



**University of Stuttgart**  
Institute of Parallel and Distributed Systems



**Distributed  
Systems**

Fachpraktikum / Lab-Course

# **Software-Defined and Time-Sensitive Networking**

Tutorial: Networking Basics  
Part 1: Networking in Linux

Frank Dürr

**Summer  
Term  
2023**

# Agenda

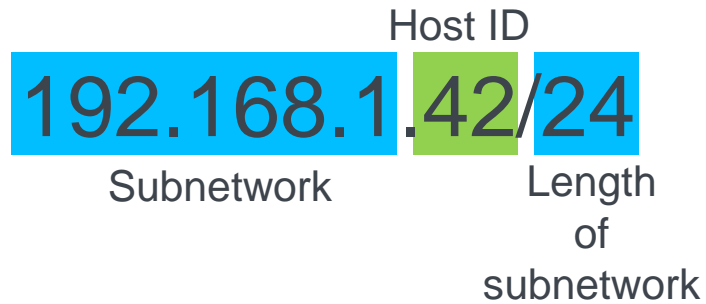
- IP addresses
  - Notation, subnetworks
  - Address configuration of network interfaces in Linux
- Network namespaces in Linux
  - Creating namespaces
  - Connecting namespaces
- Network applications
  - netcat
  - cURL
- Monitoring network traffic
  - tcpdump
  - tshark

# IP Addresses

- Addresses of **network layer**
- Used for **routing** packets between hosts
- **Assigned to network interfaces of hosts and routers**
  - **Multi-homed hosts** with several network interfaces might be reachable through different IP addresses
    - Example: laptop with WiFi interface and Ethernet interface
- **IPv4** (32 bit) and **IPv6** (128 bit) addresses used in Internet today
  - We will use IPv4 for readability reasons

# IP Addresses

## IPv4 Address Notation



# IP Addresses

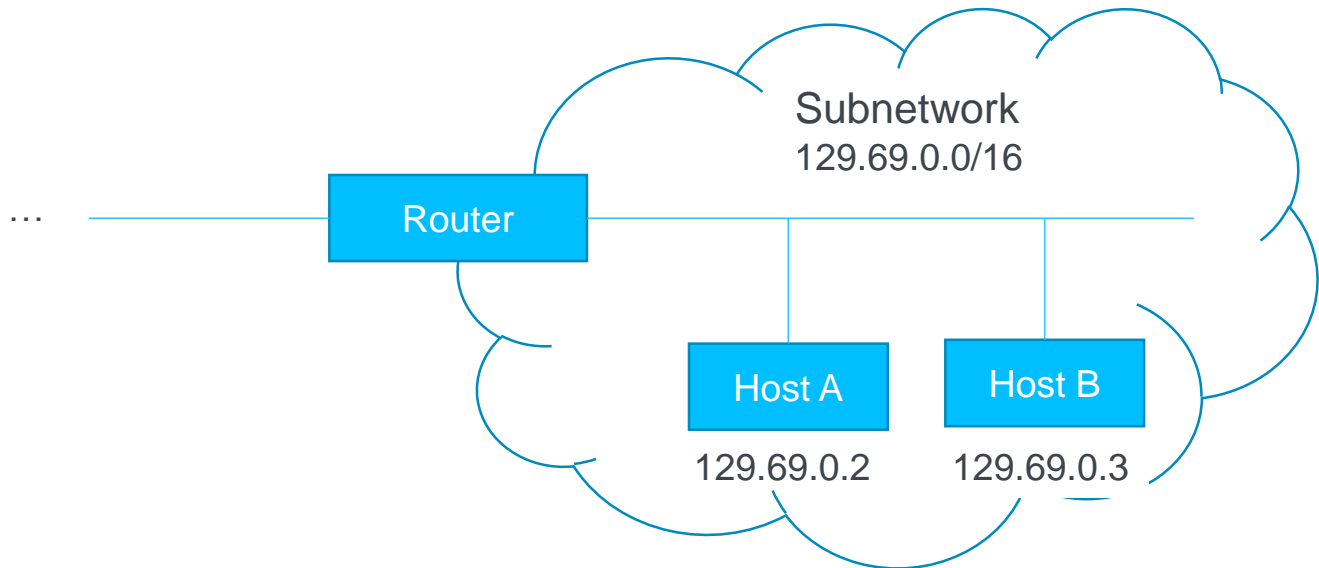
## IPv4 Address Notation

192.168.1.42

Subnet mask: 255.255.255.0

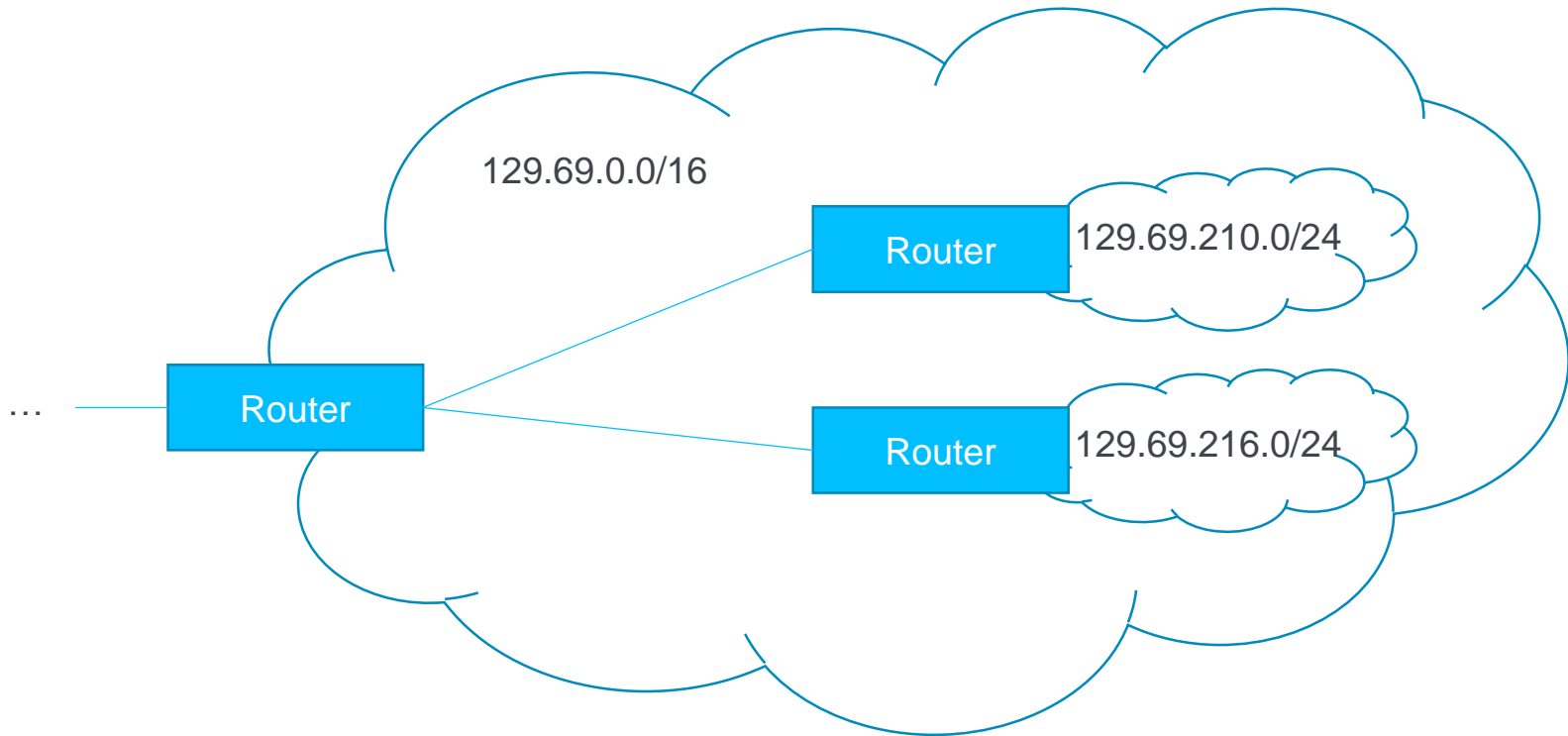
# IP Addresses

## Subnetworks



# IP Addresses

## Subnetworks



# IP Addresses

## Public and Private IP Addresses

- **Public IP address**

- Globally unique and globally routable in Internet
- Example: 129.69.210.42

- **Private IP addresses**

- Only unique within private network, only routed within private network
  - Connection to Internet requires network address translation at gateway (NAT)
- Same addresses can be reused between organizations
- Reserved private IP address ranges for IPv4:
  - 10.0.0.0/8
  - 192.168.0.0/16
  - 172.16.0.0/12



# IP Addresses

## Public and Private IP Addresses

In the lab:

- Use only private IP addresses from 10.0.0.0/8
  - In particular 192.168.0.0/16 is used already within IPVS networks
- Can create smaller subnetworks, e.g., 10.x.0.0/16

# IP Addresses

## Special IP Addresses

- Localhost: 127.0.0.1
- Broadcast to all hosts in subnet: all host bits 1
  - Example: 192.168.1.255 for broadcast in subnet 192.168.1.0/24
- Multicast group addresses: 224.0.0.0 – 239.255.255.255
- Bind to all network interfaces on this host (INADDR\_ANY): 0.0.0.0

# IP Addresses

## Configuration of Network Interfaces in Linux

Assign IP address 192.168.1.1/16 to network interface eth0:

```
$ ip address add 192.168.1.1/16 dev eth0
```

Delete IP address 192.168.1.1/16 from network interface eth0:

```
$ ip address del 192.168.1.1/16 dev eth0
```

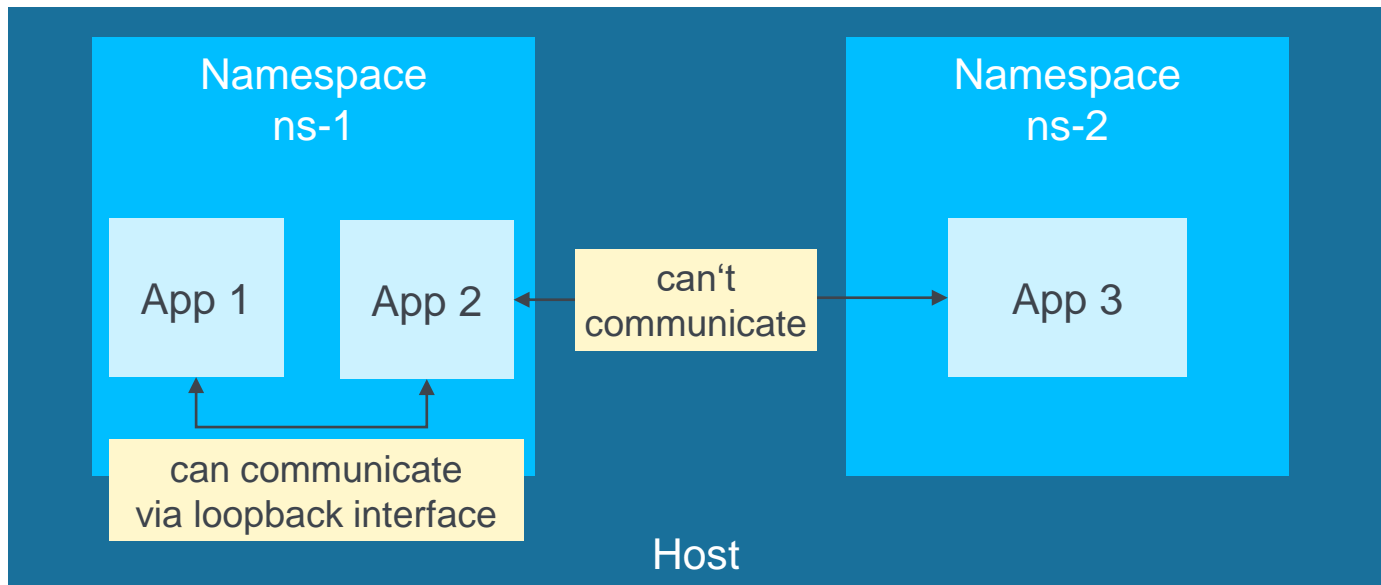
Note: before using the network interface, you have to bring it up:

```
$ ip link set eth0 up
```

# Network Namespaces

Goal: Isolation of virtual network environments on a host

- Used, for instance, to implement containerized apps
- In lab used to implement virtual network environments on single host



# Network Namespaces

## Namespaces in Linux

### Creating namespace ns-1

```
$ ip netns add ns-1
```

### Listing all namespaces

```
$ ip netns list
```

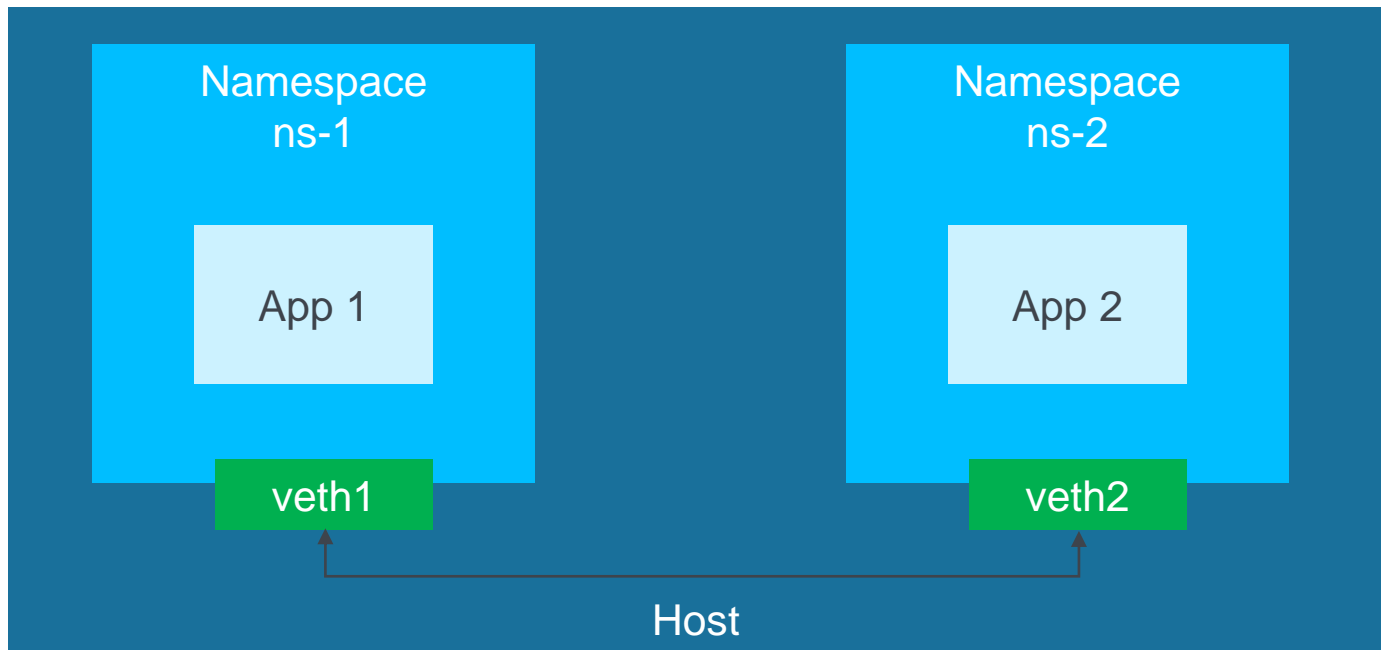
### Executing a command (here an interactive shell) in namespace ns-1

```
$ ip netns exec ns-1 /bin/bash
```

# Network Namespaces

## Communication through Virtual Ethernet Devices

Communication through pair of connected veth devices



# Network Namespaces

## Virtual Ethernet Devices in Linux

Create virtual Ethernet devices veth1 and veth2:

```
$ ip link add veth1 type veth peer name veth2
```

Assign veth1 device to a namespace ns-1 (same for veth2):

```
$ ip link set veth1 netns ns-1
```

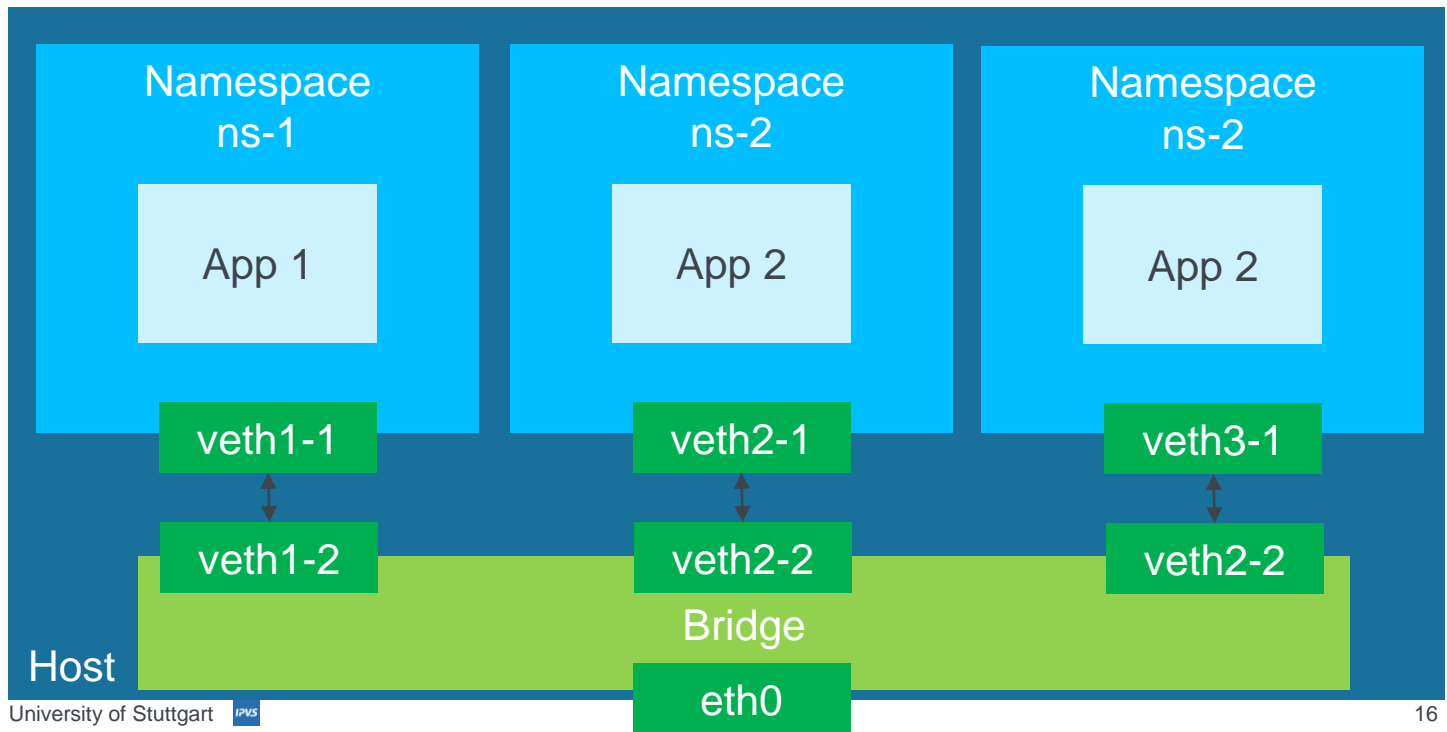
Notes:

- When a namespace is deleted, all assigned devices will return to the default namespace

# Network Namespaces

## Communication through Virtual Bridge

- Bridge connects several namespaces on same host
- Bridge connects namespaces to other hosts via physical Ethernet





# Network Namespaces

## Virtual Bridges in Linux

Create virtual bridge vbridge and bring it up:

```
$ ip link add name vbridge type bridge
```

```
$ ip link set dev vbridge up
```

Assign virtual Ethernet device veth1 to bridge vbridge  
(same for physical devices):

```
$ ip link set veth1 master vbridge
```

# Network Applications

## Netcat

Swiss army knife for communication

- Simple client/server applications without programming
- Integration with scripts
- Supports TCP, UDP, UNIX domain sockets, IPv4 & IPv6, etc.

# Network Applications

## Simple TCP Server with netcat

listen on port 5555 on all  
network interfaces

```
$ nc -k -l 5555
```

keep listening for other connections  
when one connection is finished

# Network Applications

## Simple TCP Client with netcat

`$ echo "hello" | nc 192.168.1.1 5555`

IP address of server

port of server

nc reads data to be sent from stdin

# Network Applications

## Further Useful netcat Options

- UDP: `-u`
- Timeout idle connections after 5 seconds: `-w 5`
- Set source port 6666: `-s 6666`
- Use only IPv4: `-4`

# Network Applications

## cURL

Transferring data to/from server using one of many protocols:

DICT, FILE, FTP, FTPS, GOPHER, GOPHERS, HTTP, HTTPS, IMAP, IMAPS, LDAP, LDAPS, MQTT, POP3, POP3S, RTMP, RTMPS, RTSP, SCP, SFTP, SMB, SMBS, SMTP, SMTPS, TELNET or TFTP

Download HTML page from web server of University of Stuttgart via HTTPS:

```
$ curl https://www.uni-stuttgart.de
```

# Monitoring Network Traffic

## tcpdump

- Record network traffic in promiscuous mode from network interface
  - Does not replace server application (connection would be refused)
- Record to file in pcap format
  - Can be read by other tools such as tshark
- Filters to filter relevant traffic

# Monitoring Network Traffic

## tcpdump example

network interface to capture from

filter for IPv6 packets

```
$ tcpdump -i eth0 -w file.pcap proto ipv6
```

write trace to file

The diagram shows the command `$ tcpdump -i eth0 -w file.pcap proto ipv6` with three annotations. A line from 'network interface to capture from' points to `-i eth0`. A line from 'filter for IPv6 packets' points to `proto ipv6`. A line from 'write trace to file' points to `-w file.pcap`. The arguments `-i eth0`, `-w file.pcap`, and `proto ipv6` are highlighted in blue in the original image.



# Monitoring Network Traffic

## Wireshark / tshark

- Another tool for capturing packets
- Wireshark: interactive GUI for controlling capturing, viewing traces, applying filters, etc.
- tshark: terminal (command line) version of Wireshark
- Can also be used to capture from interface or filter recorded pcap files after capturing

# Monitoring Network Traffic

Wireshark / tshark example

Interface to capture from

print values of extracted fields

```
$ tshark -i eth0 -Y http.request -T fields -e http.user_agent
```

filter for HTTP requests

extract the user agent from the request

The diagram illustrates the components of the tshark command. The command is shown as '\$ tshark -i eth0 -Y http.request -T fields -e http.user\_agent'. Each option is highlighted in a blue box. Annotations with lines pointing to these boxes explain their function: '-i eth0' is the interface to capture from; '-Y http.request' is the filter for HTTP requests; '-T fields' is used to print values of extracted fields; and '-e http.user\_agent' is used to extract the user agent from the request.

# Questions?