Netzwerk Teil 1

Linux-Kurs der Unix-AG

Andreas Teuchert

5. Januar 2015

UNIX AG
TU Kaiserslautern

RH Regionales Hochschul-Rechenzentrum Kaiserslautern RK

Wiederholung: OSI-Schichtenmodell

- Layer 1: Physical Layer (Kabel, Funk)
- Layer 2: Data Link Layer (Ethernet, WLAN)
- Layer 3: Network Layer (IP)
- Layer 4: Transport Layer (TCP, UDP)
- Layer 5-6: (in diesem Kurs nicht von Interesse)
- Layer 7: Application Layer (HTTP, SMTP, SSH)

Übersicht: Netzwerk-Werkzeuge unter Linux

- Layer 1: `ethtool`: Zustand der Netzwerkkarte abfragen und ändern
- Layer 2:
  - `brctl`: Bridges anlegen und verwalten
  - `ifenslave`: Port-Aggregation
  - `ip link`: Netzwerk-Interfaces verwalten
- Layer 3:
  - `iptables`: Paketfilter (Layer 3+)
  - `ip`: IP-Adressen, -Routen und ARP-Tabelle verwalten (früher `ifconfig`, `route`, `arp`)

ifup/ifdown, /etc/network/interfaces

- unter Debian/Ubuntu: Einstellungen für Netzwerk-Interfaces in der Datei `/etc/network/interfaces`
- IP-Adresse, Netzmaske, Gateway oder ob DHCP verwendet wird
- Interfaces können beim Hochfahren automatisch entsprechend konfiguriert werden
- manuelles Ein-/Ausschalten von Interfaces mit `ifup/ifdown`

/etc/network/interfaces – Beispiel

```
1  auto lo
2  iface lo inet loopback
3
4  auto eth0
5  iface eth0 inet static
6    address 192.0.2.4
7    netmask 255.255.255.192
8    gateway 192.0.2.62
9
10 iface eth0 inet6 static
11   address 2001:db8:1def:1c5::4
12   netmask 64
13
14 iface eth1 inet dhcp
```

ethtool

- `ethtool eth0` zeigt den Status des Interfaces an
- u. a.: Verbindung vorhanden, Geschwindigkeit, Half-/Full-Duplex
- kann auch Eigenschaften des Interfaces manipulieren:
  - Wake-On-LAN
  - Interface identifizieren (z. B. durch Blinken einer LED)
  - Geschwindigkeit und Autonegotiation festlegen
  - ...
- nützlich beim Untersuchen von Problemen (richtige Geschwindigkeit? Autonegotiation an?)

brctl

- Bridge: Verbindung von Netzwerkinterfaces auf Layer 2
- funktioniert wie ein einfacher Switch
- Weiterleitung nur in Software, daher langsamer als echte Switche
- ankommende Frames werden anhand der ARP-Tabelle weitergeleitet
- mit `ebtables` auch Filter möglich
- unterstützt optional Spanning-Tree (nur 802.1D)
- häufigste Anwendung: Virtualisierung

brctl – Beispiel

```
# brctl addbr br0
# brctl addif br0 eth0
# brctl addif br0 eth1
# ip link set eth0 up
# ip link set eth1 up
# ip link set br0 up
# brctl show br0
bridge name   bridge id       STP enabled   interfaces
br0           8000.0021...    no            eth0
                                            eth1
```

Bridges – /etc/network/interfaces

- Bridges können auch über `/etc/network/interfaces` konfiguriert werden
- Vorteil: Automatisches Anlegen beim Hochfahren

```
1 auto br0
2 iface br0 inet manual
3   bridge_ports eth0 eth1
4   bridge_stp off
5   bridge_fd 0
6   bridge_hello 1
```

VLANs

- VLANs (Virtual LAN) werden verwendet um ein physikalisches Netz in mehrere virtuelle Netze aufzuteilen
- auf Switchen können Ports zu VLANs zugeordnet werden
- mehrere VLANs auf einem Port mit Trunks nach 802.1Q
- Anwendung: gleiche VLANs auf mehreren Switchen oder Verbindung zu Router
- wichtigste Anwendungen unter Linux:
  - Virtualisierung: auf einem Host virtuelle Maschinen in verschiedenen VLANs
  - Linux-Rechner als Router im SOHO-Bereich

VLANs anlegen

- Konfiguration von VLANs unter Linux mit `ip link`
- Subinterface für VLAN 5 auf eth0 anlegen:

```
1 # ip link add link eth0 name vlan5 type vlan id 5
2 # ip link show vlan5
3 354: vlan5@eth0: <BROADCAST,MULTICAST> mtu 1500
    qdisc noop state DOWN mode DEFAULT
4     link/ether 00:21:cc:67:7d:0f brd ff:ff:ff:ff:ff:
        ff
5 # ip link del vlan5
6 # ip link show vlan5
7 Device "vlan5" does not exist.
```

VLANs – /etc/network/interfaces

```
1 iface vlan5 inet static
2   vlan-raw-device eth0
3   address 192.0.2.5
4   netmask 255.255.255.240
```

- erzeugt das Interface `vlan5`
- Paket `vlan` muss installiert sein

Lab 7.1: VLANs

- weiteres Netzwerk-Interface zur VM hinzufügen
- Subinterface für VLAN 7 anlegen
- andere VMs über VLAN 7 pingen

Port-Aggregation

- mehrere Interfaces zu einem logischen Interface zusammenfassen
- auch als Bonding, Trunking oder Teaming bekannt
- Ausfallsicherheit
- höhere Performance
- unterstützt Link Aggregation Control Protocol (LACP, 802.3ad)
- Verbindung zur Nicht-Linux-Geräten, z. B. Switchen

Port-Aggregation – Lastverteilung

- verschiedene Möglichkeiten zum Verteilen der Last, u. a.:
- Round Robin: Frames abwechselnd über die Interfaces senden
- Active-Backup: Immer nur ein Interface verwenden
- Broadcast: Frames über alle Interfaces gleichzeitig versenden
- Hashing: Aufteilung anhand von MAC- oder IP-Adressen oder TCP- oder UDP-Port-Nummern
- Einfluss auf Performance und Ausfallsicherheit
- auch abhängig vom Gerät auf der Gegenseite

ifenslave – Beispiel

```
1 # modprobe bonding
2 # ip link set bond0 up
3 # ifenslave bond0 eth0 eth1
4 # ip link sh bond0
5 350: bond0: <BROADCAST,MULTICAST,MASTER,UP,LOWER_UP>
     mtu 1500 qdisc noqueue state UP mode DEFAULT
6      link/ether 16:fb:21:d6:aa:ff brd ff:ff:ff:ff:ff:
       ff
7 # ip link sh eth0
8 5: eth0: <NO-CARRIER,BROADCAST,MULTICAST,SLAVE,UP>
   mtu 1500 qdisc noqueue master bond0 state DOWN
   mode DEFAULT
9      link/ether 16:fb:21:d6:aa:ff brd ff:ff:ff:ff:ff:
       ff
```

Port-Aggregation – /etc/network/interfaces

```
auto bond0
iface bond0 inet dhcp
  bond_slaves eth0 eth1
  bond_mode 802.3ad
  bond_miimon 100
  bond_xmit_hash_policy layer3+4
  bond_lacp_rate slow
```