Computer Networks

Exercise Session 09

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 15, 2025

General Schedule

All exercises will follow this general schedule

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

Network Layer: Addressing

You have seen ...

- the purpose and format of IPv4 and IPv6 addresses
- the original classes of IPv4 networks, what CIDR and what subnets are
- how to connect private networks to the Internet using NAT
- that IP datagrams can be fragmented if they are too big for a single frame on the data link layer
- why a successor for IPv4 was needed and how IPv6 tackles the challenges

Exercise 01: IPv4 Addressing

- An IPv4 address without a subnet mask is ambiguous
  - ⇒ Tools like *iputils* (→ `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
      ⇒ `/24` → `255.255.255.0`
- `10.1.2.3/24` is different from `10.1.2.3/16`[1]

[1] Both addresses cannot be used in the same network.

Exercise 01: IPv4 Addressing

- An IPv4 address without a subnet mask is ambiguous
  - ⇒ Tools like *iputils* (→ `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
      ⇒ `/24` → `255.255.255.0`
- `10.1.2.3/24` is different from `10.1.2.3/16`[1]
  - `00001010 00000001 00000010 00000011 AND`
    `11111111 11111111 11111111 00000000`
    versus
    `00001010 00000001 00000010 00000011 AND`
    `11111111 11111111 00000000 00000000`

[1] Both addresses cannot be used in the same network.

Exercise 01: IPv4 Addressing

- An IPv4 address without a subnet mask is ambiguous
  - ⇒ Tools like *iputils* (→ `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
      ⇒ `/24` → `255.255.255.0`
- `10.1.2.3/24` is different from `10.1.2.3/16`[1]
  - `00001010 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`

[1] Both addresses cannot be used in the same network.

Exercise 01: IPv4 Addressing

- An IPv4 address without a subnet mask is ambiguous
  - ⇒ Tools like *iputils* (→ `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
      ⇒ `/24` → `255.255.255.0`
- `10.1.2.3/24` is different from `10.1.2.3/16`[1]
  - `00001010 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?

[1] Both addresses cannot be used in the same network.

Exercise 01: IPv4 Addressing

- An IPv4 address without a subnet mask is ambiguous
  - ⇒ Tools like *iputils* (→ `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
      ⇒ `/24` → `255.255.255.0`
- `10.1.2.3/24` is different from `10.1.2.3/16`[1]
  - `00001010 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?
  - → `/28` → `11111111 11111111 11111111 11110000` → `255.255.255.240`

[1] Both addresses cannot be used in the same network.

Exercise 01: IPv4 Addressing

- An IPv4 address without a subnet mask is ambiguous
  - ⇒ Tools like *iputils* (→ `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
      ⇒ `/24` → `255.255.255.0`
- `10.1.2.3/24` is different from `10.1.2.3/16`[1]
  - `00001010 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?
  - → `/28` → `11111111 11111111 11111111 11110000` → `255.255.255.240`
    - What's the network address?

[1] Both addresses cannot be used in the same network.

Exercise 01: IPv4 Addressing

- An IPv4 address without a subnet mask is ambiguous
  - ⇒ Tools like *iputils* (→ `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
      ⇒ `/24` → `255.255.255.0`
- `10.1.2.3/24` is different from `10.1.2.3/16`[1]
  - `00001010 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?
  - → `/28` → `11111111 11111111 11111111 11110000` → `255.255.255.240`
  - What's the network address?
  - → `10.21.52.80/28`

[1] Both addresses cannot be used in the same network.

Exercise 2: Forwarding Process

- The OS uses a forwarding table (or forwarding information base (FIB)) to select the appropriate interface for sending a packet
- The selector is the destination IP address of the outgoing (or forwarded) packet
- The FIB contains at least two columns:
  - The destination network address
  - The interface
- Optionally it may contain a gateway
- The OS performs a longest prefix match on the selector

Exercise 2: Longest Prefix Matching

- The longest (*best*) matching prefix from the FIB is chosen
- The destination IP address is compared bit by bit with the network addresses in the FIB
- The number of compared bits depends on the prefix length of the FIB entry
- The longest matching prefix is selected and the according interface will be chosen
- There is typically a default entry (`0.0.0.0/0` for IPv4) that always matches

Exercise 2: Inspect the FIB

- On Linux you can query your routing table with *iputils* (→ `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
```

Exercise 3: Subnetting

IP address	172.21.240.90	10101100 00010101 11110000 01011010
Class B	255.255.0.0	11111111 11111111 00000000 00000000
Subnet mask	255.255.255.224	11111111 11111111 11111111 11100000

Exercise 3: Subnetting

IP address	172.21.240.90	10101100 00010101 11110000 01011010
Class B	255.255.0.0	11111111 11111111 00000000 00000000
Subnet mask	255.255.255.224	11111111 11111111 11111111 11100000
Network address	172.21.240.64	10101100 00010101 11110000 01000000

Exercise 3: Subnetting

IP address	172.21.240.90	10101100 00010101 11110000 01011010
Class B	255.255.0.0	11111111 11111111 00000000 00000000
Subnet mask	255.255.255.224	11111111 11111111 11111111 11100000
Network address	172.21.240.64	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 11111111 11111111 11100000
Inverse subnet mask	000.000.000.31	00000000 00000000 00000000 00011111
Host ID	26	00000000 00000000 00000000 00011010

Exercise 4: 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

[2] RFC 1918

Exercise 4: 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!)
```

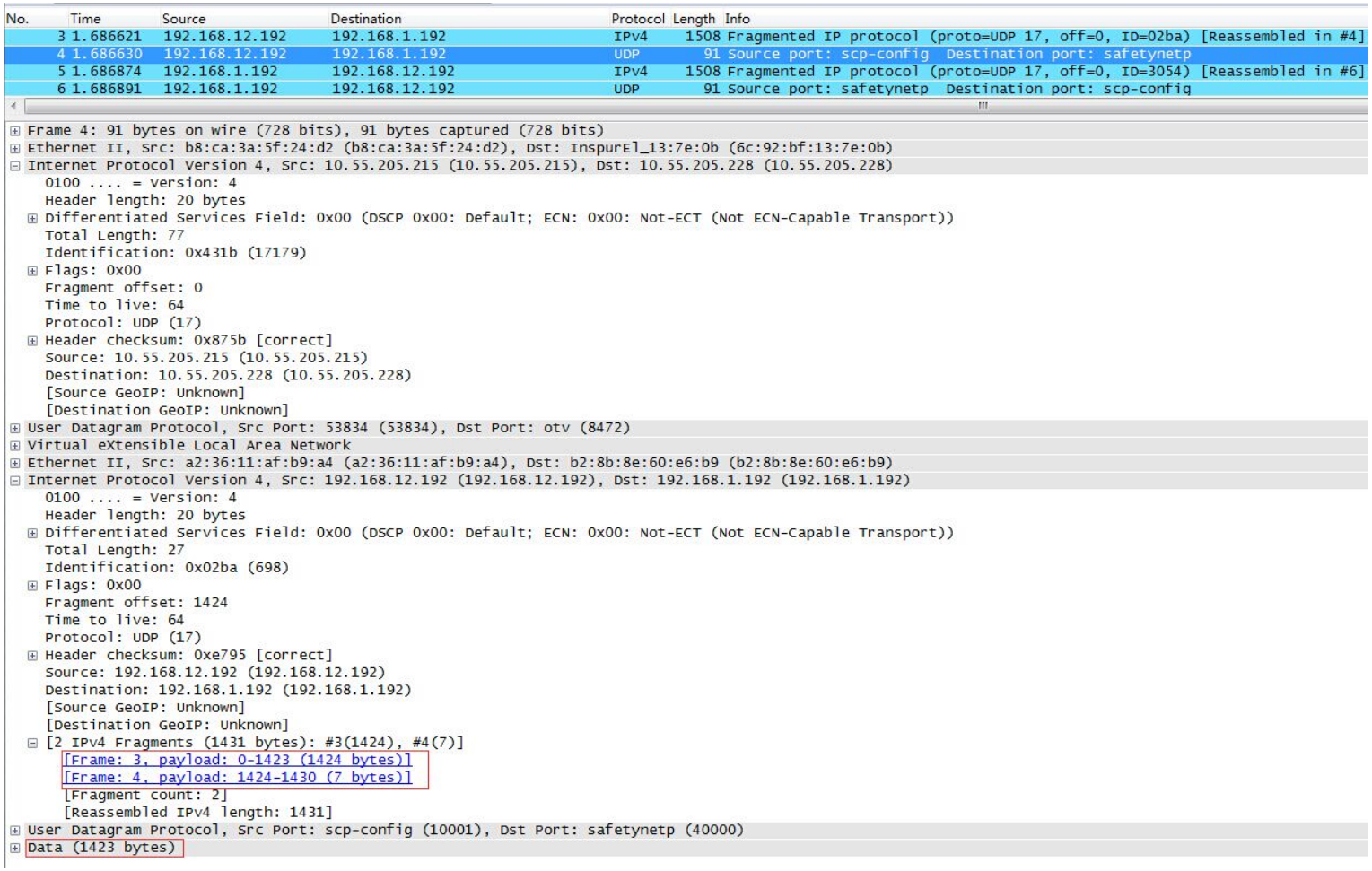
[2] RFC 1918

Exercise 5: 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

[3] RFC 791
[4] RFC 2460

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

```
⊞ Frame 4: 91 bytes on wire (728 bits), 91 bytes captured (728 bits)
⊞ Ethernet II, Src: b8:ca:3a:5f:24:d2 (b8:ca:3a:5f:24:d2), Dst: InspurEl_13:7e:0b (6c:92:bf:13:7e:0b)
⊟ Internet Protocol Version 4, Src: 10.55.205.215 (10.55.205.215), Dst: 10.55.205.228 (10.55.205.228)
    0100 .... = Version: 4
    Header length: 20 bytes
  ⊞ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
    Total Length: 77
    Identification: 0x431b (17179)
  ⊞ Flags: 0x00
    Fragment offset: 0
    Time to live: 64
    Protocol: UDP (17)
  ⊞ Header checksum: 0x875b [correct]
    Source: 10.55.205.215 (10.55.205.215)
    Destination: 10.55.205.228 (10.55.205.228)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
⊞ User Datagram Protocol, Src Port: 53834 (53834), Dst Port: otv (8472)
⊞ Virtual eXtensible Local Area Network
⊞ Ethernet II, Src: 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)
⊟ Internet Protocol Version 4, Src: 192.168.12.192 (192.168.12.192), Dst: 192.168.1.192 (192.168.1.192)
    0100 .... = Version: 4
    Header length: 20 bytes
  ⊞ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
    Total Length: 27
    Identification: 0x02ba (698)
  ⊞ Flags: 0x00
    Fragment offset: 1424
    Time to live: 64
    Protocol: UDP (17)
  ⊞ Header checksum: 0xe795 [correct]
    Source: 192.168.12.192 (192.168.12.192)
    Destination: 192.168.1.192 (192.168.1.192)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
  ⊟ [2 IPv4 Fragments (1431 bytes): #3(1424), #4(7)]
      [Frame: 3, payload: 0-1423 (1424 bytes)]
      [Frame: 4, payload: 1424-1430 (7 bytes)]
      [Fragment count: 2]
      [Reassembled IPv4 length: 1431]
⊞ User Datagram Protocol, Src Port: scp-config (10001), Dst Port: safetynetp (40000)
⊞ Data (1423 bytes)
```

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