Solution of Exercise Sheet 4

Exercise 1 (Routers, Layer-3-Switches, Gateways)

1. What is the purpose of **Routers** in computer networks?
   (*Also explain the difference to Layer-3-Switches.*)

   They forward packets between networks with different logical address ranges and provide a WAN interface.

2. What is the purpose of **Layer-3-Switches** in computer networks?
   (*Also explain the difference to Routers.*)

   They are Routers too which means they forward packets between networks with different logical address ranges but they do not provide a WAN interface.

3. What is the purpose of **Gateways** in computer networks?

   They enable communication between networks, which base on different protocols.

4. Why are **Gateways** in the network layer of computer networks seldom required nowadays?

   Modern computer networks operate almost exclusively with the Internet Protocol (IP). For this reason, a protocol conversion at the Network Layer is mostly not required.
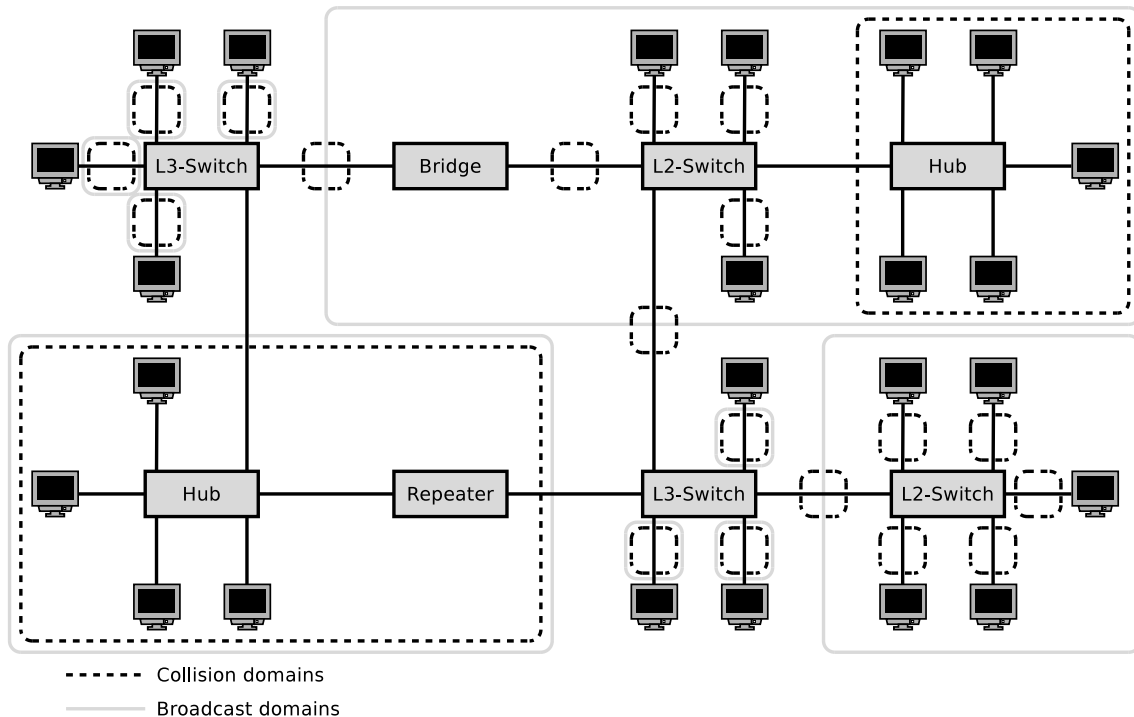
Exercise 2 (Collision Domain, Broadcast Domain)

1. Which devices **divide the collision domain**?

   ☐ Repeater   ☒ Bridge   ☒ Router
   ☐ Hub   ☒ Layer-2-Switch   ☒ Layer-3-Switch

2. Which devices **divide the broadcast domain**?

   ☐ Repeater   ☐ Bridge   ☒ Router
   ☐ Hub   ☐ Layer-2-Switch   ☒ Layer-3-Switch

3. Sketch in the diagram of the network topology all **collision domains** and all **broadcast domains**.

Exercise 3   (Addressing in the Network Layer)

1. What is the meaning of **Unicast** in the network layer of computer networks?

   An IP address is assigned to a single receiver.

2. What is the meaning of **Broadcast** in the network layer of computer networks?

   An IP address is assigned to all receivers in the subnet.

3. What is the meaning of **Anycast** in the network layer of computer networks?

   An IP address is used to reach a single device of a group of devices.

4. What is the meaning of **Multicast** in the network layer of computer networks?

   An IP address is assigned to a group of receivers.

5. Why contains the IPv4 **address space** only 4,294,967,296 addresses?

   IPv4 addresses have a length of 32 bits (4 bytes). Thus, the address space contains $2^{32} = 4,294,967,296$ possible addresses.

6. Why was **Classless Interdomain Routing (CIDR)** introduced?

Because with address classes, many addresses are wasted and it is impossible to dynamically adjust address classes.

7. Describe in simple words the **functioning of CIDR**.
   Focus on the way, IP addresses are treated and subnets are created.

   Since the introduction of CIDR, the address class of an IPv4 address is no longer important. All hosts in a network have a subnet mask assigned, which has a length of 32 bits (4 bytes) and is used to specify the number of subnets and hosts. The network mask splits the host ID of an IP address into subnet ID and host ID. 1-bits in the subnet mask indicate, which part of the address space is used for subnet IDs and 0-bits indicate, which part of the address space is used for host IDs.

Exercise 4   (Addressing in the Network Layer)

Calculate for each subtask of this exercise the **first and last host addresses**, the **network address** and the **broadcast address** of the subnet.

```
IP Address:        151.175.31.100    10010111.10101111.00011111.01100100
Subnet mask:       255.255.254.0     11111111.11111111.11111110.00000000
Part for host IDs:                                            x xxxxxxx
Network address?   151.175.30.0      10010111.10101111.00011110.00000000
First host address? 151.175.30.1     10010111.10101111.00011110.00000001
Last host address? 151.175.31.254    10010111.10101111.00011111.11111110
Broadcast address? 151.175.31.255    10010111.10101111.00011111.11111111


IP Address:        151.175.31.100    10010111.10101111.00011111.01100100
Subnet mask:       255.255.255.240   11111111.11111111.11111111.11110000
Part for host IDs:                                                 xxxx
Network address?   151.175.31.96     10010111.10101111.00011111.01100000
First host address? 151.175.31.97    10010111.10101111.00011111.01100001
Last host address? 151.175.31.110    10010111.10101111.00011111.01101110
Broadcast address? 151.175.31.111    10010111.10101111.00011111.01101111


IP Address:        151.175.31.100    10010111.10101111.00011111.01100100
Subnet mask:       255.255.255.128   11111111.11111111.11111111.10000000
Part for host IDs:                                              xxxxxxx
Network address?   151.175.31.0      10010111.10101111.00011111.00000000
First host address? 151.175.31.1     10010111.10101111.00011111.00000001
Last host address? 151.175.31.126    10010111.10101111.00011111.01111110
Broadcast address? 151.175.31.127    10010111.10101111.00011111.01111111
```

binary representation	decimal representation	binary representation	decimal representation
10000000	128	11111000	248
11000000	192	11111100	252
11100000	224	11111110	254
11110000	240	11111111	255

Exercise 5 (Addressing in the Network Layer)

In each subtask of this exercise, a sender transmits an IP packet to a receiver. Calculate for each subtask the **subnet ID of sender and receiver** and specify whether the IP packet **leaves the subnet during transmission** or not.

```
Sender:      11001001.00010100.11011110.00001101    201.20.222.13
Subnet mask: 11111111.11111111.11111111.11110000    255.255.255.240
        AND ----------------------------------
             11001001.00010100.11011110.00000000    => Subnet ID = 0


Receiver:    11001001.00010100.11011110.00010001    201.20.222.17
Subnet mask: 11111111.11111111.11111111.11110000    255.255.255.240
        AND ----------------------------------
             11001001.00010100.11011110.00010000    => Subnet ID = 1
```

        Subnet ID of sender? 0

        Subnet ID of receiver? 1

        Does the IP packet leave the subnet [yes/no]? yes

```
Sender:      00001111.11001000.01100011.00010111    15.200.99.23
Subnet mask: 11111111.11000000.00000000.00000000    255.192.0.0
        AND ----------------------------------
             00001111.11000000.00000000.00000000    => Subnet ID = 3



Receiver:    00001111.11101111.00000001.00000001    15.239.1.1
Subnet mask: 11111111.11000000.00000000.00000000    255.192.0.0
        AND ----------------------------------
             00001111.11000000.00000000.00000000    => Subnet ID = 3
```

        Subnet ID of sender? 3

        Subnet ID of receiver? 3

```
            Does the IP packet leave the subnet [yes/no]? no
```

Exercise 6   (Addressing in the Network Layer)

Calculate for each subtask of this exercise the **subnet masks** and answer the **questions**.

1. Split the class C network `195.1.31.0` for implementing 30 subnets.

```
Network ID:    11000011.00000001.00011111.00000000    195.1.31.0
Number of bits for subnet IDs? 30 => 32 (= 2^5) => 5 bits
Subnet mask:   11111111.11111111.11111111.11111000    255.255.255.248
Number of bits for host IDs? 3
Number of host IDs per subnet? 2^3 − 2 = 6
```

2. Split the class A network `15.0.0.0` for implementing 333 subnets.

```
Network ID:    00001111.00000000.00000000.00000000    15.0.0.0
Number of bits for subnet IDs? 333 => 512 (= 2^9) => 9 bits
Subnet mask:   11111111.11111111.10000000.00000000    255.255.128.0
Number of bits for host IDs? 15
Number of host IDs per subnet? 2^15 − 2 = 32766
```

3. Split the class B network `189.23.0.0` for implementing 20 subnets.

```
Network ID:    10111101.00010111.00000000.00000000    189.23.0.0
Number of bits for subnet IDs? 20 => 32 (= 2^5) => 5 bits
Subnet mask:   11111111.11111111.11111000.00000000    255.255.248.0
Number of bits for host IDs? 11
Number of host IDs per subnet? 2^11 − 2 = 2046
```

4. Split the class C network `195.3.128.0` into subnets, which contain 17 hosts each.

```
Network ID:    11000011.00000011.10000000.00000000    195.3.128.0
Number of bits for host IDs? 17 => 32 (= 2^5) => 5 bits
Number of bits for subnet IDs? 8 − 5 = 3 bit
Number of possible subnets? 2^3 = 8
Subnet mask:   11111111.11111111.11111111.11100000    255.255.255.224
```

5. Split the class B network `129.15.0.0` into subnets, which contain 10 hosts each.

```
Network ID:      10000001.00001111.00000000.00000000      129.15.0.0
Number of bits for host IDs? 10 => 16 (= 2^4) => 4 bits
Number of bits for subnet IDs? 16 − 4 = 12 bit
Number of possible subnets? 2^12 = 4096
Subnet mask:     11111111.11111111.11111111.11110000      255.255.255.240
```

binary representation	decimal representation	binary representation	decimal representation
10000000	128	11111000	248
11000000	192	11111100	252
11100000	224	11111110	254
11110000	240	11111111	255

Exercise 7   (Collision   Domain,   Broadcast   Domain)

1. Sketch in the diagram of the network topology all **collision domains** and all **broadcast domains**.
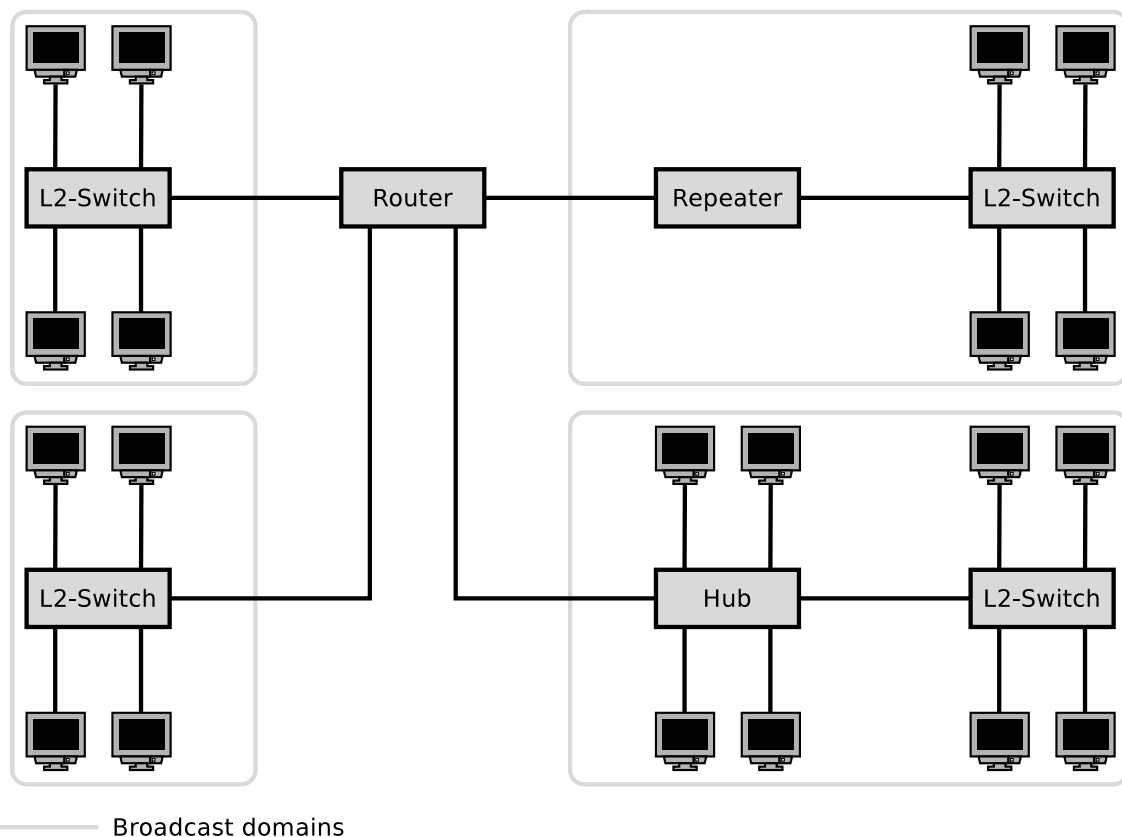

2. Sketch in the diagram of the network topology all **collision domains** and all **broadcast domains**.
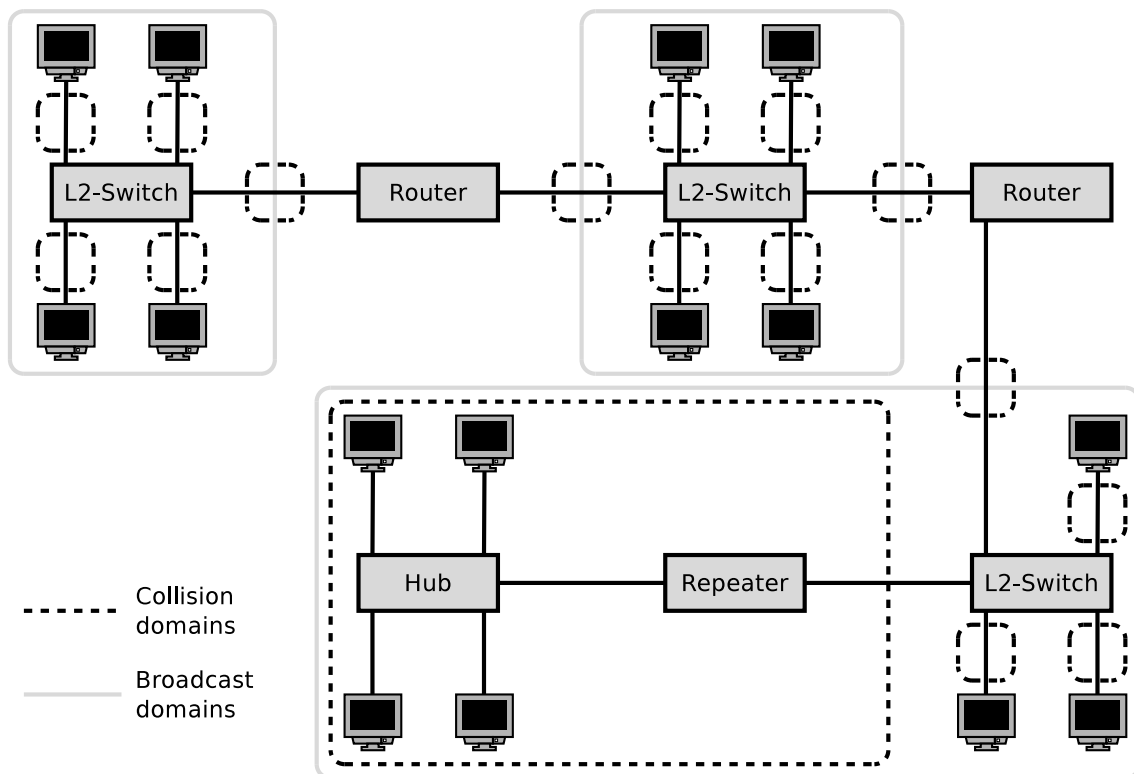
- - - - - - Collision domains
──────── Broadcast domains
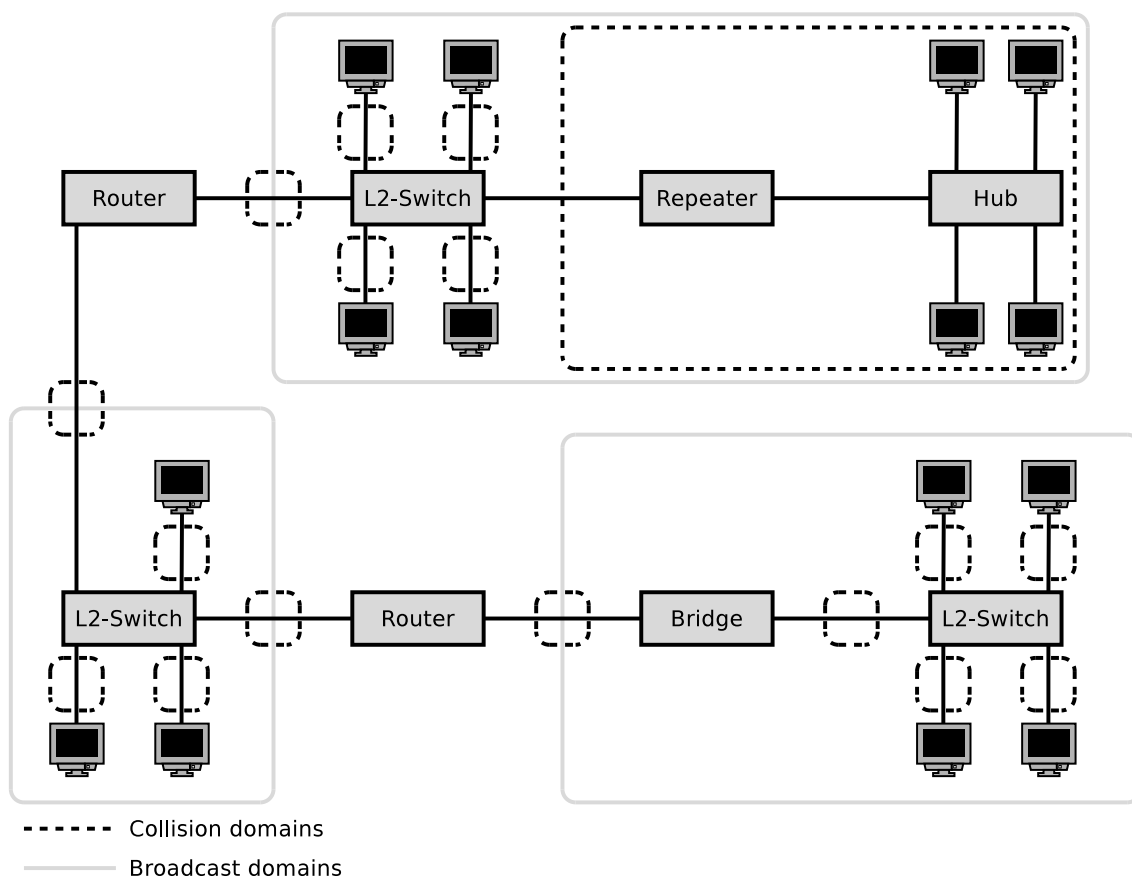
Exercise 8   (Broadcast Domain)

1. Sketch in the diagram of the network topology all **broadcast domains**.

2. What is the **required number of subnets** for this network topology?

Broadcast domains

4 subnets are required because each port of a Router is connected to a different IP network. It is impossible to operate an IP subnet on multiple ports of a Router.

Exercise 9   (Private IP Address Spaces)

Name the three private IPv4 address spaces.

- 10.0.0.0/8
- 172.16.0.0/12
- 192.168.0.0/16

Exercise 10   (Addressing in the Network Layer)

Calculate for each network configuration in the table whether an IP packet, which is send from the given IP address to the destination address, **leaves the subnet during transmission** or not.

economize

economize

```
    11010010.00000101.00010000.11000110      210.5.16.198
AND 11111111.11111111.11111111.11111100      255.255.255.252
----------------------------------------
    11010010.00000101.00010000.11000100
                              ^^^^^^           => 49 = subnet ID sender


    11010010.00000101.00010000.11000101      210.5.16.197
AND 11111111.11111111.11111111.11111100      255.255.255.252
----------------------------------------
    11010010.00000101.00010000.11000101
                              ^^^^^^           => 49 = subnet ID sender
```

Sender and receiver have equal subnet IDs $\implies$ the subnet is not left.

```
    11010010.00000101.00010000.11000110      210.5.16.198
AND 11111111.11111111.11111111.11111100      255.255.255.252
----------------------------------------
    11010010.00000101.00010000.11000100
                              ^^^^^^           => 49 = subnet ID sender


    11010010.00000101.00010000.11001001      210.5.16.201
AND 11111111.11111111.11111111.11111100      255.255.255.252
----------------------------------------
    11010010.00000101.00010000.11001000
                              ^^^^^^           => 50 = subnet ID receiver
```

Sender and receiver have different subnet IDs $\implies$ the subnet is left.

```
    00000101.00000101.00000101.00000101      5.5.5.5
AND 11111111.11111110.00000000.00000000      255.254.0.0
----------------------------------------
    00000101.00000100.00000000.00000000
            ^^^^^^^^                          => 2 = subnet ID sender


    00000101.00000110.00000110.00000110      5.6.6.6
AND 11111111.11111110.00000000.00000000      255.254.0.0
----------------------------------------
    00000101.00000110.00000000.00000000
            ^^^^^^^^                          => 3 = subnet ID sender
```

Sender and receiver have different subnet IDs $\implies$ the subnet is left.

Exercise 11   (Addressing in the Network Layer)

Specify for each subtask of this exercise the correct **subnet mask**.

1. A maximum number of subnets with 5 hosts each in a class B network.

   5 hosts $\Longrightarrow$ 3 bit are required.
   11111111 11111111 11111111 11111000   *255.255.255.248*

2. 50 subnets with 999 hosts each in a class B network.

   999 hosts $\Longrightarrow$ 10 bit are required.
   11111111 11111111 11111100 00000000   *255.255.252.0*

3. 12 subnets with 12 hosts each in a class C network.

   12 hosts $\Longrightarrow$ 4 bit are required.
   11111111 11111111 11111111 11110000   *255.255.255.240*

Source: Jörg Roth. Prüfungstrainer Rechnernetze. Vieweg (2010)

Exercise 12   (Checksums in IP Packets)

The figure shows the structure of IPv4 packets as discussed in the computer networks course.

32 bits (4 bytes)

Version	IHL	Differentiated services	Total length		
Identification			Flags	Fragment offset	
Time To Live		Protocol ID	Header checksum		
IP Address (sender)					
IP Address (destination)					
Options / Padding					
Payload (data from the transport layer)					
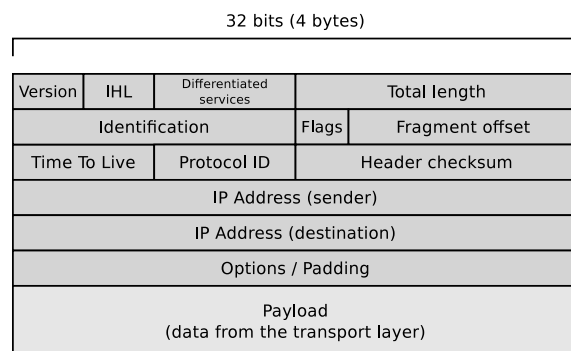
The given data in hexadecimal notation is a truncated excerpt of an IP packet:

4500 0034 B612 4000 4006 6F80 0A00 008B 5BC6 AEE0

The data contains the values of the fields of the IP packet header.

4	= Version
5	= IHL = IP Header Length ($\Longrightarrow 5 * 4$ Byte words = 20 bytes)
00	= Differentiated services
0034	= Total length ($\Longrightarrow$ 52 bytes)
B612	= Identification
4000	= Flags + Fragment offset
40	= Time To live ($\Longrightarrow$ 62 hops)

```
  06        = Protocol ID (⟹ TCP)
6F80        = Header Checksum
0A00 008B = IP address (sender)
5BC6 AEE0 = IP address (destination)
```

Example: Calculate checksum

RFC 791 says on page 14: „*The checksum field is the 16 bit one's complement of the one's complement sum of all 16 bit words in the header*".

To calculate the checksum of the packet, the sum of each 2 byte word inside the header must be calculated. The checksum field itself is skipped here!

```
4500 + 0034 + B612 + 4000 + 4006 + 0A00 + 008B + 5BC6 + AEE0 = 2907D
```

Next, the result of the calculation is converted to binary:

```
2907D ⟹ 10 1001 0000 0111 1101
```

The first two bits are the carry and need to be added to the rest of the value:

```
10 + 1001 0000 0111 1101 = 1001 0000 0111 1111
```

Next, every bit of the result is flipped to obtain the checksum:

```
     1001 0000 0111 1111
=> 0110 1111 1000 0000
```

The result `0110 1111 1000 0000` is equal to the value `6F80` in hexadecimal notation, as already shown in the original IP packet header.

Example: Verify checksum

RFC 791 says on page 14: „*For purposes of computing the checksum, the value of the checksum field is zero*".

To verify a checksum, the same procedure is used as above, with a single exception: The original header checksum is not omitted.

```
4500 + 0034 + B612 + 4000 + 4006 + 6F80 + 0A00 + 008B + 5BC6 + AEE0 = 2FFFD
```

Next, the result of the calculation is converted to binary:

```
2FFFD ⟹ 10 1111 1111 1111 1101
```

The first two bits are the carry and need to be added to the rest of the value:

```
10 + 1111 1111 1111 1101 = 1111 1111 1111 1111
```

Next, every bit of the result is flipped:

```
    1111 1111 1111 1111
=> 0000 0000 0000 0000
```

This indicates: No error detected! Any result, which is $\neq 0$ indicates: Error!

Source: RFC 791 and Wikipedia and
`http://mars.netanya.ac.il/~unesco/cdrom/booklet/HTML/NETWORKING/node020.html`

---

1. **Calculate** the checksum for each IP header:

   - `4500 0034 4C22 4000 F706 ???? C163 9055 0A00 008B`

     Attention: Always groups of 4 bits are required to flip!

     ```
     4500 + 0034 + 4C22 + 4000 + F706 + C163 + 9055 + 0A00 + 008B => 3249F
     3249F => 11 0010 0100 1001 1111
     The first two bits are the carry and need to be added to the rest of the value!
     11 + 0010 0100 1001 1111 = 0010 0100 1010 0010
     Flip: 0010 0100 1010 0010
           1101 1011 0101 1101 => DB5D
     ```

     $CRC =$ DB5D

   - `4500 0034 671E 4000 4006 ???? 0A00 008b C163 9055`

     ```
     4500 + 0034 + 671E + 4000 + 4006 + 0A00 + 008b + C163 + 9055 => 2889B
     2889B => 10 1000 1000 1001 1011
     The first two bits are the carry and need to be added to the rest of the value!
     10 + 1000 1000 1001 1011 = 1000 1000 1001 1101
     Flip: 1000 1000 1001 1101
           0111 0111 0110 0010 => 7762
     ```

     $CRC =$ 7762

   - `4500 00F2 0000 4000 4011 ???? 0A00 008b 0A00 00FF`

     ```
     4500 + 00F2 + 0000 + 4000 + 4011 + 0A00 + 008b + 0A00 + 00FF = DB8D
     DB8D => 1101 1011 1000 1101
     There is no carry!
     Flip: 1101 1011 1000 1101
           0010 0100 0111 0010 => 2472
     ```

     $CRC =$ 2472

2. **Verify** the checksum of each IP header:

   - `4500 0034 02FD 4000 3606 276C 6CA0 A330 0A00 008B`

     Correct.

   - `4500 00E7 02FC 4000 3606 37BC 6CA0 A330 0A00 008B`

     Error!

   - `4500 0034 A9D5 4000 4006 814E 0A00 008B ADC2 4613`

Error!

Exercise 13   (Fragmenting IP Packets)

4,000 bytes payload need to be transmitted via the IP protocol. The payload must be fragmented, because it is transmitted over multiple physical networks, whose MTU is $< 4,000$ bytes.
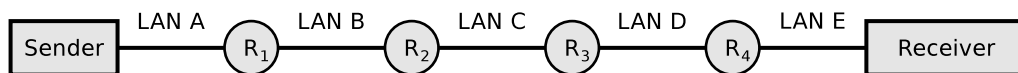
	LAN A	**LAN B**	**LAN C**	**LAN D**	**LAN E**
Network technology	Ethernet	PPPoE	ISDN	Ethernet	WLAN
MTU [bytes]	1,500	1,492	576	1,400	2,312
IP-Header [bytes]	20	20	20	20	20
maximum payload [bytes]	1,480	1,472	556	1,380	2,292

Display graphically the way, the payload is fragmented, and how many bytes of payload each fragment contains.
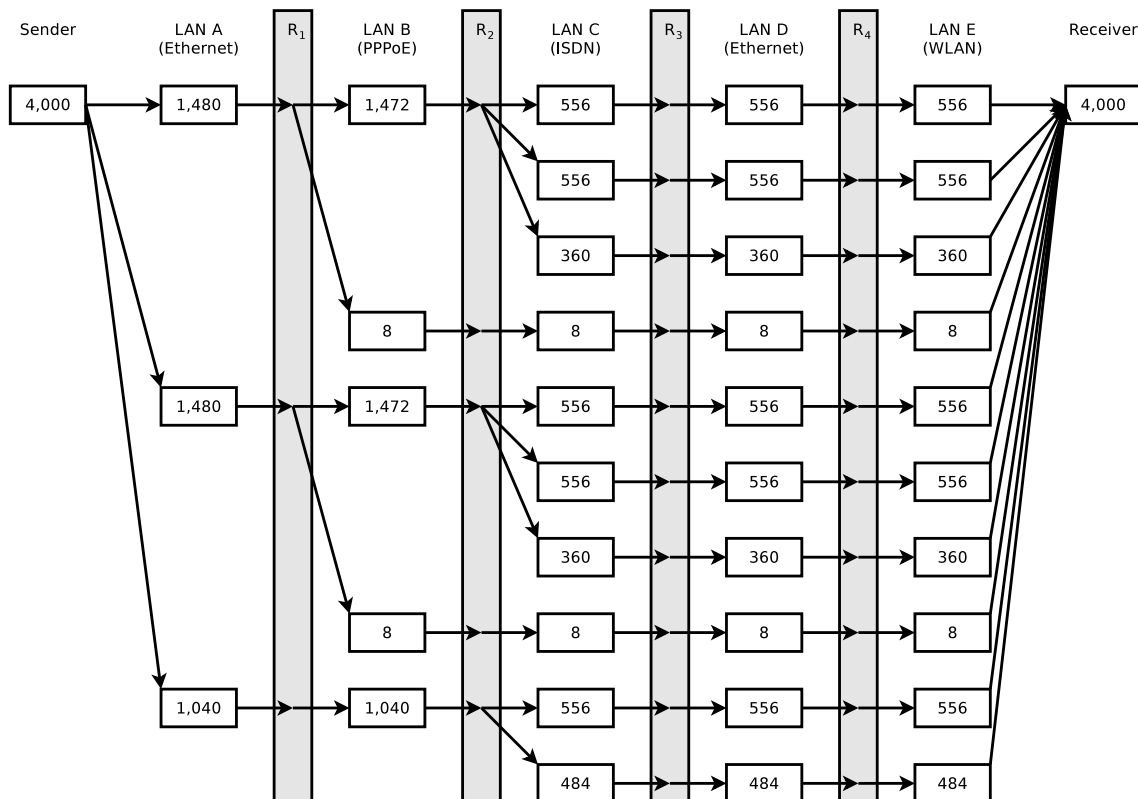
Exercise 14   (Forwarding and Path Calculation)

1. Which two major classes of **routing protocols** exist?

   Distance Vector Routing Protocols and Link State Routing Protocols.

2. Which **algorithms for best path calculation** implement the routing protocol classes from subtask 1?

   Distance Vector Routing Protocols implement the Bellman-Ford algorithm.

   Link State Routing Protocols implement the Dijkstra algorithm.

3. What is an **autonomous system**?

   Each AS consists of a group of logical networks, which use the Internet Protocol, are operated and managed by the same organization (e.g. an Internet Service Provider, a corporation or university) and use the same routing protocol.

4. The **Border Gateway Protocol** (BGP) is a protocol for. . .

   ☐ Intra-AS routing   ☒ Inter-AS routing

5. Which **routing protocol class** from subtask 1 implements the BGP?

   Link state routing.

6. **Open Shortest Path First** (OSPF) is a protocol for. . .

   ☒ Intra-AS routing   ☐ Inter-AS routing

7. Which **routing protocol class** from subtask 1 implements OSPF?

   Link state routing.

8. The **Routing Information Protocol** (RIP) is a protocol for. . .

   ☒ Intra-AS routing   ☐ Inter-AS routing

9. Which **routing protocol class** from subtask 1 implements the RIP?

   Distance vector routing.

10. When RIP is used, each Router communicates only with its **direct neighbors**. What are the **advantages** and **drawbacks** of method?

    Advantage: The network is not flooded $\implies$ protocol causes little overhead.

    Drawback: Long convergence time because updates propagate slowly.

11. When RIP is used, the path cost (metric) depend only on the number of Routers (**hops**), which need to be passed on the way to the destination network. What is the **drawback** of this method?

    The metric hop count often results in routes, which are not optimal, because all network segments have an equal weight.

12. When OSPF is used, **all Routers** communicate with each other. What are the **advantages** and **drawbacks** of method?

    Advantage: Short convergence time.

    Drawback: The network is flooded $\implies$ protocol causes strong overhead.
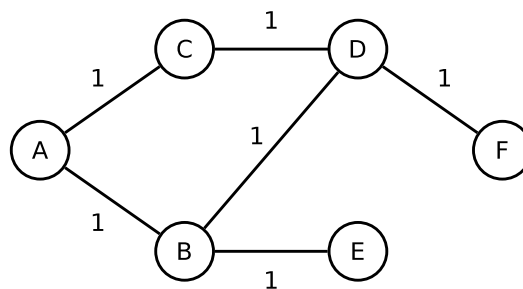
Exercise 15   (Bellman-Ford Algorithm)

1. **Calculate** the entries of the routing tables for each advertisement round of the Routing Information Protocol (RIP).
   (*The hop metric is used.*)

Step 1

Dest.	Hop	Metric
A	?	∞
B	?	∞
C	C	0
D	?	∞
E	?	∞
F	?	∞

Dest.	Hop	Metric
A	?	∞
B	?	∞
C	?	∞
D	D	0
E	?	∞
F	?	∞

Dest.	Hop	Metric
A	A	0
B	?	∞
C	?	∞
D	?	∞
E	?	∞
F	?	∞

Dest.	Hop	Metric
A	?	∞
B	?	∞
C	?	∞
D	?	∞
E	?	∞
F	F	0

Dest.	Hop	Metric
A	?	∞
B	B	0
C	?	∞
D	?	∞
E	?	∞
F	?	∞

Dest.	Hop	Metric
A	?	∞
B	?	∞
C	?	∞
D	?	∞
E	E	0
F	?	∞

Step 2

Dest.	Hop	Metric
A	A	1
B	?	∞
C	C	0
D	D	1
E	?	∞
F	?	∞

Dest.	Hop	Metric
A	?	∞
B	B	1
C	C	1
D	D	0
E	?	∞
F	F	1

Dest.	Hop	Metric
A	A	0
B	B	1
C	C	1
D	?	∞
E	?	∞
F	?	∞
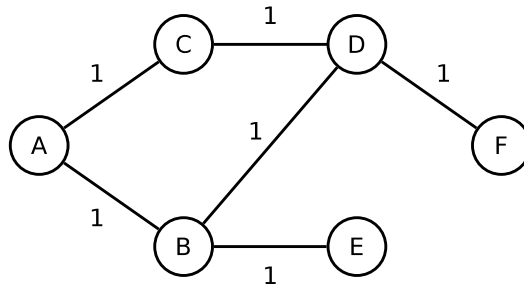
Dest.	Hop	Metric
A	?	∞
B	?	∞
C	?	∞
D	D	1
E	?	∞
F	F	0

Dest.	Hop	Metric
A	A	1
B	B	0
C	?	∞
D	D	1
E	E	1
F	?	∞

Dest.	Hop	Metric
A	?	∞
B	B	1
C	?	∞
D	?	∞
E	E	0
F	?	∞

Step 3

Dest.	Hop	Metric
A	A	1
B	A	2
C	C	0
D	D	1
E	?	∞
F	D	2

Dest.	Hop	Metric
A	B	2
B	B	1
C	C	1
D	D	0
E	B	2
F	F	1

Dest.	Hop	Metric
A	A	0
B	B	1
C	C	1
D	B	2
E	B	2
F	?	∞
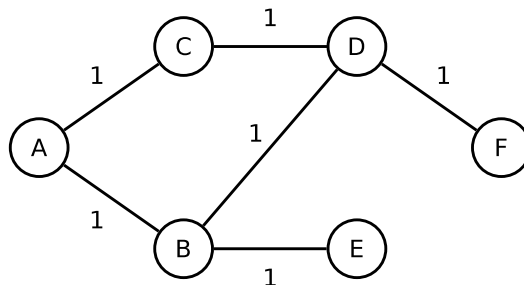
Dest.	Hop	Metric
A	?	∞
B	D	2
C	D	2
D	D	1
E	?	∞
F	F	0

Dest.	Hop	Metric
A	A	1
B	B	0
C	D	2
D	D	1
E	E	1
F	D	2

Dest.	Hop	Metric
A	B	2
B	B	1
C	?	∞
D	B	2
E	E	0
F	?	∞

Step 4

Dest.	Hop	Metric
A	A	1
B	A	2
C	C	0
D	D	1
E	A	3
F	D	2

Dest.	Hop	Metric
A	B	2
B	B	1
C	C	1
D	D	0
E	B	2
F	F	1

Dest.	Hop	Metric
A	A	0
B	B	1
C	C	1
D	B	2
E	B	2
F	B	3
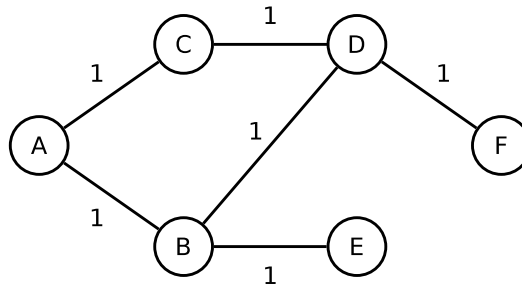
Dest.	Hop	Metric
A	D	3
B	D	2
C	D	2
D	D	1
E	D	3
F	F	0

Dest.	Hop	Metric
A	A	1
B	B	0
C	D	2
D	D	1
E	E	1
F	D	2

Dest.	Hop	Metric
A	B	2
B	B	1
C	B	3
D	B	2
E	E	0
F	B	3

Exercise 16   (Dijkstra's Algorithm)

1. Calculate the shortest path from node A to all other nodes using Dijkstra's algorithm.

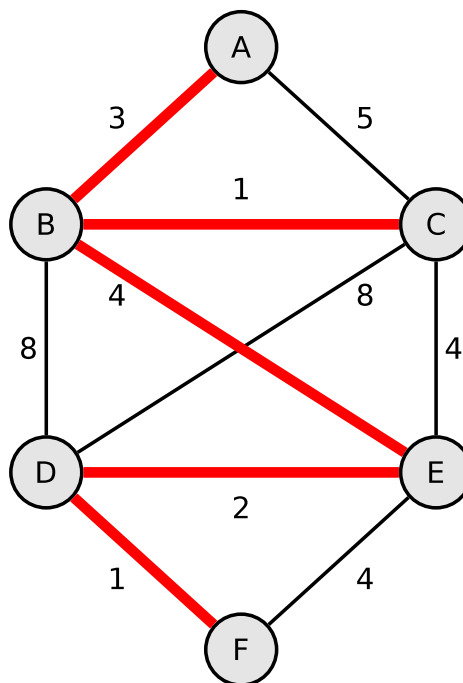Source: Jörg Roth. Prüfungstrainer Rechnernetze. Vieweg (2010)
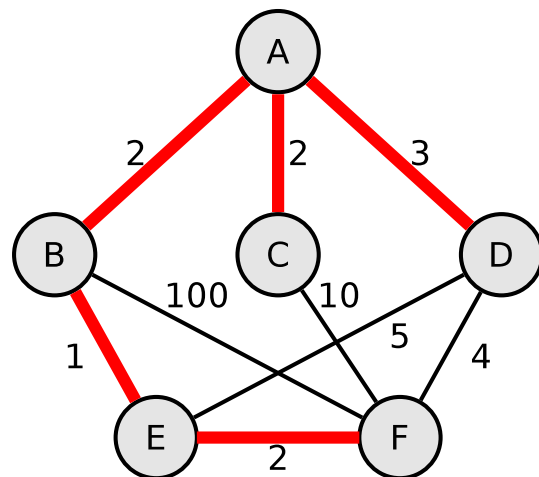
		Distance values				
	Initial	Step 1	Step 2	Step 3	Step 4	Step 5
d_A	**0** ← min	$\underline{0}$ ✓	0 ✓	0 ✓	0 ✓	0 ✓
d_B	∞	**2** ← min	$\underline{2}$ ✓	2 ✓	2 ✓	2 ✓
d_C	∞	2	**2** ← min	$\underline{2}$ ✓	$\underline{2}$ ✓	$\underline{2}$ ✓
d_D	∞	3	3	**3** ← min	$\underline{3}$ ✓	$\underline{3}$ ✓
d_E	∞	∞	3	3	**3** ← min	$\underline{3}$ ✓
d_F	∞	∞	102	12	7	**5** ← min

The active node is underlined.

Nodes visited = {A, B, C, D, E, F}

Shortest paths = {A, A⟶B, A⟶C, A⟶D, B⟶E, E⟶F}

2. Calculate the shortest path from node A to all other nodes using Dijkstra's algorithm.

		Distance values				
	Initial	Step 1	Step 2	Step 3	Step 4	Step 5
d_A	**0** ← min	$\underline{0}$ ✓	0 ✓	0 ✓	0 ✓	0 ✓
d_B	∞	**3** ← min	$\underline{3}$ ✓	3 ✓	3 ✓	3 ✓
d_C	∞	5	**4** ← min	$\underline{4}$ ✓	4 ✓	4 ✓
d_D	∞	∞	11	11	**9** ← min	$\underline{9}$ ✓
d_E	∞	∞	7	**7** ← min	$\underline{7}$ ✓	7 ✓
d_F	∞	∞	∞	∞	11	**10** ← min

The active node is underlined.

Nodes visited = {A, B, C, E, D, F}

Shortest paths = {A, A⟶B, B⟶C, B⟶E, E⟶D, D⟶F}

Exercise 17   (Internet Control Message Protocol)

1. What is the **function** of the Internet Control Message Protocol (ICMP)?

   It is used for the exchange of diagnostic and control messages, as well as error messages.

2. Give two examples for **command line tools**, which use ICMP.

   ping, tranceroute

Exercise 18   (IPv6)

1. Simplify these IPv6 addresses:

   - `1080:0000:0000:0000:0007:0700:0003:316b`

     Solution: `1080::7:700:3:316b`

   - `2001:0db8:0000:0000:f065:00ff:0000:03ec`

     Solution: `2001:db8::f065:ff:0:3ec`

   - `2001:0db8:3c4d:0016:0000:0000:2a3f:2a4d`

     Solution: `2001:db8:3c4d:16::2a3f:2a4d`

   - `2001:0c60:f0a1:0000:0000:0000:0000:0001`

     Solution: `2001:c60:f0a1::1`

   - `2111:00ab:0000:0004:0000:0000:0000:1234`

     Solution: `2111:ab:0:4::1234`

2. Provide all positions of these simplified IPv6 addresses:

   - `2001::2:0:0:1`

     Solution: `2001:0000:0000:0000:0002:0000:0000:0001`

   - `2001:db8:0:c::1c`

     Solution: `2001:0db8:0000:000c:0000:0000:0000:001c`

   - `1080::9956:0:0:234`

     Solution: `1080:0000:0000:0000:9956:0000:0000:0234`

- `2001:638:208:ef34::91ff:0:5424`

  Solution: `2001:0638:0208:ef34:0000:91ff:0000:5424`

- `2001:0:85a4::4a1e:370:7112`

  Solution: `2001:0000:85a4:0000:0000:4a1e:0370:7112`