FRANKFURT UNIVERSITY OF APPLIED SCIENCES

Practical Computer Networks and Applications

Exercise 1 – IP Version 4 Networks

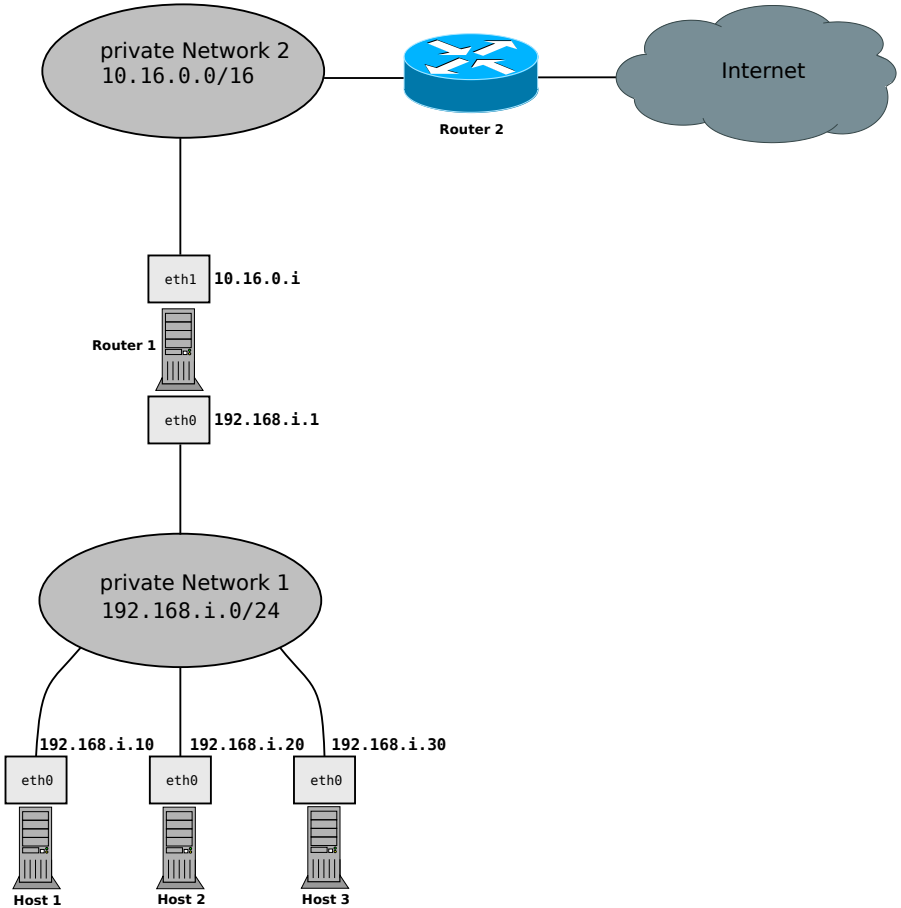
Prof. Dr. Baun, Prof. Dr. Ebinger, Prof. Dr. Hahm, Prof. Dr. Kappes, Dipl. Inf. (FH) Maurizio Petrozziello

{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

Network Topology – Exercise 1



Network Topology of lab exercise 1

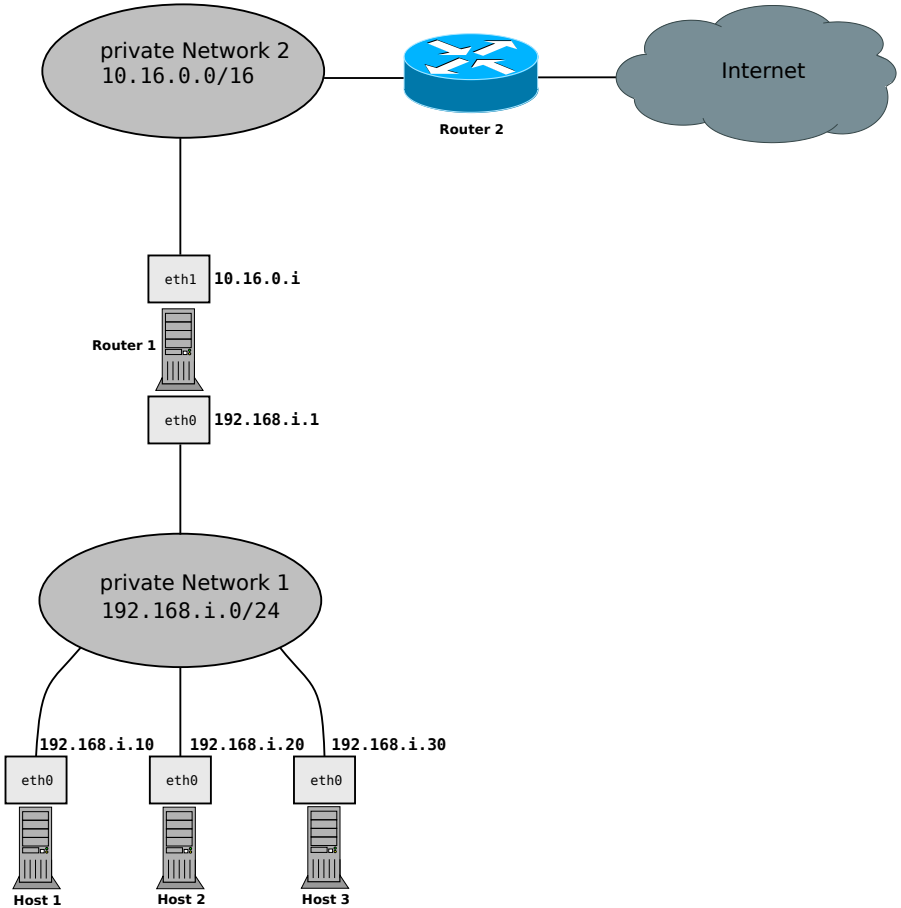
Private Network 1:

- `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

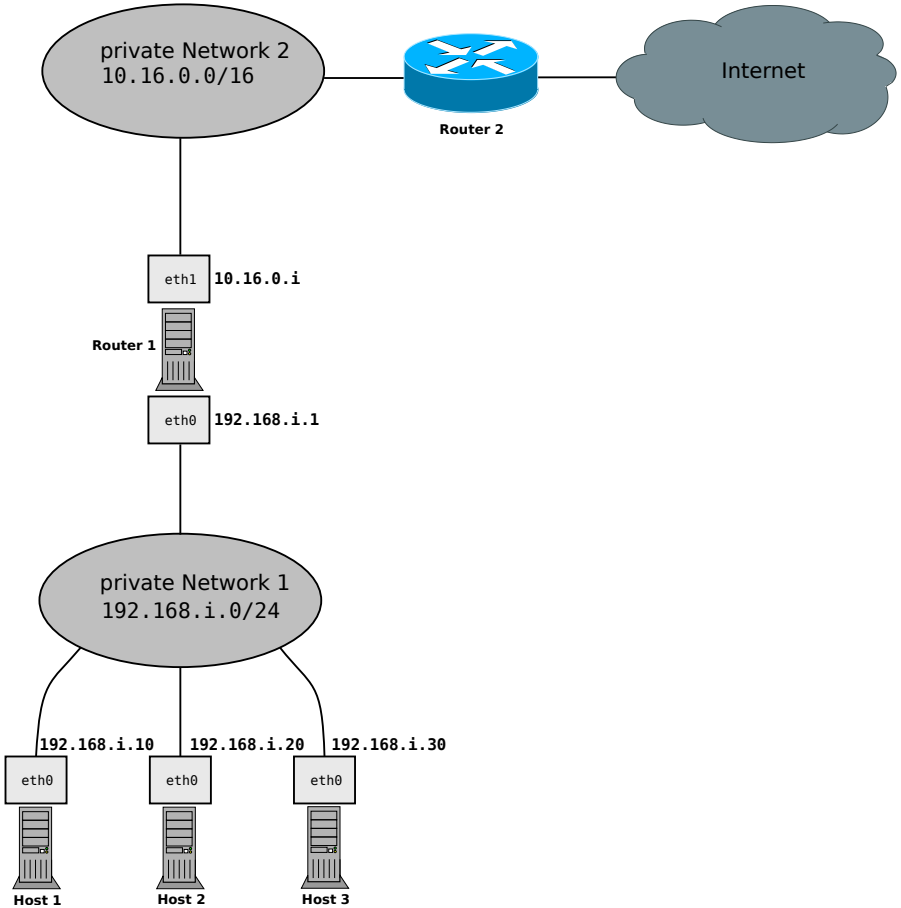
Private Network 2:

- `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

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![1]

[1] 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![2]
- 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!

[2] 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`[3] command:

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

[3] The manpage of `ip` gives you the full list of functions and options!

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

The `ping`[4] 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

[4] 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

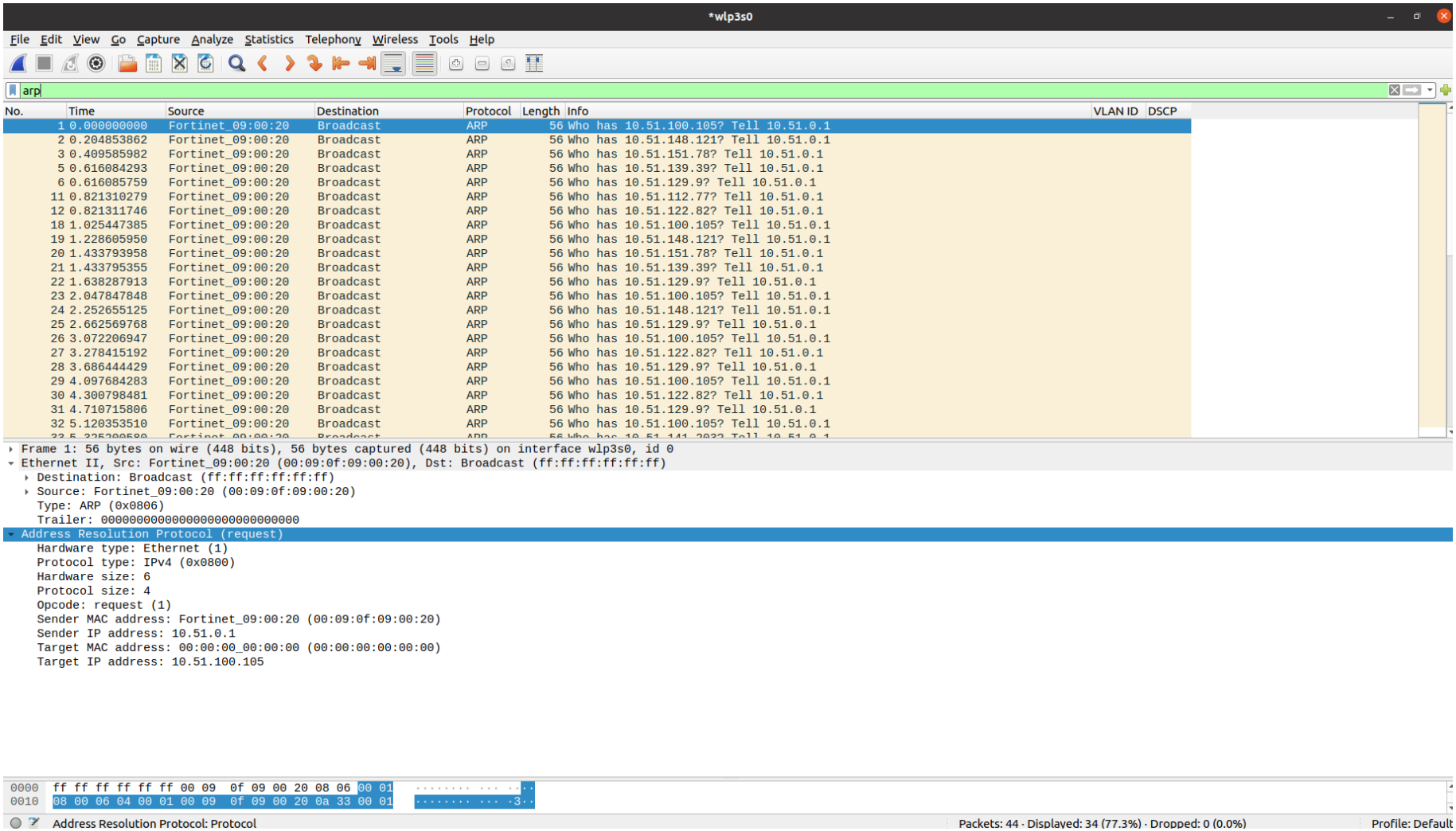
Wireshark

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

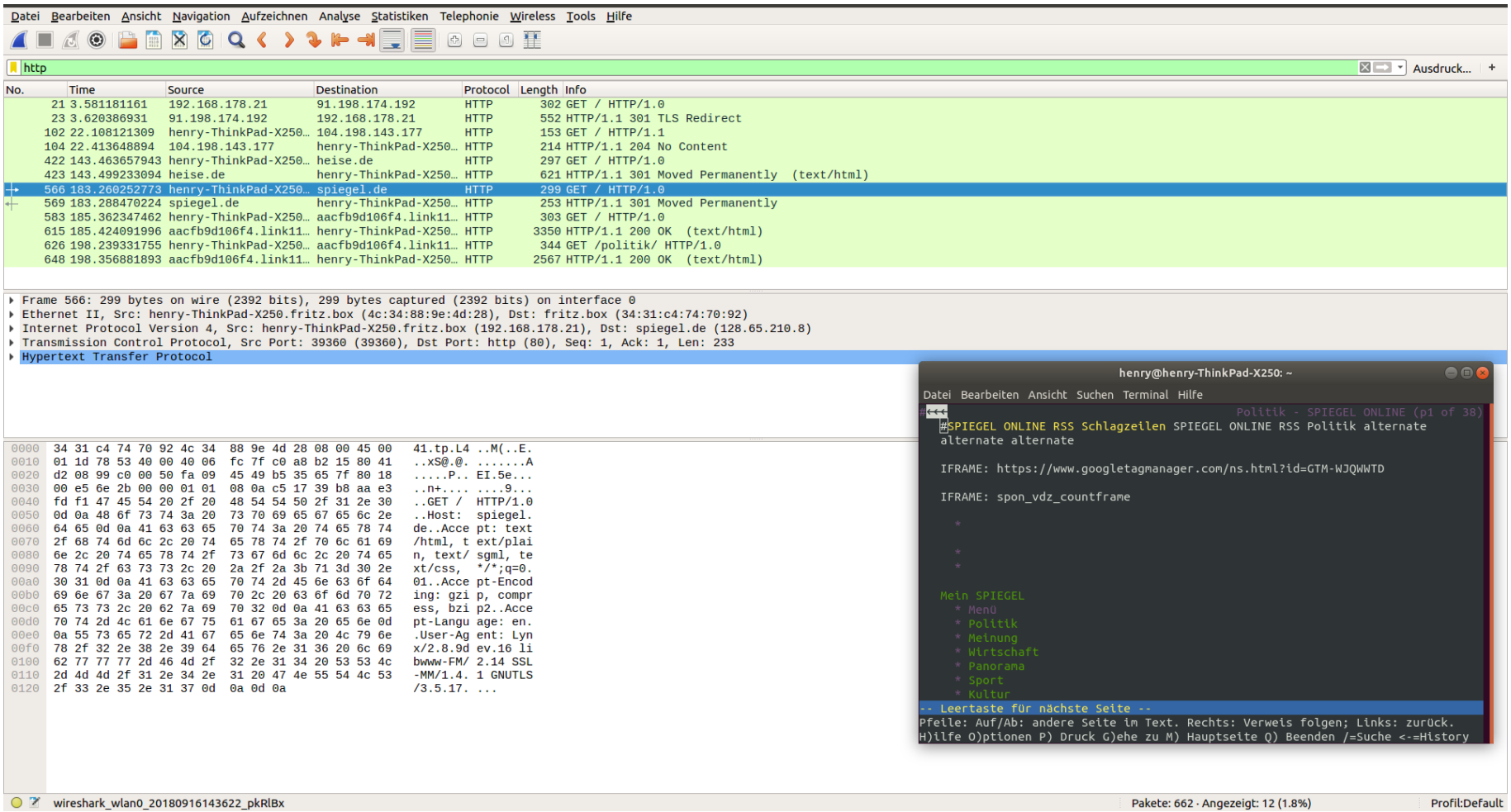
Wireshark



Wireshark Desktop

An Example on Using Wireshark

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



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

Some useful commands

Execute a command with super user rights:
```
sudo <COMMAND>
```

Led the LED of the NIC blink:
```
ethtool -p <INTERFACE>
```

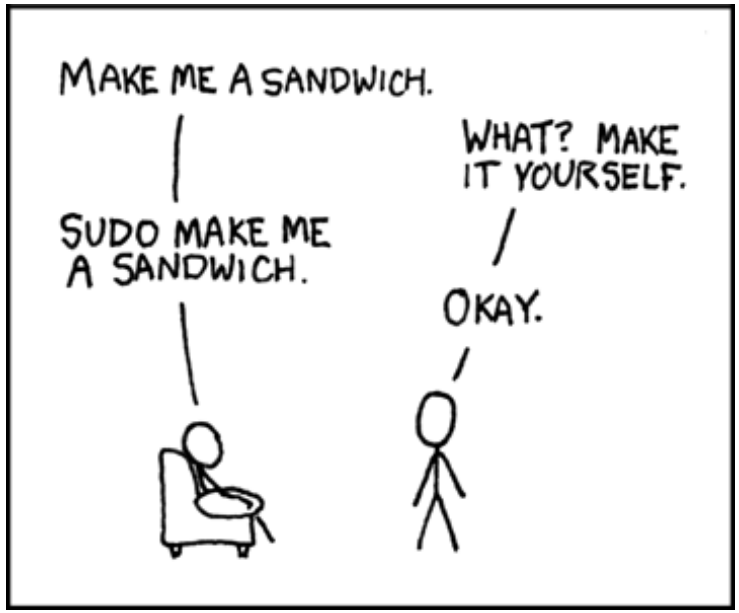
Show configurable kernel parameters:
```
sysctl -a
```

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 pinging a 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>
```