

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.

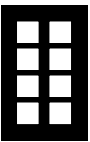
Left room from _____ to _____ / Early submission at _____

Problem 1 Quiz (6.5 credits)

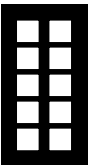
The following questions cover multiple topics and can be solved independently of each other.

0 1  a)* There exist multiple technologies to create virtual networks. Name **two** important differences between networks using VLAN (IEEE 802.1q) and networks using IPsec.

- VLAN operates on Layer 2, IPsec operates on Layer 3.
- IPsec offers encryption, VLAN does not.

0 1  b)* Name three implementation or architectural differences between the measurement tools nmap and ZMap.

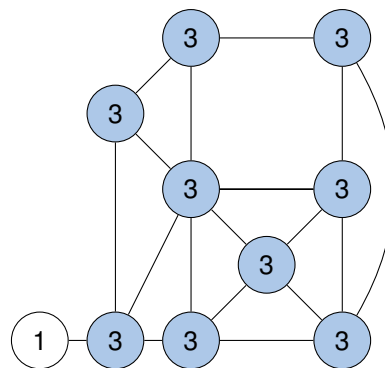
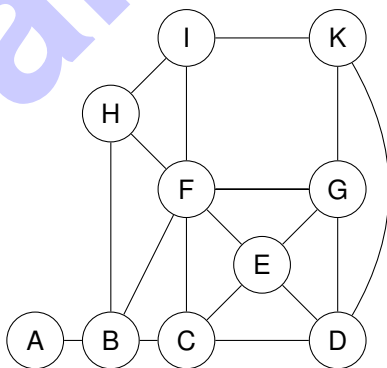
stateless architecture/approach, no timeout detection, randomization based on multiplicative group, no sophisticated protocol exchanges, no TCP payload, keep track of sent packets, Internet-wide scans possible, zmap can be distributed/sharding

0 1 2  c)* The TCP congestion algorithm BBR achieves lower round trip times compared to Cubic. How does BBR achieve this?

- BBR avoids filling buffers compared to Cubic to lower the latency
- BBR regularly measures the connection and adapts send rate to keep low buffer fill states

0 1 2  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 octets of an hexdump of an Ethernet frame given in Figure 2.1.

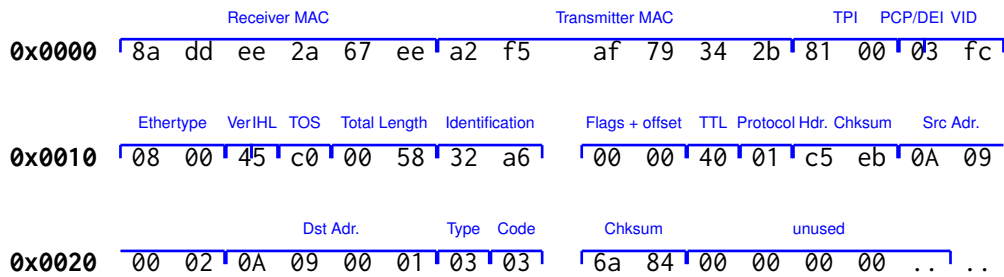
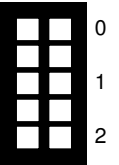
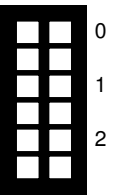


Figure 2.1: Hexdump of an incomplete Ethernet frame excluding FCS

- a)* Mark and name the fields of the Ethernet frame present in Figure 2.1.
 b) Mark the address of the L3 sender and the L3 receiver. Report them in their typical address format.

Source: 10.9.0.2, Destination: 10.9.0.1



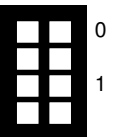
- c) Mark the start of the payload.

Payload of the IPv4 packet starts at octett number 0x0026.



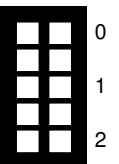
- d) Reason what kind of payload it is.

The packet is a ICMPv4 (IPv4 protocol field: 01)



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

Type of the message: 0x03 Destination unreachable
 Code of the message: 0x03 Destination port unreachable
 Port is not reachable, because it may be blocked by a firewall or the application is not running

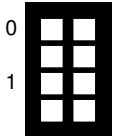


Problem 3 Routing (13.5 credits)

Table 3.1 shows the entries of a routing table:

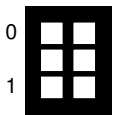
Subnet/Prefix	Next hop interface	Next hop IP address (simplified)
2001:db80::/128	0	A
2001:db81::/96	1	B
2001:db82::/64	2	C
2001:db82::/32	3	D
::/0	4	E

Table 3.1: Routing table entries



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

The algorithm is called longest prefix matching (LPM).
This algorithm determines the most specific, i.e. longest matching entry of in a routing table



b)* What is a network mask?

The network mask is a sequence of bits where the bits of the network part are set to 1.

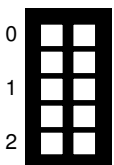
The lookup algorithm in Listing f) determines the next hop for an IP address (matchIP).

Listing 1: Routing algorithm (pseudo code)

```

1 def routes = [
2     # contains the entries of Table 2.1 in the given order as 128 bit integers
3     {subnet = 0x2001 db80 0000 0000 0000 0000 0000 0000,
4      ntmask = 0xFFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF,
5      nxthop = {nxthopInterface = 0, nextHopAddr = A}}, ...
6 ]
7
8 def matchWrong(subntAdr, matchAdr):
9     return (subntAdr == matchAdr)
10
11 def lookup(matchIP):
12     for route in routes:
13         if matchWrong(route.subnet, matchIP):
14             return route.nxthop
15     return None

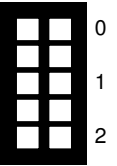
```



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

- Lookup 2001:db80:: : interface 0 with next hop addr. A
- Lookup 2001:db82::1 : None

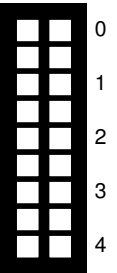
d) Subproblem c) may have results differing from what is expected from a lookup algorithm. Explain both results of Subproblem c).



- The first lookup of Subproblem c) is correct the second one is incorrect
- The reason for that is `matchWrong()` only results true for identical addresses. However, only the network part of subnet and matching address must match.

Listing f) uses a wrong matching algorithm `matchWrong`.

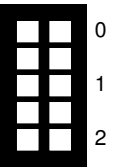
e) Create the `matchCorrect()` function which detects if an IP address (`matchAdr`) is part of a subnet (`subntAdr`). The function should return `True` if `matchAdr` is part of `subntAdr` and `false` in any other case. **Remark:** Assume that `subntAdr` and `matchAdr` are 128 bit integers.



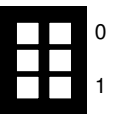
```
1
2 def match(subnetAddr, networkMask, matchAdr):
3     return subnetAddr == (matchAdr & networkMask)
```

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

```
1
2 def lookup(matchIP):
3     for route in routes:
4         if matchCorrect(route.subnt, route.ntmask, matchIP):
5             return route.nxthop
6     return None
```



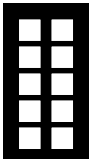
g) Even with the correct matching function the code in Listing f) can only work correctly if the routes are sorted according to their prefix length. Explain why.



The lookup returns the first matching entry. For the first match to be the most specific one the longest prefixes must precede shorter prefixes.

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.

0  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.


1
2

- Active timeout is triggered when the maximum duration of a flow is reached, even though packets are still arriving.
- Inactive timeout is triggered when no packets arrive for a certain time.

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


```
1 ACTIVE_TIMEOUT = 120  
2 INACTIVE_TIMEOUT = 600
```

Listing 2: Timeout values in seconds

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

1

No, the values are not sensible. The active timeout needs to be larger than the inactive timeout if it should have any effect.

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

1

Biflow aggregation combines request-response traffic into the same flow identifier.

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

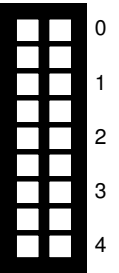
```
1 def flow_id_biflow(flow):
2     ip_forward_direction = min(flow.src_ip, flow.dst_ip)
3     ip_backward_direction = max(flow.src_ip, flow.dst_ip)
4     proto = flow.proto
5     port_forward_direction = min(flow.src_port, flow.dst_port)
6     port_backward_direction = max(flow.src_port, flow.dst_port)
7
8     return str(ip_forward_direction) + str(ip_backward_direction) + \
9           str(proto) + str(port_forward_direction) + \
10          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.

There are two problems with the code:

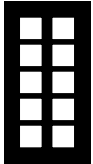
- The direction calculation and swapping needs to be done **jointly** for IP address and port. In the current code, however, it is done independently.
- The elements making up the flow ID need to be separated by e.g. a comma. In the current code, however, there is no separation e.g. between protocol number and port.



Sample Solution

Problem 5 Performance measurements (11 credits)

0
1
2



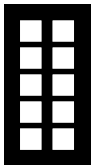
a)* What is the packet rate (packets per second) on a 10 Gbit/s Ethernet connection if the all packets have a size of 105 Byte. The packet size of 105 Byte already includes the frame check sequence (FCS). **Remark:** Preamble + SFD + IPG = 20 Byte.

$$\frac{10 \text{ Gbit}}{(105 \text{ B} + 20 \text{ B}) \cdot 8} = 10 \text{ Mpps}$$

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.

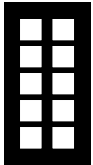
0
1
2



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

- Routing: only header of packet is processed for routing, therefore routing has almost constant per-packet costs.
- IPsec tunneling: encrypts the entire packet therefore costs are increasing with every byte to encrypt.

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.

- Router have dominating per packet cost, therefore a throughput measured in Gbit/s might be misleading
- If the throughput was measured for large packets, e.g. 1500 Byte the router might not be able to fully load the 10 Gbit/s when using minimum sized packets of 64 Byte

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 μ s?

- Processing introduces minimal delay , no queuing (would change under load), no serialization/propagation as 40 μ s is way to high for that to be the cause.

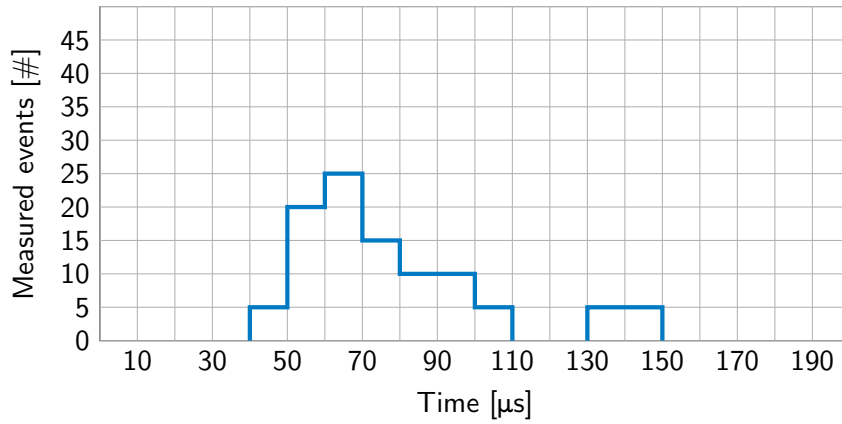


Figure 5.1: Latency histogram for 100 packets

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

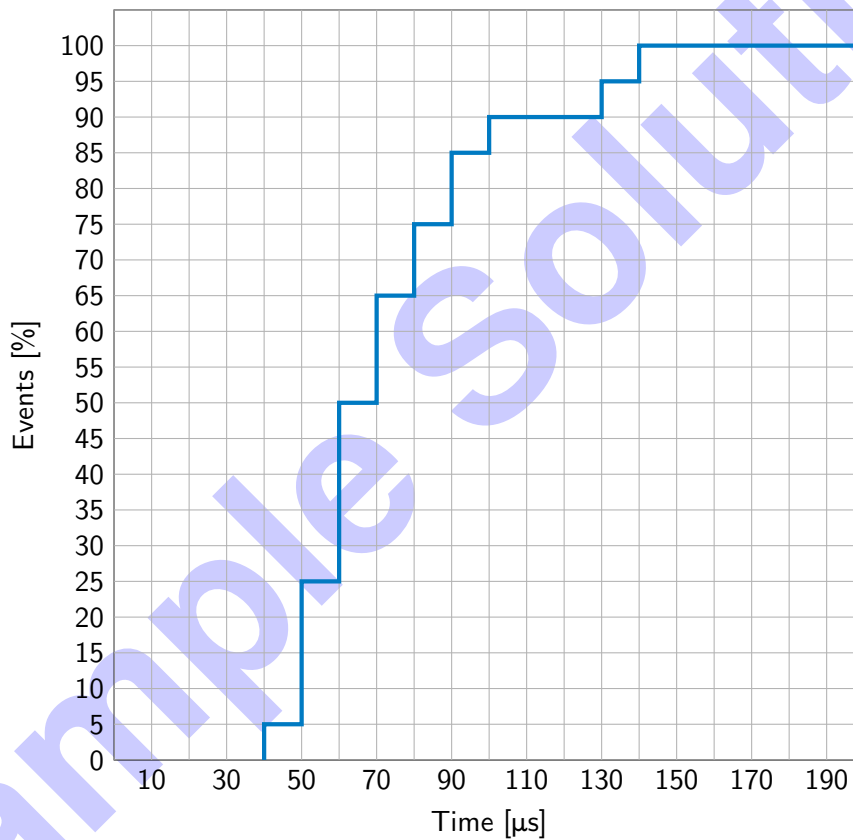
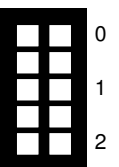
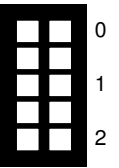


Figure 5.2: CDF

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.

$$0.90^3 = 0.729$$



Problem 6 Traceroute (12.5 credits)

For this problem the network topology given in Figure 6.1 is investigated.

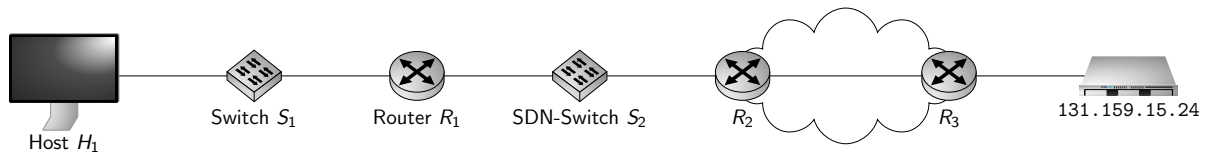


Figure 6.1: Network topology

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.

```
1 ovs-ofctl add-flow S2 dl_type=0x0806 , actions=flood
2 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.

a)* Explain the rules specified in Listing 4.

- `add-flow S2`: installs rule on switch S_2
- `dl_type=0x0806`: rule is executed for packets with Ethertype 0x0806 (ARP)
- `dl_type=0x0800`: rule is executed for packets with Ethertype 0x0800 (IPv4)
- `nw_proto=6`: rule is executed for TCP segments (protocol type 0x6)
- `actions=flood`: packets are forwarded on every switch port except the receiving one

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.

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

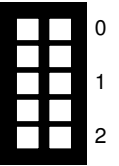
- TCP port 80 (HTTP) is often allowed even for firewalled networks whereas ICMP is not

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.

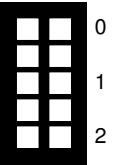
c)* Create a sample output for the command `traceroute -I 131.159.15.24` executed on Host H_1 .

- 0: [R_1 .ip] 10ms
- 1: *
- 2: *
- ...
- 7: *



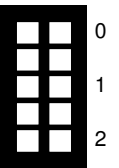
d) Describe your output for `traceroute -I 131.159.15.24`, explaining the involved protocols and flow rules (see Listing 4).

- Requests and replies use ICMP
- There is no rule for ICMP therefore ICMP cannot pass S_2
- R_1 is reached by ICMP and can answer
- No packets travel beyond R_1



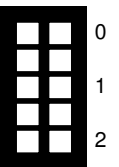
e) Create an output for the command `traceroute -T 131.159.15.24` executed on Host H_1 .

- 0: [R_1 .ip] 10ms
- 1: *
- 2: *
- ...
- 7: *



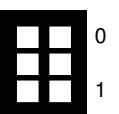
f) Describe your output for `traceroute -T 131.159.15.24`, explaining the involved protocols and flow rules (see Listing 4).

- Requests use TCP , replies ICMP
- There is no rule for ICMP therefore replies cannot pass S_2
- R_1 is reached by ICMP and can answer

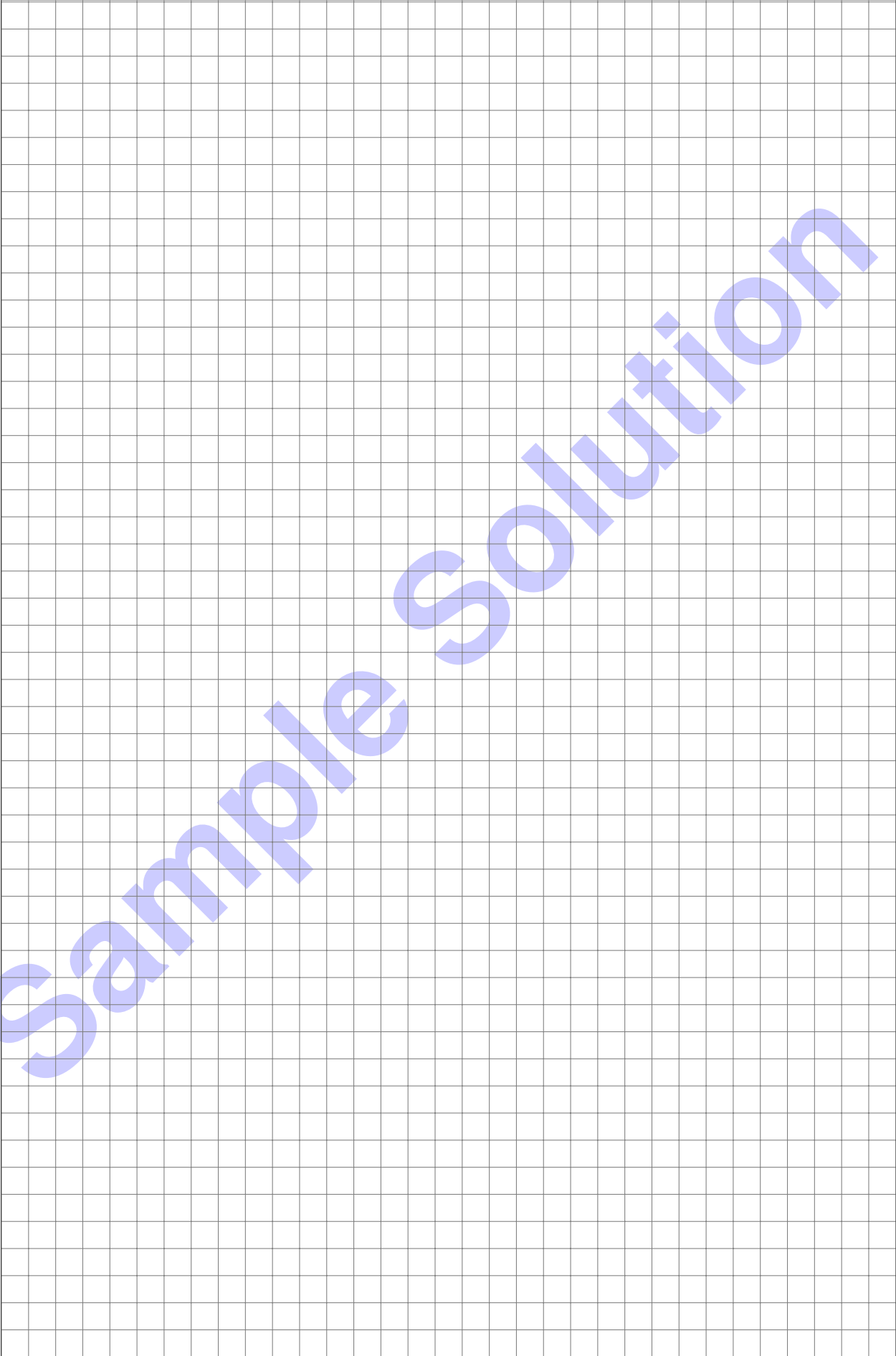


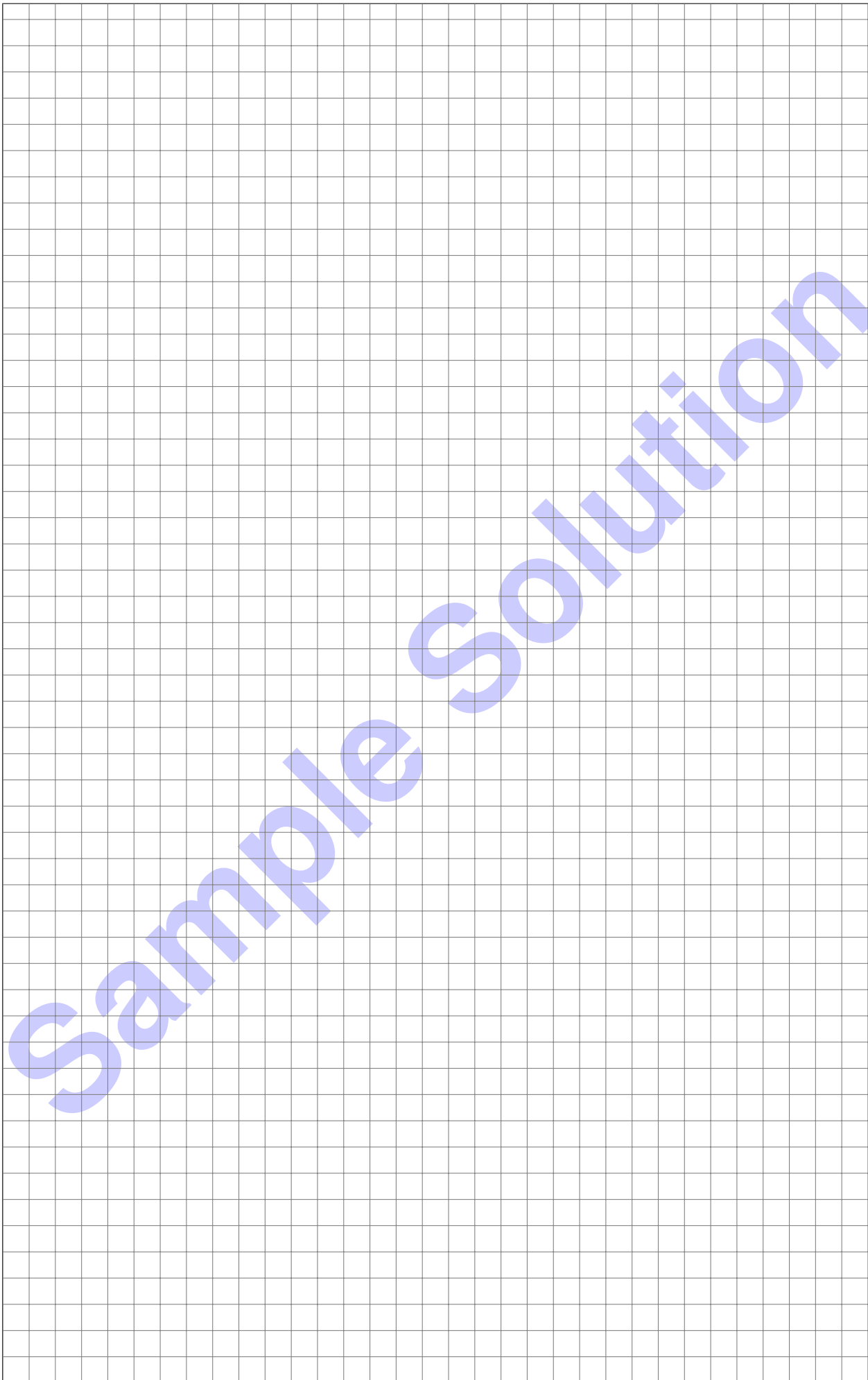
g) Create an OpenFlow rule which fixes the problem observed in Subproblem c). Do not allow more protocols than absolutely necessary to fix the problem.

```
ovs-ofctl add-flow S2 dl_type=0x0800,nw_proto=1,actions=flood
This rule allows ICMP messages to pass  $S_2$ 
```

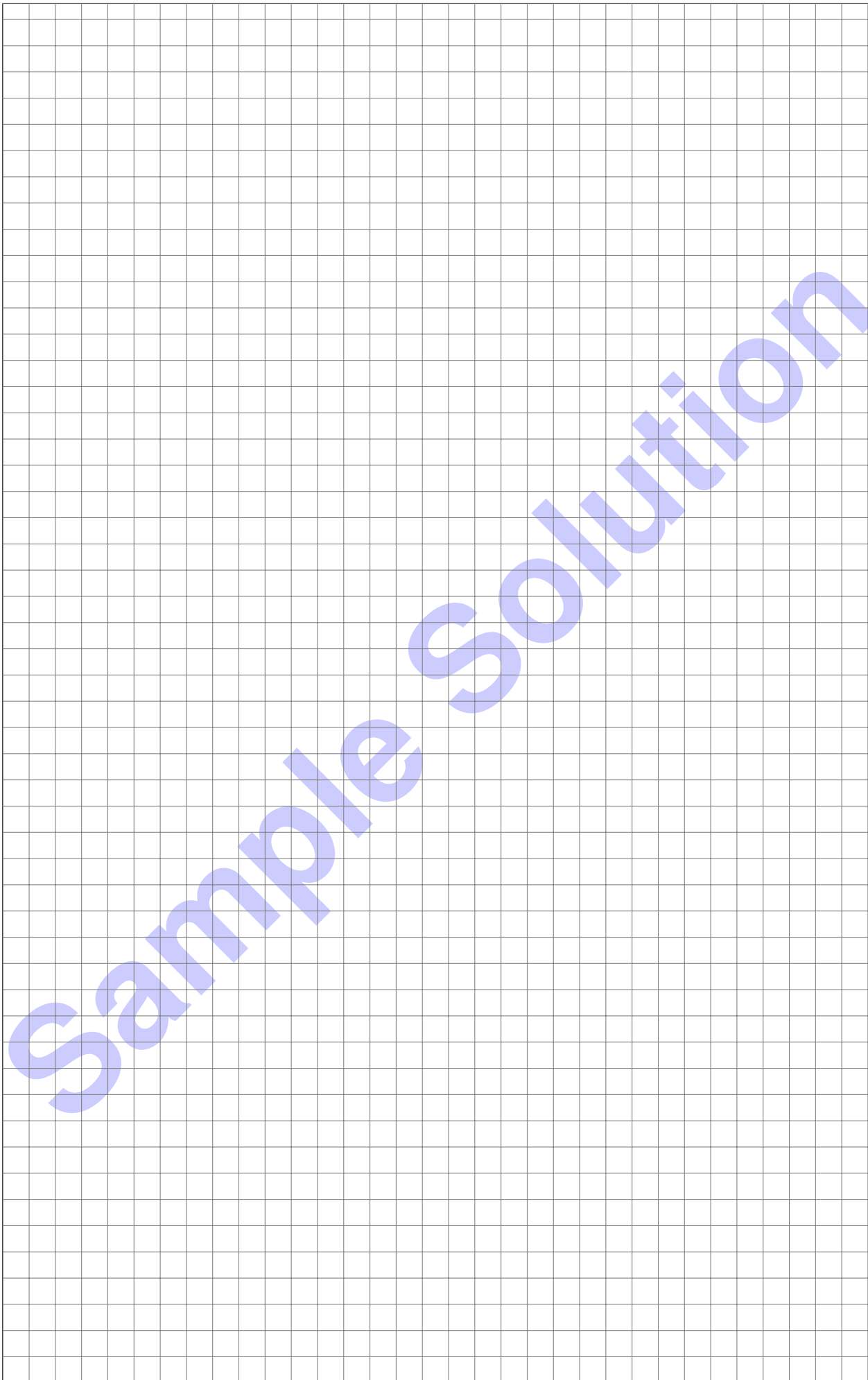


Additional space for solutions—clearly mark the (sub)problem your answers are related to and strike out invalid solutions.





Sample Solution



Sample Solution