip1 and ip2, use one of them consistently throughout one traceroute run)
- Response times can be chosen arbitrarily.
- Traceroute generates one query for every node and queries up to 7 nodes.
- `-I` parameter enables queries via ICMP.
- `-T` parameter enables queries via TCP.

c)* Create a sample output for the command `traceroute -I 131.159.15.24` executed on Host H_1.

0 1 2

d) Describe your output for `traceroute -I 131.159.15.24`, explaining the involved protocols and flow rules (see Listing 4).

0 1 2

e) Create an output for the command `traceroute -T 131.159.15.24` executed on Host H_1.

0 1 2

f) Describe your output for `traceroute -T 131.159.15.24`, explaining the involved protocols and flow rules (see Listing 4).

0 1 2

g) Create an OpenFlow rule which fixes the problem observed in Subproblem c). Do not allow more protocols than absolutely necessary to fix the problem.

0 1

Additional space for solutions–clearly mark the (sub)problem your answers are related to and strike out invalid solutions.