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

Note:

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

Advanced Computer Networks

Module: IN2097 **Date:** 8.4.2016
Examiner: Prof. Dr.-Ing. Georg Carle **Exam:** Final exam

	P 1	P 2	P 3	P 4	P 5
First correction					
Second correction					

Left room from ____________ to ________________

from ____________ to ________________

Early submission at ________________

Notes ____

Final exam

Advanced Computer Networks

Prof. Dr.-Ing. Georg Carle
Chair of Network Architectures and Services
Department of Informatics
Technical University of Munich

Friday, 8.4.2016
8:30 – 9:30

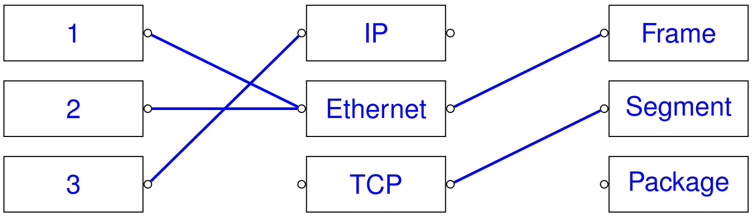
- This exam consists of
  - **16 pages** with a total of **5 problems** and
  - a two-sided printed **cheat sheet**.

  Please make sure now that you received a complete copy of the exam.
- 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.
- Allowed resources:
  - a **non-annotated, printed vocabulary** English ↔ native language.
- Physically turn off all electronic devices, put them into your bag and close the bag.

Problem 1 Quiz (11 credits)

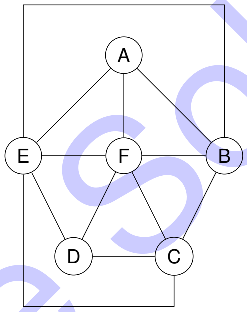
The following questions cover multiple topics and can be solved independently of each other.

a)* Connect the protocols to their corresponding ISO/OSI layer as well as to their corresponding term for their PDU. **Hint:** There may be terms where no/several pairings are possible.



b)* Add a single edge to the network in Figure 1b) to increase the maximum value of *k* when performing the *k*-core algorithm.

Add an edge between A and D or between B and D.



c)* Given the IP address `61.149.21.16` and subnet mask `255.255.255.192`, determine the corresponding network and broadcast addresses.

Netmask `255.255.255.192` corresponds to a prefix length of 26, leaving 6 bit for the host part, i. e., the subnet has a total of 64 addresses. `61.149.21.16` thus belongs to the first subnet starting at `61.149.21.0` (network address) and ending at `61.149.21.63` (broadcast address).

d)* Longest prefix matching is an algorithm commonly used in computer networks. Where is it used specifically and how does it work?

- Used in routers for determining the best next-hop for a given packet.
- Bitwise AND between the packet's destination address and subnet mask of each entry in a router's routing table, starting at the longest (most specific prefix). If the result matches the corresponding route's network address, the best match is found.

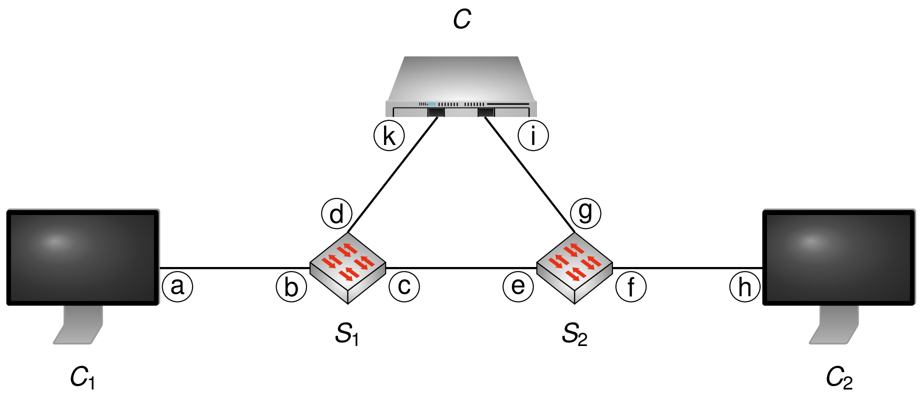


Figure 1.1: Software defined network topology

Consider the software defined network in Figure 1.1. The network consists of two clients C_1 and C_2 and an OpenFlow controller C. In between the clients two switches S_1 and S_2 are installed. Each of the switches is connected to a single client, to the respective other switch and to the controller.
There are two flow tables already installed on the switches:

Match fields	Action
`dl_type = 0x0800, nw_proto = 0x1, nw_src =` $C_1\_src\_ip$	`[port_c]`

Table 1.1: Flow table of S_1

Match fields	Action
`dl_type = 0x0800, nw_proto = 0x6, nw_src =` $C_1\_src\_ip$	`[port_e]`

Table 1.2: Flow table of S_2

The standard action for a packet if no rule matches is to transmit it to the controller. The default action of the controller is to send incoming packets back to the switch and to instruct it to forward the packet to all interfaces except the original source interface and the management port.
Explanation for match fields:

- `dl_type`: used type of payload for the data link layer protocol
(**Hint:** see cheat sheet for values, section Ethernet)
- `nw_proto`: used type of payload for the network layer protocol
(**Hint:** see cheat sheet for values, section IPv4)
- `nw_src`: source address of network protocol

e)* C_1 pings C_2. List all interfaces an ICMP echo request packet travels through the topology in the correct order (only the request packet, not the answer packet).

	i	ii
0	□	□
1	□	□
2	□	□
3	□	□

a, b, c, e, g, i, (i), g, f, h

	i	ii
0	□	□
	□	□
1	□	□
	□	□
2	□	□

f)* Consider a switch based on off-the-shelf hardware such as Open vSwitch and a dedicated SDN switch based on specialized hardware. List two advantages for each of the switches.

Benefits of off-the-shelf hardware switches

- Hardware costs are lower
- Standard hardware is available from different vendors so one is less dependent on a single manufacturer
- Larger memory

Benefits of hardware switches

- Higher port density
- Higher bandwidth
- Lower latency

Problem 2 Receive Side Scaling (9 credits)

Receive Side Scaling (RSS) is a feature to distribute the network traffic to different hardware queues assigned to different CPU cores. Figure 2.1 presents the sequence performed during packet reception on an RSS system.

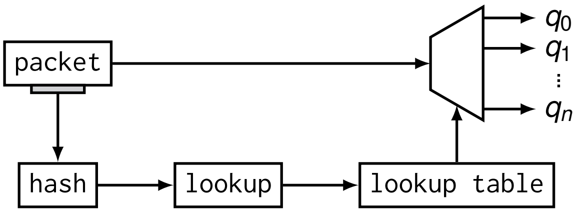


Figure 2.1: Receive Side Scaling

a)* Why is the support for multiple hardware queues beneficial for distributing CPU load to different cores instead of using a single queue.

- A single queue is used only by a dedicated core
- Therefore, no synchronization between different consumers of a queue is necessary

The following calculations use the *XOR* (exclusive or) and *MOD* (modulo) operators. The hash function is applied on a given IPv4 address `A.B.C.D`:

$$hash(IPv4_{src\_addr}) =$$
$$hash(A.B.C.D) = A\ XOR\ B\ XOR\ C\ XOR\ D$$

After calculating the hash a lookup in a hardware lookup table is performed. This determines the hardware queue where packets are enqueued. CPU cores assigned to a queue process the packets afterwards. The lookup is calculated as follows:

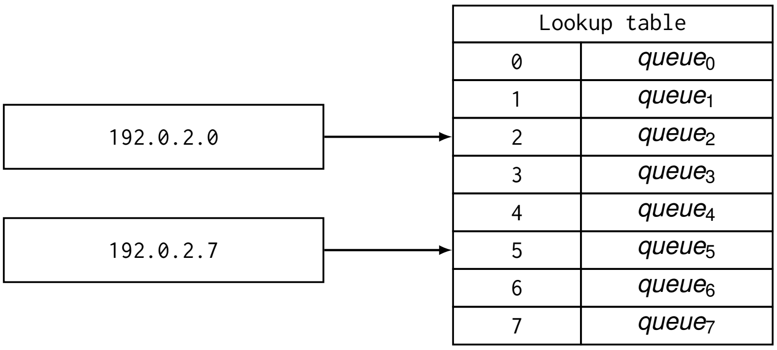
$$lookup(hash(IPv4_{src\_addr})) = hash(IPv4_{src\_addr}) \quad \bmod 2^n$$

For this problem: $n = 3$.

b)* Perform a hash and a lookup operation on the IP address `192.0.2.1`.

IP to hex: `192.0.2.1` = `0xC0000201`
Hash & Lookup: $(0xC0\ XOR\ 0x00\ XOR\ 0x02\ XOR\ 0x01) \mod 2^3 = 0xC3 \mod 8 = 3$

c) Determine the IP addresses of the `192.0.2.0 / 29` subnet for the following mappings.



Lookup table	
0	$queue_0$
1	$queue_1$
2	$queue_2$
3	$queue_3$
4	$queue_4$
5	$queue_5$
6	$queue_6$
7	$queue_7$

The lookup table has a fixed size of 8. All cells of the lookup table must be filled at all times. As the cores should have a similar utilization, the incoming packets should be distributed in a uniform manner among the cores. You can assume that the source IP addresses are uniformly distributed which leads to a uniform distribution for hash and lookup function alike.

d)* Give the content of a lookup table if three queues/cores are in use. Try to approximate the uniform distribution as closely as possible.

MOD result	0	1	2	3	4	5	6	7
Core number	0	0	0	1	1	1	2	2

e) Calculate the relative probability of packet distribution for each core in percent.

$P_{0,1} = \frac{3}{8} = 37.5\%$
$P_2 = \frac{2}{8} = 25\%$

f) What would be the ideal value and how big is the maximum deviation.

$P_{ideal} = 33.33\%$
$P_{max\_dev} = P_{ideal} - P_2 = 33.33\% - 25\% = 8.33\%$

g)* Depending on n, how many CPU cores/queues should be used if the distribution should be met as accurately as possible?

2^i for $i = 1...n$

Problem 3 BGP (7 credits)

Figure 3.1 shows a small AS topology including border routers R1, R2 and R3. AS20 and AS77 are costumers of AS10. AS20 and AS77 have a peering agreement, i. e., they exchange traffic for free. AS10 and AS77 own prefixes that are announced to their customers/peering partners. AS77 owns the prefix `191.168.0.0/20`, AS 10 owns the prefix `151.17.0.0./16`.

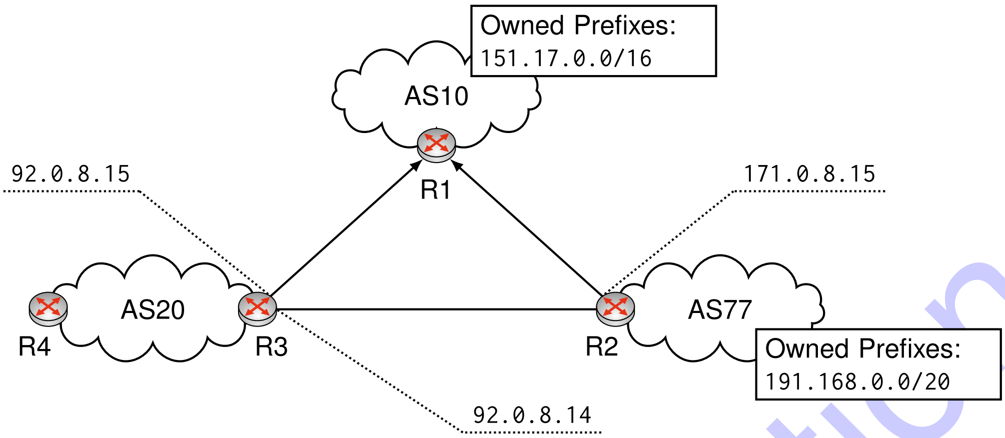


Figure 3.1: AS topology

a)* There exist two “flavors” of BGP. Which ones are used between the listed routers?

- R4, R3: iBGP
- R1, R3: eBGP

b)* Create the routing table entry/entries for the border router of A10 (R1) for the owned prefixes of AS77.

Network destination	Subnetmask	Next hop IP address
191.168.0.0	255.255.240.0	171.0.8.15

An attacker controlling AS20 wants to sniff the traffic that is exchanged between AS10 and AS77 for their respective prefixes `151.17.0.0/16` and `191.168.0.0/20`. Therefore, this traffic shall be routed through AS20, i.e., AS20 performs a man-in-the-middle attack.

	i	ii
0	□	□
1	□	□
2	□	□

c)* Describe what the attacker has to do to perform the man-in-the-middle attack.

- The attacker AS20 has to announce more specific entries to AS10 to get the traffic from AS10 to AS77.
- The attacker AS20 has to announce a route for prefix 151.17.70.0 to AS77. AS77 will prefer this route over the route to its provider AS10 because it is cheaper for AS77.

	i	ii
0	□	□
1	□	□
2	□	□
3	□	□

d) List the new routing table entry/entries for the border router of AS10 (R1) after a successful man-in-the-middle attack of AS20 with as little entries as possible.

Network destination	Subnetmask	Next hop IP address
191.168.0.0	255.255.248.0	92.0.8.15
191.168.8.0	255.255.248.0	92.0.8.15

Problem 4 Wireshark (14 credits)

We consider the IP packet depicted in Figure 4.1 as hexdump in network byte order.

```
 IHL   TOS   Total Length  Identification  Flags/FragOffset  TTL
 45 |  c0 |  00  5c  |  d4  f6  |  00  00       |  3f |  01  6e  6c  C0  00  02  00

 0A    00    00  02    03  03    34  ae            00    00  00  00
```

Figure 4.1: Partial hexdump of an IP packet in network byte order

The topology of the network for this problem is given in Figure 4.2. A client *C* wants to connect from its local network to a server *S* on the Internet via the NAT router *R*.

Note: To solve this problem use the cheat sheet that is handed out separately.

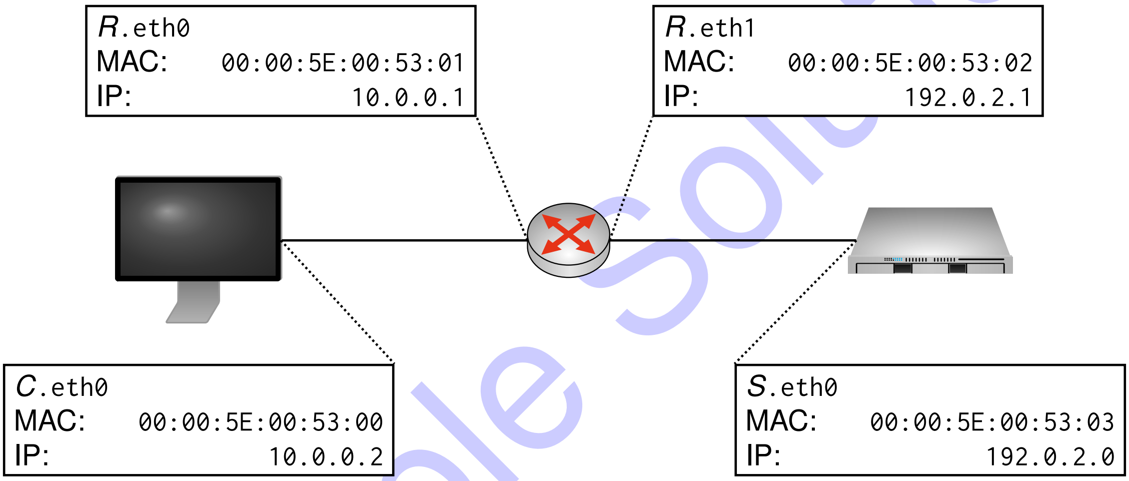


Figure 4.2: The network topology in which the packet was recorded

a)* Mark and name the header fields in Figure 4.1.

b)* Argue from which interface to which interface the packet (see Figure 4.1) was transferred.

The packet was transferred from *R*.eth0 to *C*.eth0 , because the source IP address is *S*.eth0 and the destination IP address is already translated to the private address 10.0.0.2.

c)* Argue what protocol is contained in the payload of the IP packet in Figure 4.1.

The protocol field of the IP packet contains, 0x01 so a ICMP message is transferred.

The IP packet in Figure 4.1 contains an incomplete ICMP message starting at offset `0x0014`.

d)* Argue which kind of ICMP message it is and why this message could be generated.

The type and the code field of the ICMP message are `0x03`, i.e. a destination unreachable message with a destination port unreachable code field. The message indicates an error that a specific port of the destination could not be reached. For instance the application may be not running and the port is closed or port could be blocked by a firewall.

This ICMP message was received after client *C* tried to open an **SSH connection** to *S*. With this information it is possible to recreate the missing payload of the ICMP message.

e) Name the protocol(s) contained in the payload of the ICMP message.

IP (header) + TCP (header but only first 8 byte)

f)* Create a **hexdump** of the missing payload for the ICMP message of Figure 4.1.
Hint: The calculation of correct checksums is not necessary, fill in `0xFF` blocks if needed. Not all listed headers might be used for SSH.

- IP Version: `4`
- TTL: Values between `0x01` and `0xFE`
- Protocol: TCP (`0x06`)
- Header checksum: `0xFF 0xFF`
- Source IP: `0x0A 0x00 0x00 0x02` also `0xC0 0x00 0x02 0x01` possible for some NATs.
- Destination IP: `0xC0 0x00 0x02 0x00`
- Destination Port: `0x16` (SSH)
- Source Port: all ports as long as higher than 1024
- Sequence Number: `0x00 0x01`

g)* Generate a hexdump of the Ethernet frame for the IP packet given in Figure 4.1. The entire payload of the Ethernet packet may be abbreviated with '. . .'.
Hint: The calculation of correct checksums is not necessary, fill in `0xFF` blocks if needed.

```
00 00 5E 00 53 00 | 00 00 5E 00 53 01 | 08 00 ... FF FF FF FF
```

	i	ii
0	□	□
	□	□
1	□	□
	□	□
2	□	□

Problem 5 TCP (19 credits)

TCP has the ability to detect packet loss via timeouts. This timeout value, called RTO, must be adopted to the properties of a TCP connection to work properly.

a)* What happens to the detection of packet loss when the RTT of a TCP connection is estimated too high?

Losses are detected too late, which leads to unnecessarily longer connection times.

b)* What happens to the detection of packet loss when the RTT of a TCP connection is estimated too low?

Packets may be wrongly assumed as lost , i.e. packets may be retransmitted unnecessarily.

For a network experiment the connections S_1 to C and S_2 to C over router R as shown in Figure 5.1 are tested. During the experiment different buffer configurations for R shall be tested. The bandwidth of each depicted link is the same.

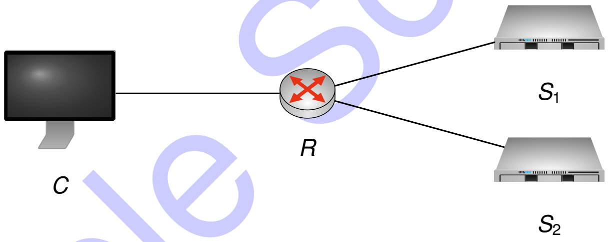


Figure 5.1: Network topology

Several active TCP connections try to use the full bandwidth available between S_1 and C and between S_2 and C. Additionally a ping from C to S_1 is executed. The average RTT measured by ping for a large buffer configuration and a small buffer configuration is plotted in Figure 5.2.

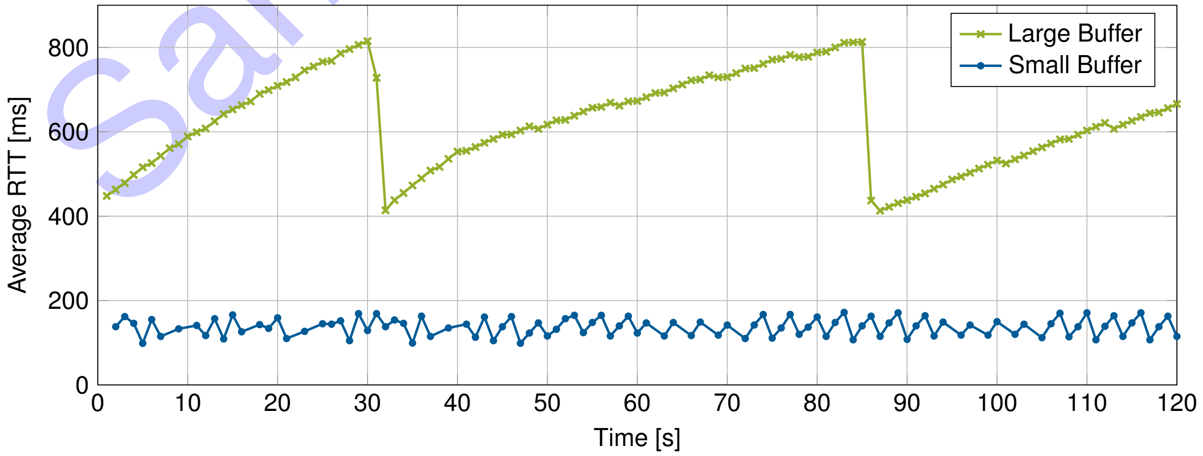


Figure 5.2: Latency with different buffer sizes

c) What is the influence of different buffer sizes on the accuracy of the RTO estimation. Argue with the results for the two buffer sizes from Figure 5.2.

	i	ii
0	□	□
	□	□
1	□	□
	□	□
2	□	□
	□	□
3	□	□

- The latency in Figure 5.2 ranges from 400ms to 800ms for the large buffer size.
- The latency in Figure 5.2 ranges from 100ms to 150ms for the small buffer size.
- Therefore the jitter is higher for large buffers which makes it harder to guess the RTO accurately.

d)* What happens to latency critical applications such as VoIP in the presence of large buffers?

	i	ii
0	□	□
	□	□
1	□	□

- Latency critical applications will not work properly in this situation because the large buffer introduce additional latency

e) How can the situation for latency critical applications be improved, without changing the buffer size?

	i	ii
0	□	□
	□	□
1	□	□

- Situation can be improved by prioritizing VoIP traffic (QoS) i.e. high priority packets leave the buffer earlier than other low priority packets .

To improve the behavior of TCP with large buffers an active queue management is introduced.

f)* Name and explain two different discard policies.

	i	ii
0	□	□
	□	□
1	□	□
	□	□
2	□	□

- Tail drop: drop arriving packet
- Priority drop: drop on priority basis
- Random drop: drop random packet

The algorithm applied is weighted fair queuing. For that three classes of traffic shall be considered A, B and C with their corresponding weights $w_A = 2$, $w_B = 4$, $w_C = 3$. The weight of the traffic classes gives the maximum burst size which is allowed for a certain traffic class. The algorithm starts to send a burst of A traffic, after the burst size is exhausted or the queue for this traffic class is empty, the same is done for the traffic class B and afterwards for traffic class C. At each timeslot an arbitrary number of packets can be received but only a single packet can be sent.

g)* Calculate the overall possible bandwidth capacity for each traffic class in percent. Assume that enough packets for all three classes are available to always fill the respecive bursts.

$$\frac{w_i}{\sum_n w_n}$$

$w_A = \frac{2}{9} = 22.2\%$

$w_C = \frac{3}{9} = 33.3\%$

$w_B = \frac{4}{9} = 44.4\%$

For the next problems consider the following incoming packets:

Timeslot	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
A	1	0	0	1	1	0	0	0	0	0	0	0
B	1	1	0	1	1	0	0	0	0	0	0	0
C	1	2	2	0	0	0	0	0	0	0	0	0

h) Compute which packet is sent out at each time slot. Additionally, give the content of the queues at this time. **Hint:** Timeslot 1 is already computed, proceed accordingly.

Timeslot	1	2	3	4	5	6	7	8	9	10	11	12
Queue A	0	0	0	1	2	2	2	2	1	0	0	0
Queue B	1	1	0	0	0	0	0	0	0	0	0	0
Queue C	1	3	5	5	5	4	3	2	2	2	1	0
Sent	A	B	B	B	B	C	C	C	A	A	C	C

In the following a token bucket approach for QoS shall be applied. The initial tokens are $t_A = 0$, $t_B = 2$, and $t_C = 2$. An additional token is generated for the queue A at every fourth, queue B at every second, and queue C at every third timeslot. Token handling/checking is done before making the decision for sending.

i)* Compute which packet is sent out at each time slot. Additionally, give the content of the queues and the available tokens for the respective traffic class at this time. **Hint:** Timeslot 1 is already computed, proceed accordingly.

Timeslot	1	2	3	4	5	6	7	8	9	10	11	12
Queue A	1	1	1	2	3	3	2	1	1	1	1	0
Token A	0	0	0	1	1	1	0	0	0	0	0	0
Queue B	0	0	0	1	2	2	2	2	1	0	0	0
Token B	1	1	1	2	2	3	3	4	3	3	3	4
Queue C	1	3	4	3	2	1	1	1	1	1	0	0
Token C	2	2	2	1	0	0	0	0	1	1	0	1
Sent	B	B	C	C	C	C	A	A	B	B	C	A

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