Computer Networks

Exercise Session 10

Prof. Dr. Oliver Hahm

Frankfurt University of Applied Sciences
Faculty 2: Computer Science and Engineering
 oliver.hahm@fb2.fra-uas.de
 https://teaching.dahahm.de

January 13, 2023

Prof. Dr. Oliver Hahm - Computer Networks - Exercise Session 10 - WS 22/23

General Schedule

All exercises will follow this general schedule

- Identify potential understanding problems
 - \rightarrow Ask your questions
 - \rightarrow Recap of the lecture
- Address the understanding problems
 - \rightarrow Answer your questions
 - \rightarrow Repeat certain topics
- \blacksquare Walk through the exercises/solutions \rightarrow Some hints and guidance
 - \rightarrow Work time or presentation of results

Network Layer: Routing Schemes

- the requirements for a routing protocol
- how routing algorithms can be categorized
- flooding and hot-potato as examples for local routing algorithms
- the difference between source routing and hop-by-hop routing
- the difference between reactive and proactive routing algorithms
- how metrics are used to calculate the path costs

Network Layer: Distance Vector Routing

- that distance vector routing protocols exchange forwarding tables between neighbors
- RIP as an example for a distance vector routing protocol
- how the Bellman-Ford Algorithm works
- what the Count-to-Infinity problem is
- how Split Horizon (with Poison Reversed) can be used to mitigate this problem

Network Layer: Link State Routing

- that link state routing protocols exchange information between all routers
- OSPF as an example for a link state routing protocol
- that OSPF allows for routing hierarchies
- how the Dijkstra Algorithm works

Network Layer: More Routing Protocols

- IS-IS as another example for a link state routing protocol
- RPL as routing protocol for resource-constrained node networks (aka IOT networks)
- OLSR as link state routing protocol for wireless ad-hoc networks
- BGP as an example for an inter-domain routing protocol

Transport Layer: Characteristics

- the properties, tasks, and challenges of transport layer protocols
- how port numbers are used for addressing on the transport layer
- which ranges for these port numbers are defined by the IANA
- that the common interface on the transport layer is a socket

Transport Layer: TCP

- the functioning and segment structure of TCP
- how flow control works in TCP
- what congestion control is
- which enhancements for TCP exist
- how a TCP connection is implemented with sockets
- what SYN Flood DOS attack is

Transport Layer: UDP

- the functioning and segment structure of UDP
- that UDP is much simpler compared to TCP and allows for best-effort communication
- how a UDP server and client is implemented with sockets

Transport Layer: Other Protocols

- SCTP as another connection-oriented transport layer protocol
- DCCP to be used for real-time applications
- QUIC as the newest relevant transport layer protocol to deal with shortcomings of TCP for web traffic

- An IPv4 address without a subnet mask is ambiguous
 - \Rightarrow Tools like *iputils* (\rightarrow ip) require the IPv4 address in CIDR notation
 - E.g., ip addr add 192.168.7.3/24 dev wlan0
 - Reminder: CIDR notations specifies the number of masked bits $\Rightarrow /24 \rightarrow 255.255.255.0$
- 10.1.2.3/24 is different from 10.1.2.3/16¹

¹Both addresses cannot be used in the same network.

- An IPv4 address without a subnet mask is ambiguous
 - \Rightarrow Tools like *iputils* (\rightarrow ip) require the IPv4 address in CIDR notation
 - E.g., ip addr add 192.168.7.3/24 dev wlan0
 - Reminder: CIDR notations specifies the number of masked bits $\Rightarrow /24 \rightarrow 255.255.255.0$
- 10.1.2.3/24 is different from 10.1.2.3/16¹

00001010 0000001 0000010 00000011 AND 11111111 1111111 1111111 00000000 versus 00001010 00000001 00000010 00000011 AND 11111111 11111111 00000000 00000000

- An IPv4 address without a subnet mask is ambiguous
 - \Rightarrow Tools like *iputils* (\rightarrow ip) require the IPv4 address in CIDR notation
 - E.g., ip addr add 192.168.7.3/24 dev wlan0
 - Reminder: CIDR notations specifies the number of masked bits $\Rightarrow /24 \rightarrow 255.255.255.0$
- 10.1.2.3/24 is different from 10.1.2.3/16¹

00001010 00000001 0000010 00000011 AND
11111111 1111111 1111111 00000000
versus
00001010 00000001 00000010 00000011 AND
11111111 11111111 00000000 00000000

 Subnet masks are often multiples of eight bits, but not always e.g., 10.21.42.83/28

- An IPv4 address without a subnet mask is ambiguous
 - \Rightarrow Tools like *iputils* (\rightarrow ip) require the IPv4 address in CIDR notation
 - E.g., ip addr add 192.168.7.3/24 dev wlan0
 - Reminder: CIDR notations specifies the number of masked bits $\Rightarrow /24 \rightarrow 255.255.255.0$
- 10.1.2.3/24 is different from 10.1.2.3/16¹

O0001010 00000001 0000010 00000011 AND
11111111 11111111 11111111 00000000
versus
00001010 00000001 00000010 00000011 AND
11111111 11111111 00000000 00000000

- Subnet masks are often multiples of eight bits, but not always e.g., 10.21.42.83/28
 - What's the subnet mask for this address?

- An IPv4 address without a subnet mask is ambiguous
 - \Rightarrow Tools like *iputils* (\rightarrow ip) require the IPv4 address in CIDR notation
 - E.g., ip addr add 192.168.7.3/24 dev wlan0
 - Reminder: CIDR notations specifies the number of masked bits $\Rightarrow /24 \rightarrow 255.255.255.0$
- 10.1.2.3/24 is different from 10.1.2.3/16¹

00001010 00000001 0000010 00000011 AND
11111111 1111111 1111111 00000000
versus
00001010 00000001 00000010 00000011 AND
11111111 11111111 00000000 00000000

Subnet masks are often multiples of eight bits, but not always e.g., 10.21.42.83/28

- What's the subnet mask for this address?
- \rightarrow /28 \rightarrow 11111111 11111111 11111111 11110000 \rightarrow 255.255.255.240

- An IPv4 address without a subnet mask is ambiguous
 - \Rightarrow Tools like *iputils* (\rightarrow ip) require the IPv4 address in CIDR notation
 - E.g., ip addr add 192.168.7.3/24 dev wlan0
 - Reminder: CIDR notations specifies the number of masked bits $\Rightarrow /24 \rightarrow 255.255.255.0$
- 10.1.2.3/24 is different from 10.1.2.3/16¹

O0001010 00000001 0000010 00000011 AND
11111111 1111111 1111111 00000000
versus
00001010 00000001 00000010 00000011 AND
11111111 11111111 00000000 00000000

Subnet masks are often multiples of eight bits, but not always

- e.g., 10.21.42.83/28
 - What's the subnet mask for this address?
 - \rightarrow /28 \rightarrow 11111111 11111111 11111111 11110000 \rightarrow 255.255.255.240
 - What's the network address?

- An IPv4 address without a subnet mask is ambiguous
 - \Rightarrow Tools like *iputils* (\rightarrow ip) require the IPv4 address in CIDR notation
 - E.g., ip addr add 192.168.7.3/24 dev wlan0
 - Reminder: CIDR notations specifies the number of masked bits $\Rightarrow /24 \rightarrow 255.255.255.0$
- 10.1.2.3/24 is different from 10.1.2.3/16¹

O0001010 00000001 00000010 00000011 AND
11111111 11111111 11111111 00000000
versus
00001010 00000001 00000010 00000011 AND
11111111 11111111 00000000 00000000

Subnet masks are often multiples of eight bits, but not always

- e.g., 10.21.42.83/28
 - What's the subnet mask for this address?
 - \rightarrow /28 \rightarrow 11111111 11111111 11111111 11110000 \rightarrow 255.255.255.240
 - What's the network address?
 - \rightarrow 10.21.52.80/28

¹Both addresses cannot be used in the same network.

Prof. Dr. Oliver Hahm - Computer Networks - Exercise Session 10 - WS 22/23

Exercise 2: Inter-Networking

- On Linux you can query your routing table with *iputils*
 - $(\rightarrow \text{ ip route show or simply ip r})$
- On Windows and Linux you can also use netstat -r[n]
- The result may look like this:

Kernel IP routing table							
Destination	Gateway	Genmask	Flags	MSS Window	irtt Iface		
default	10.51.0.1	0.0.0.0	UG	0 0	0 wlan0		
10.2.0.0	0.0.0.0	255.255.255.0	U	0 0	0 enp0s31f6		
10.51.0.0	0.0.0.0	255.255.0.0	U	0 0	0 wlan0		
192.168.0.0	0.0.0.0	255.252.0.0	U	0 0	0 wlan0		

Whenever an IPv4 packet has to be sent a longest prefix match between the destination address and the entries in the table is performed

What does it mean that we have multiple entries for the interface wlan0?

Exercise 3: Subnetting

IP address	172.21.240.90	10101100	00010101	11110000	01011010
Class B	255.255.0.0	11111111	11111111	0000000	0000000
Subnet mask	255.255.255.224	11111111	11111111	11111111	111 00000

Exercise 3: Subnetting

IP address	172.21.240.90	10101100 000	010101	11110000	01011010
Class B	255.255.0.0	11111111 111	11111	00000000	00000000
Subnet mask	255.255.255.224	11111111 111	11111	11111111	11100000
Subnet ID	1922	10101100 000)10101	11110000	0100000

Exercise 3: Subnetting

IP address	172.21.240.90	10101100	00010101	11110000	01011010
Class B	255.255.0.0	11111111	11111111	00000000	0000000
Subnet mask	255.255.255.224	11111111	11111111	11111111	111 00000
Subnet ID	1922	10101100	00010101	11110000	01000000

IP address AND (NOT subnet mask) = host ID

IP address	172.21.240.90	10101100 00010101 11110000 01011010
Subnet mask	255.255.255.224	11111111 1111111 11111111 11100000
Inverse subnet mask	000.000.000.31	0000000 0000000 0000000 00011111
Host ID	26	0000000 0000000 0000000 00011010

Exercise 4: IPv4 Checksum

RFC 791, 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. For purposes of computing the checksum, the value of the checksum field is zero".

- 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: $2907D \implies 10\ 1001\ 0000\ 0111\ 1101$
- 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.

Exercise 4: IPv4 Checksum

- 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: $2FFD \implies 10 \ 1111 \ 1111 \ 1111 \ 1101$

```
    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

Exercise 5: Address Types and Spaces

- Private addresses (unique local addresses in IPv6)
 - "have no global meaning"²
 - "routing information [...] shall not be propagated"² in the Internet, and
 - "packets with private source or destination addresses should not be forwarded"²
- May be forwarded inside a LAN (\rightarrow *link-local addresses* are never forwarded)
- Edge routers ideally filter traffic using address from private address space

Exercise 5: Address Types and Spaces

- Private addresses (unique local addresses in IPv6)
 - "have no global meaning"²
 - "routing information [...] shall not be propagated"² in the Internet, and
 - "packets with private source or destination addresses should not be forwarded"²
- May be forwarded inside a LAN (\rightarrow *link-local addresses* are never forwarded)
- Edge routers ideally filter traffic using address from private address space

Pinging broadcast addresses

user@host> ping -b 10.0.34.255 PING 10.0.34.0 (10.0.34.0) from 10.0.34.197 : 56(84) bytes of data. 64 bytes from 10.0.34.197: icmp_seq=1 ttl=64 time=0.049 ms 64 bytes from 10.0.34.236: icmp_seq=1 ttl=255 time=0.163 ms (DUP!) 64 bytes from 10.0.34.206: icmp_seq=1 ttl=255 time=0.211 ms (DUP!) 64 bytes from 10.0.34.196: icmp_seq=1 ttl=255 time=0.213 ms (DUP!) 64 bytes from 10.0.34.181: icmp_seq=1 ttl=255 time=0.220 ms (DUP!) 64 bytes from 10.0.34.174: icmp_seq=1 ttl=255 time=0.243 ms (DUP!) 64 bytes from 10.0.34.133: icmp_seq=1 ttl=255 time=0.245 ms (DUP!)

²RFC 1918

Exercise 6: Fragmenting IP Packets

- Any router can fragment (unless the DF bit is not set)
- Only the receiver reassembles
- In IPv4:
 - Any router "must be able to forward a datagram of 68 octets without further fragmentation"³
 - Any host "must be able to receive a datagram of 576 octets either in one piece or in fragments to be reassembled"³
- "IPv6 requires that every link in the internet have an MTU of 1280"⁴ octets or greater

Exercise 6: Fragmenting IP Packets

No.	Time	Source	Destination	Protocol	Length Info			
	3 1,686621	192,168,12,192	192,168,1,192	IPv4	1508 Fragmented IP protocol	(proto=UDP 17, off=0, ID=02ba)	[Reassembled in #4]	
	4 1.686630	192.168.12.192	192.168.1.192	UDP	91 Source port: scp-config	Destination port: safetynetp	-	
	5 1,686874	192,168,1,192	192,168,12,192	IPv4	1508 Fragmented IP protocol ((proto=UDP 17, off=0, ID=3054)	[Reassembled in #6]	
	6 1.686891	192.168.1.192	192.168.12.192	UDP	91 Source port: safetynetp	Destination port: scp-config	-	
<								
-	ama 4 · 91 hz	tes on wire (728 h	dts) 91 bytes cantured (778 hirs)				
E ET	hernet II. S	rc: b8:ca:3a:5f:24	:d2 (b8:ca:3a:5f:24:d2)	Dst: TospurEl 13	7e:0b (6c:92:bf:13:7e:0b)			
E In	ternet Proto	col version 4. Src	: 10, 55, 205, 215 (10, 55, 20	(5,215), Dst: 10.	55, 205, 228 (10, 55, 205, 228)			
	0100 =	version: 4	. 1013312031213 (20133120					
	Header Tengt	n: 20 bytes						
œ	Differentiat	ed Services Field:	0x00 (DSCP 0x00: Default	; ECN: 0x00: Not-	-ECT (Not ECN-Capable Transport	())		
	Total Length	. //						
	Identificati	on: 0x431b (1/1/9)						
1±	Flags: 0x00							
	Fragment off:	Set: U						
	Time to rive							
	Protocor: OD	P (17)						
	Fourcos 10 5	5 205 215 (10 55 2	05 215)					
	Destination:	10 55 205 228 (10	55 205 228)					
	Source GeoT	P: Unknown]						
	Destination	GeoTP: Unknown]						
H US	er Dataoran I	Protocol, Src Port	: 53834 (53834), Dst Port	: oty (8472)				
e vi	rtual extens	ible Local Area Ne	Twork					
I Et	hernet II. S	rc: a2:36:11:af:b9	:a4 (a2:36:11:af:b9:a4).	Dst: b2:8b:8e:60	e6:b9 (b2:8b:8e:60:e6:b9)			
E In	ternet Proto	col version 4, src	: 192,168,12,192 (192,168	.12.192), DST: 1	92,168,1,192 (192,168,1,192)			
	0100 = '	version: 4						
1	Header lengt	h: 20 bytes						
Đ	Differentiat	ed Services Field:	0x00 (DSCP 0x00: Default	; ECN: 0x00: Not-	-ECT (Not ECN-Capable Transport	t))		
	Total Length	: 27						
8	Identification: 0x02ba (698)							
æ	Flags: 0x00							
	Fragment offset: 1424							
(°	Time to live: 64							
	Protocol: UDP (17)							
Đ	Header check	sun: Oxe795 [corre	ct]					
1 1	source: 192.	168.12.192 (192.16	8.12.192)					
2	Destination: 192.168.1.192 (192.168.1.192)							
	Lource Geoi	P: Unknownj						
	Destination	Geoip: Unknownj						
ы	LZ IPV4 Frag	ments (1451 bytes)	: #5(1424), #4(7)]					
	franc, s.	payload: 1424 142	0 (7 hyter)]					
	Trane. 4.	pay 10au. 1424-145	U C/ UYCES/I					
1	[Roass orb]	ad TOyd longth: 14	21]					
-	er Dataoran I	Protocol Sec Port	sco-config (10001) DST	Port - safetynet	(40000)			
(i) 03	ta (1423 hvt	notocor, sic Porc	. sep com (g (10001), bsc	or cr. sarecynec	(40000)			
~ ~	an farma nate							

Source: https://hustcat.github.io/

Recap of the Lecture

Exercises

Exercise 7-9: Routing



User

Prof. Dr. Oliver Hahm - Computer Networks - Exercise Session 10 - WS 22/23

Recap of the Lecture

Exercises

Exercise 7-9: Routing



Prof. Dr. Oliver Hahm - Computer Networks - Exercise Session 10 - WS 22/23

Exercise 9: Dijkstra Algorithm

Given: The following network



Determine the spanning tree of shortest paths using the link state routing protocol (Dijkstra's algorithm) of node A.

Source: Jörg Roth. Prüfungstrainer Rechnernetze: Aufgaben und Lösungen. Vieweg (2010)