

# Practical Computer Networks and Applications

## Exercise 1 – IP Version 4 Networks

Prof. Dr. Baun, Prof. Dr. Ebinger, Prof. Dr. Hahm, Prof. Dr. Kappes,  
Dipl. Inf. (FH) Maurizio Petrozziello

`{baun, peter.ebinger, oliver.hahm, kappes, petrozziello}@fb2.fra-uas.de`

**Frankfurt University of Applied Sciences**  
**Faculty of Computer Science and Engineering**  
**Nibelungenplatz 1**  
**60318 Frankfurt am Main**

# Contents

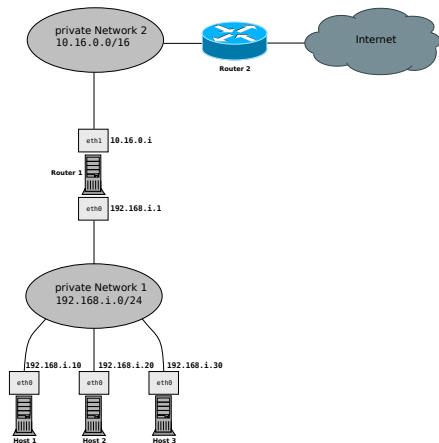
## Exercise 1

Networking in Linux

Wireshark

Some useful commands

# Network Topology – Exercise 1



Network Topology of lab exercise 1

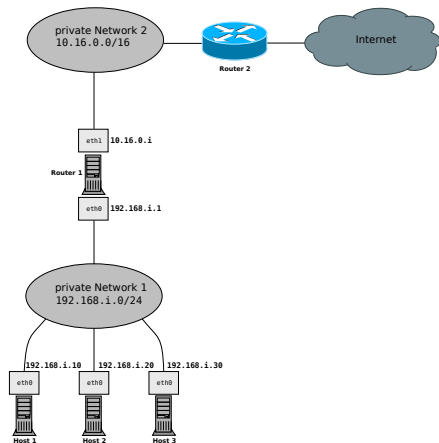
## Private Network 1:

- 192.168.i.0/24
- Private network for host machines and Routers
- The number *i* in the IP address is a placeholder for your **group number**!

## Private Network 2:

- 10.16.0.0/16
- Private network connecting all networks

## Network Topology – Private Network 2



Network Topology of lab exercise 1

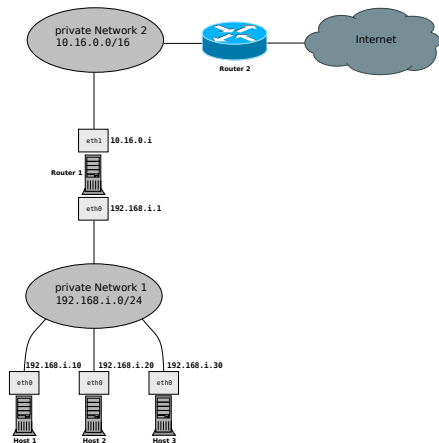
### Private Network 2:

- 10.16.0.0/16

### Router 2:

- Address: 10.16.0.200
- **Router 2** is the gateway for all routers in private network 1!
- The route to **Router 2** needs to be configured on **Router 1**!
- **Router 2** runs a web server on **port 80**!

# Network Topology – Private Network 1



Network Topology of lab exercise 1

## Private network 1:

- 192.168.i.0/24

## Router 1:

- Has two interfaces
- eth0: 192.168.i.1
- eth1: 10.16.0.i

## Host network:

- Router 1:** 192.168.i.1
- Host 1:** 192.168.i.10
- Host 2:** 192.168.i.20
- Host 3:** 192.168.i.30

# Network Topology – Exercise 1 – Objectives

In the lab exercise you need to accomplish. . .

- a successful static configuration of the machines!
- working static routing on the machines!
- reachability of all machines (all hosts including **Router 1** and **2**)!
- captures of various protocols using Wireshark!

# Contents

Exercise 1

**Networking in Linux**

Wireshark

Some useful commands

# Network Interface Names in Linux

The network interfaces in Linux can be configured with the tool `ip`

- In the literature and the internet you often find the interface names `eth0, eth1, ..., ethX!`
- In practice the device names differ and often the naming schemes `enp1sXfX` or `enoX` can be seen!<sup>1</sup>

---

<sup>1</sup>This is a good source of information: <https://www.freedesktop.org/wiki/Software/systemd/PredictableNetworkInterfaceNames/>



# Name Resolution in Linux

- The name resolution is configured in the file `/etc/resolv.conf`!
- The entry of that file is used to resolve domain names into IP addresses!<sup>2</sup>
- The format of the entries is `nameserver <IP ADDRESS>`!
- The most commonly known entry is `8.8.8.8` and refers to a Google name server!

---

<sup>2</sup>More detailed information on the name resolution in the lecture slides!

# Commandline-tools for Networking in Linux (1/4)

The `ip` command from the `iproute2` package is a very powerful tool and replaces the deprecated `ifconfig` tool!

Its primary use cases are:

- statistics of the network links
- network configuration tasks (address assignment, ARP cache inspection, routing tables, etc.)
- configuration of static routes

## Importance of the `ip` command

Please get familiar with the `ip` command and its options, since it plays a pivotal role in the configuration of the machines for lab exercise 1!

# Commandline-tools for Networking in Linux (2/4)

The `ip`<sup>3</sup> command:

- `ip link ...` – configuration of interfaces
- `ip addr ...` – configuration of addresses
- `ip route ...` – configuration of routes

---

<sup>3</sup>The manpage of `ip` gives you the full list of functions and options!

## Commandline-tools for Networking in Linux (3/4)

The `ping`<sup>4</sup> command is used to send ICMP packets to a host machine in the network! Its primary use cases are:

- gathering statistics on the network
- testing the reachability of a host and the network link
- first step in debugging of network errors

---

<sup>4</sup>The manpage of `ping` gives you the full list of functions and options!

## Commandline-tools for Networking in Linux (4/4)

- `traceroute` – tracks the route of packets to the destination
- `curl` – a tool to transfer data over multiple protocols (HTTP, FTP, etc.)
- `ss` (`socket statistics`) – generates statistics on Transport Layer protocols
- `nc` (`netcat`) – listens and analyzes Transport Layer protocols
- `nmap` – a tool for network analysis and port scanning
- `dig` – displays the domain name lookups and the available name servers

## IP forwarding in Linux routers

In order to enable routing in Linux IP forwarding needs to be activated!

This can be done by setting the Kernel parameter:

- `net.ipv4.ip_forward=1`
- alternatively
- `/proc/sys/net/ipv4/ip_forward`

### Setting Kernel parameters

The options presented above can be set by using `sysctl -w` followed by the parameter to set (here `net.ipv4.ip_forward=1`). Alternatively the parameter can be set by using `cat` and writing the value `1` to the file `/proc/sys/net/ipv4/ip_forward`!

# Contents

Exercise 1

Networking in Linux

**Wireshark**

Some useful commands

# Wireshark

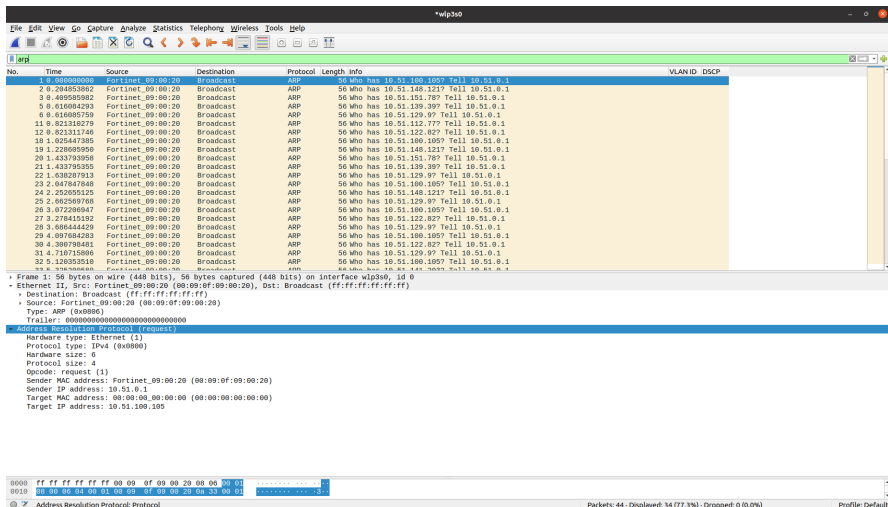
Wireshark is an open-source tool for network analysis

Wireshark features the following functions:

- Graphical user interface
- Collection of transmitted data
- Detailed view of each packet and protocol
- Enables a detailed analysis of network traffic



# Wireshark



**\*wlp350**

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

No.	Time	Source	Destination	Protocol	Length	Info	VLAN ID	DSCP
1	0.080090000	Fortinet_09:00:20	Broadcast	ARP	56	Who has 10.51.100.10? Tell 10.51.0.1		
2	0.204853662	Fortinet_09:00:20	Broadcast	ARP	56	Who has 10.51.148.121? Tell 10.51.0.1		
3	0.409595982	Fortinet_09:00:20	Broadcast	ARP	56	Who has 10.51.151.78? Tell 10.51.0.1		
5	0.616084293	Fortinet_09:00:20	Broadcast	ARP	56	Who has 10.51.139.39? Tell 10.51.0.1		
6	0.616085759	Fortinet_09:00:20	Broadcast	ARP	56	Who has 10.51.129.9? Tell 10.51.0.1		
11	0.821316279	Fortinet_09:00:20	Broadcast	ARP	56	Who has 10.51.112.77? Tell 10.51.0.1		
12	0.821317146	Fortinet_09:00:20	Broadcast	ARP	56	Who has 10.51.122.82? Tell 10.51.0.1		
18	1.025447385	Fortinet_09:00:20	Broadcast	ARP	56	Who has 10.51.100.10? Tell 10.51.0.1		
19	1.228609590	Fortinet_09:00:20	Broadcast	ARP	56	Who has 10.51.148.121? Tell 10.51.0.1		
20	1.433793958	Fortinet_09:00:20	Broadcast	ARP	56	Who has 10.51.151.78? Tell 10.51.0.1		
21	1.433795335	Fortinet_09:00:20	Broadcast	ARP	56	Who has 10.51.139.39? Tell 10.51.0.1		
22	1.638207913	Fortinet_09:00:20	Broadcast	ARP	56	Who has 10.51.129.9? Tell 10.51.0.1		
23	2.047847848	Fortinet_09:00:20	Broadcast	ARP	56	Who has 10.51.100.10? Tell 10.51.0.1		
24	2.252655125	Fortinet_09:00:20	Broadcast	ARP	56	Who has 10.51.148.121? Tell 10.51.0.1		
25	2.662569708	Fortinet_09:00:20	Broadcast	ARP	56	Who has 10.51.129.9? Tell 10.51.0.1		
26	3.072086947	Fortinet_09:00:20	Broadcast	ARP	56	Who has 10.51.100.10? Tell 10.51.0.1		
27	3.278415192	Fortinet_09:00:20	Broadcast	ARP	56	Who has 10.51.122.82? Tell 10.51.0.1		
28	3.686444429	Fortinet_09:00:20	Broadcast	ARP	56	Who has 10.51.129.9? Tell 10.51.0.1		
29	4.097684283	Fortinet_09:00:20	Broadcast	ARP	56	Who has 10.51.100.10? Tell 10.51.0.1		
30	4.308704811	Fortinet_09:00:20	Broadcast	ARP	56	Who has 10.51.122.82? Tell 10.51.0.1		
31	4.710715806	Fortinet_09:00:20	Broadcast	ARP	56	Who has 10.51.129.9? Tell 10.51.0.1		
32	5.126353510	Fortinet_09:00:20	Broadcast	ARP	56	Who has 10.51.100.10? Tell 10.51.0.1		

1 Frame 1: 56 bytes on wire (448 bits), 56 bytes captured (448 bits) on interface wlp350, id 0  
 Ethernet II, Src: Fortinet\_09:00:20 (08:09:0f:09:00:20), Dst: Broadcast (ff:ff:ff:ff:ff:ff)  
 Destination: Broadcast (ff:ff:ff:ff:ff:ff)  
 Source: Fortinet\_09:00:20 (08:09:0f:09:00:20)  
 Type: ARP (0x0806)  
 Trailer: 00000000000000000000000000000000

**Address Resolution Protocol (request)**

Hardware type: Ethernet (1)  
 Protocol type: IPv4 (0x0806)  
 Hardware size: 6  
 Protocol size: 4  
 Opcode: request (1)  
 Sender MAC address: Fortinet\_09:00:20 (08:09:0f:09:00:20)  
 Sender IP address: 10.51.0.1  
 Target MAC address: 00:00:00\_00:00:00 (00:00:00:00:00:00)  
 Target IP address: 10.51.100.10

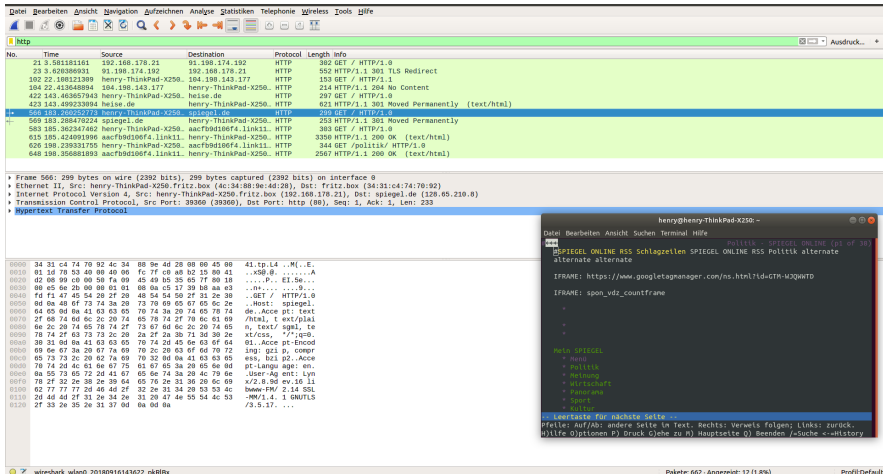
0000 ff ff ff ff ff ff 00 00 0f 09 00 20 08 09 00 01 .....  
 0010 00 00 00 00 01 00 09 0f 09 00 20 0a 33 00 01 .....  
 Address Resolution Protocol: Protocol

Packets: 44 - Discarded: 34 (77.3%) - Dropped: 0 (0.0%) Profile: Default

## Wireshark Desktop

# An Example on Using Wireshark

- The picture shows Wireshark collecting data for a HTTP-connection using lynx to access `www.heise.de`.



The screenshot displays the Wireshark network protocol analyzer interface. The main pane shows a list of captured packets, with packet 621 selected, representing an HTTP GET request to `https://www.heise.de`. The details pane on the right shows the structure of the response, including the status line `200 OK` and the `Content-Type` header `text/html`. The packet bytes pane at the bottom shows the raw data of the response, including the `HTTP/1.1` status line and the `Content-Type` header.

No.	Time	Source	Destination	Protocol	Length	Info
21	3.581181161	192.168.178.21	91.198.174.192	HTTP	382	GET / HTTP/1.0
23	3.628369931	91.198.174.192	192.168.178.21	HTTP	552	HTTP/1.1 301 TLS Redirect
102	22.108211399	henry-ThinkPad-X250	184.198.143.177	HTTP	153	GET / HTTP/1.1
104	22.413648894	184.198.143.177	henry-ThinkPad-X250	HTTP	214	HTTP/1.1 204 No Content
422	143.463657943	henry-ThinkPad-X250	heise.de	HTTP	297	GET / HTTP/1.0
423	143.499233994	henry-ThinkPad-X250	heise.de	HTTP	621	HTTP/1.1 301 Moved Permanently (text/html)
569	153.28476224	spiegel.de	henry-ThinkPad-X250	HTTP	253	HTTP/1.1 301 Moved Permanently
583	185.362347462	henry-ThinkPad-X250	aacfb9d106f4.link11	HTTP	383	GET / HTTP/1.0
615	189.424091996	aacfb9d106f4.link11	henry-ThinkPad-X250	HTTP	3350	HTTP/1.1 200 OK (text/html)
626	190.239321755	henry-ThinkPad-X250	aacfb9d106f4.link11	HTTP	344	GET /polliticr/ HTTP/1.0
648	198.356881893	aacfb9d106f4.link11	henry-ThinkPad-X250	HTTP	2567	HTTP/1.1 200 OK (text/html)

Frame 666: 299 bytes on wire (2392 bits), 299 bytes captured (2392 bits) on interface 0  
 Ethernet II, Src: henry-ThinkPad-X250.fritz.box (4c:3d:88:9e:4d:28), Dst: fritz.box (34:31:c4:74:70:92)  
 Internet Protocol Version 4, Src: henry-ThinkPad-X250.fritz.box (192.168.178.21), Dst: spiegel.de (128.65.210.8)  
 Transmission Control Protocol, Src Port: 39366 (39366), Dst Port: http (80), Seq: 1, Ack: 1, Len: 233  
 Hypertext Transfer Protocol

```

0000  34 31 c4 74 70 92 4c 34 88 9e 4d 28 08 00 45 08  41.tp.L4 .LM[.E.
0010  01 1d 79 53 40 88 40 06 fc 7f c8 a8 62 15 80 41  ..XSB@.....A
0020  02 08 90 0c 60 58 fa 09 45 40 b5 35 65 ff 08 18  ..P...E!56...
0030  86 e5 6e 2b 08 80 01 01 88 0a c5 17 39 18 aa e3  ..#.....9...
0040  fd f1 47 45 54 28 2f 20 48 54 54 50 2f 31 2e 30  ..GET / HTTP/1.0
0050  86 0a 48 6f 73 74 3a 29 73 70 69 65 67 65 6c 2e  ..host: spiegel
0060  64 65 08 0a 41 63 63 65 70 74 3a 20 74 05 78 74  de .Accpe pt: text
0070  2f 68 74 66 6c 2c 20 74 65 78 74 2f 70 6c 61 09  /html, l ext/plai
0080  0e 2c 20 74 65 78 74 2f 73 67 6c 2c 20 74 65  ..n, text/sgml, te
0090  78 74 2f 63 73 73 2c 20 2a 2f 2a 3b 71 3d 30 2e  xt/css, /*?q=0.
0100  31 31 0d 8a 41 63 63 65 70 74 2d 45 6e 63 6f 64  01 .Accpe pt-Encod
0110  69 6e 67 30 69 70 2c 20 63 6f 66 70 72 1ng: g21 b. compr
0120  65 73 73 2c 20 62 78 69 70 32 8d 0a 41 63 63 65  ebb, b21 02 .Accpe
0130  70 74 2d 4c 61 6e 67 75 61 67 65 3a 20 65 6e 6d  pt-Langu age: en.
0140  8a 55 73 65 72 41 67 65 6e 74 3a 20 4c 70 9e  .User-Ag ent: Lyn
0150  78 2f 32 2e 38 2e 39 64 65 70 2c 31 36 20 6c 69  x/2.8.96 ev.16 11
0160  62 77 77 77 20 46 4d 2f 32 2e 31 3a 20 53 53 4c  bww-FM/ 2.14 SSL
0170  20 4d 4d 2f 31 2e 3a 2e 31 20 47 4e 55 54 4c 53  -MM/1.4. 1 GNUFLS
0180  2f 33 2e 35 2e 31 37 0d 6a 6d 6a  /3.5.17. ...
  
```

henry@henry-ThinkPad-X250 -  
 Paket: 662 - Angezeigt: 12 (1.8%)  
 ProfilDefault

## Some Wireshark Display Filter Examples

- `tcp.port eq 25 or icmp`  
Show only SMTP (port 25) and ICMP traffic
- `ip.src==192.168.0.0/16 and ip.dst==192.168.0.0/16`  
Show only traffic in the IP network `192.168.x.x`
- `ip.addr == 10.0.0.142`  
Show only traffic that has a specific IP address inside (sender or destination)
- `ip.src == 10.0.0.142`  
Show only traffic, send from the device with IP address `10.0.0.142`
- `ip.dst == 10.0.0.142`  
Show only traffic, send to the device with IP address `10.0.0.142`
- `ip.src == 10.0.0.139 || ip.src == 10.0.0.142`  
Show only traffic, send from device `10.0.0.139` or `10.0.0.142`
- `tcp.dstport == 80`  
Show communication to TCP port number 80
- `( ip.src == 10.0.0.142 ) && ( tcp.dstport == 80 )`  
Show only traffic, send from device `10.0.0.142` to TCP port number 80

## Configuration of the machines

Please follow these rules:

- **Make your configurations statically! Use the tool `ip` exclusively!**
- **Save your static configuration on file! Use an USB-Drive for the extraction!**
- **Test your setup! Document it accurately! Demonstrate it in the lab exercise!**
- **Create slides of your configurations! Use the command-line snippets, screenshots and Wireshark captures for your documentation!**

### Non persistent configuration on machines

Please be aware, that the configurations on the machines are static and will be deleted after a reboot! Make sure to save your progress on an external drive!

# Contents

Exercise 1

Networking in Linux

Wireshark

**Some useful commands**

# Some useful commands

## Execute a command with super user rights:

```
sudo <COMMAND>
```

## Led the LED of the NIC blink:

```
ethtool -p <INTERFACE>
```

## Show configurable kernel parameters:

```
sysctl -a
```

## Enable forwarding of IPv4 packets:

```
sysctl -w net.ipv4.ip_forward=1
```

## Enable forwarding of IPv6 packets:

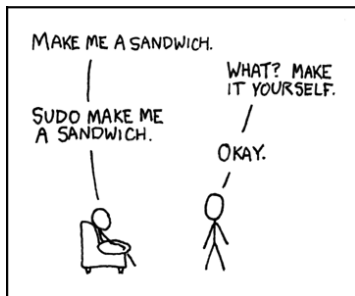
```
sysctl -w net.ipv6.conf.all.forwarding=1
```

## Allow for ping to an IPv4 broadcast address:

```
sysctl -w net.ipv4.icmp_echo_ignore_broadcasts=0
```

## Enable IPv6 for all interfaces:

```
sysctl -w net.ipv6.conf.all.disable_ipv6=0
```



# Management of Services

Services (daemons) in most modern Linux systems are controlled via **systemd**. The according tool is called `systemctl`.

## Starting a service:

```
systemctl start <SERVICE>
```

## Stopping a service:

```
systemctl stop <SERVICE>
```

## Restarting a service:

```
systemctl restart <SERVICE>
```

## Requesting the status of a service:

```
systemctl status <SERVICE>
```

## Read the log entries for a given service:

```
journalctl -u <SERVICE>
```