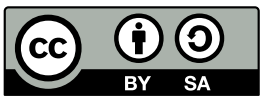
# Netzwerk Teil 1

Zinching Dang

27. Mai 2015



## 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)



Das OSI-Schichtenmodell unterteilt Netzwerke in funktionelle Schichten. Diese Unterteilung findet sich auch in den Netzwerk-Werkzeugen unter Linux wieder. Auch beim Konfigurieren und Troubleshooting von Netzwerken ist es sinnvoll, „von unten nach oben“ vorzugehen, da z. B. ein Austausch von IP-Paketen ohne physikalische Verbindung nicht möglich ist.

## 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`)



In diesem Kurs werden die wichtigsten Netzwerk-Werkzeuge unter Linux behandelt. Manche Werzeuge sind klar einer Netzwerkschicht zugeordnet, z. B. `ethtool`, bei anderen ist keine Zuordnung zu nur einem Layer möglich. Das Tool `ip` kann z. B. sowohl genutzt werden um Netzwerk-Interfaces ein- und auszuschalten (Layer 2) als auch um IP-Adressen auf diesen Interfaces zu konfigurieren (Layer 3). Der Paketfilter `iptables` filtert zwar IP-Pakete, kann aber auch Informationen aus höheren Schichten, wie Portnummern mit einbeziehen.

## 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`



Da Netzwerkeinstellungen unter Linux beim Herunterfahren verloren gehen, werden Programme benötigt, die diese beim Hochfahren automatisch wieder herstellen. Unter Debian und Debian-Derivaten wie Ubuntu wird dazu `ifup` und `ifdown` verwendet. Diese Programme laden die Interface-Einstellungen aus der Datei `/etc/network/interfaces` und konfigurieren die Interfaces entsprechend. Mit den entsprechenden Konfigurations-Optionen geschieht dies automatisch beim Hochfahren.

Mit `ifup <interface>` können Interfaces auch manuell konfiguriert werden. Mit dem Programm `ifdown` können sie dekonfiguriert werden. So sind auch ohne Neustart Konfigurationsänderungen möglich.

Vorsicht ist beim Ändern der Netzwerk-Konfiguration von nur über SSH erreichbaren Servern geboten. Die Änderungen treten sofort in Kraft und können bei Fehlern in der Konfiguration zu Nicht-Erreichbarkeit des Servers führen. Bei einer fehlerhaften `/etc/network/interfaces` können diese Probleme auch durch einen Neustart nicht behoben werden.

## /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
```



Das Beispiel zeigt eine typische Netzwerk-Konfiguration eines Rechners mit statischer IP-Adresse. Es sind die Netzwerk-Interfaces `lo`, `eth0` und `eth1` konfiguriert.

Das Interface `lo` ist auf jedem Rechner, sogar wenn dieser an kein Netzwerk angeschlossen ist, notwendig. Dieses Loopback-Interface wird von vielen Programmen zu Kommunikation mit anderen Programmen auf dem gleichen Rechner verwendet. Durch die Zeile `auto lo` wird das Interface automatisch beim Hochfahren konfiguriert. Eine IP-Adresse ist nicht angegeben, da für Loopback-Interfaces automatisch die IP-Adresse 127.0.0.1 bzw. ::1 verwendet wird.

Das Interface `eth0` ist mit einer statischen IPv4- und IPv6-Adresse konfiguriert. Das Default-Gateway des Rechners ist auch über `eth0` zu erreichen. Auch dieses Interface wird beim Hochfahren automatisch aktiviert.

Das Interface `eth1` wird automatisch über DHCP konfiguriert. Allerdings wird es beim Hochfahren nicht automatisch aktiviert sondern muss manuell über `ifup eth1` gestartet werden.

## 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?)



Mit dem Programm `ethtool` können Status-Informationen zu Netzwerk-Interfaces abgefragt werden. Bei Ethernet-Interfaces kann angezeigt werden, ob eine Verbindung vorhanden ist (`Link detected`) und mit welcher Geschwindigkeit die Verbindung konfiguriert wurde (`Speed`).

Beim Untersuchen von Netzwerk-Problemen ist `ethtool` sehr nützlich, da es benutzt werden kann um falsche Geschwindigkeits- und Autonegotiation-Einstellungen zu ermitteln. Diese können zu Verbindungsproblemen und reduzierter Übertragungsgeschwindigkeit führen.

Mit `ethtool` können je nach Netzwerkkarte auch erweiterte Interface-Einstellungen, wie Wake-On-LAN konfiguriert werden. Nützlich ist es manchmal auch, eine LED an der Netzwerkkarte blinken zu lassen, um so den gewünschten Server in einem Rack zu finden.

## 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



Eine Bridge ermöglicht es, mehrere Netzwerk-Interfaces auf Layer 2 zu verbinden, sodass Ethernet-Frames zwischen den an den Interfaces angeschlossenen Netzwerken ausgetauscht werden können. Bridges unter Linux arbeiten ähnlich wie einfache Ethernet-Switche, sind aber in Performance und Funktionsumfang nicht mit dedizierten Switchen vergleichbar.

Mit dem Programm `ebtables` kann die Kommunikation zwischen den Bridge-Ports eingeschränkt werden, sodass von professionellen Switchen bekannte Sicherheitsfeatures auch unter Linux realisiert werden können.

Die Linux-Bridge-Implementierung unterstützt auch das Spanning-Tree-Protokoll, allerdings nur in der nicht mehr zeitgemäßen Variante 802.1D. Allerdings lassen sich Bridging-Loops unter Linux auch leichter vermeiden, da Bridges üblicherweise nicht miteinander verbunden werden.

Die häufigste Anwendung von Bridges unter Linux ist es, virtuellen Maschinen Netzwerkzugriff zu ermöglichen oder diese zumindest untereinander zu verbinden. Verwaltungswerkzeuge wie libvirt können VMs automatisch an vorhandene Bridges anschließen und teilweise auch neue Bridges anlegen.

## 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
```



Das Beispiel zeigt, wie eine neue Bridge, `br0`, angelegt wird. Die Interfaces `eth0` und `eth1` werden mit dieser Bridge verbunden.

Zur Nutzung der Bridge müssen die drei Interfaces noch aktiviert werden. Dazu wird hier das Werkzeug `ip link` verwendet. Dieses wird in Teil 2 näher erläutert.

Mit dem Befehl `brctl show br0` wird dann der Status der Bridge angezeigt. Es wird der Name und die Bridge-ID angezeigt. Außerdem ist ersichtlich, dass das Spanning-Tree-Protokoll standardmäßig nicht aktiv ist. In der letzten Spalte werden die mit der Bridge verbundenen Interfaces angezeigt.

Beim Anschluss dieses Rechners an ein bestehendes Netzwerk sollte mit Vorsicht vorgegangen werden, da durch das abgeschaltete STP die Gefahr besteht, eine Loop zu erzeugen, die durch Broadcast-Stürme zu größeren Netzwerkausfällen führen kann.

## 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
```



In diesem Beispiel wurde die auf der vorherigen Folie gezeigte Konfiguration über `/etc/network/interfaces` realisiert. Dadurch wird die Bridge beim Hochfahren automatisch angelegt.

Dazu muss das Paket `bridge-utils` installiert sein.

Zusätzlich sind noch drei das Spanning-Tree-Protokoll betreffende Einstellungen getätigt worden. Wie schon im ersten Beispiel ist auch hier STP nicht aktiv. Zusätzlich wurde das Forward-Delay deaktiviert, sodass die Bridge sofort nach Anschluss eines Interfaces Frames weiterleitet. Die Einstellung der Hello-Time hat durch das deaktivierte STP in diesem Fall keine Bedeutung.

Beim Anschluss an ein bestehendes Netzwerk können falsche STP-Einstellungen zu Instabilitäten und Ausfällen führen. Im Zweifel sollte der zuständige Netzwerk-Ingenieur konsultiert werden.

## 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



Aus Gründen der Sicherheit und Robustheit werden Netzwerke häufig in VLANs unterteilt. Dadurch kann der Netzwerkzugriff von angeschlossenen Rechner begrenzt und Auswirkungen von Fehlkonfigurationen (z. B. Broadcast-Stürme) verringert werden.

VLANs werden üblicherweise auf Switchen konfiguriert. Dort werden dann die Switchports VLANs zugeordnet. Sollen VLANs zwischen zwei Geräten (z. B. zwei Switchen oder einem Switch und einem Router) verbunden sein, kommen Trunks nach 802.1Q zum Einsatz, über die mehrere VLANs transportiert werden können.

Unter Linux wird die 802.1Q-Unterstützung vornehmlich verwendet um auf Virtualisierungs-Hosts VMs in mehreren VLANs anlegen zu können oder um im SOHO-Bereich einen Linux-Rechner als Router zu verwenden, der dafür mit allen VLANs verbunden sein muss.

## 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.
```



Unter Linux können VLANs mit dem Werkzeug `ip link` konfiguriert werden. In dem Beispiel ist das Interface `eth0` mit einem Trunk verbunden, auf dem das VLAN 5 vorhanden ist. Für dieses VLAN wird das Subinterface `vlan5` angelegt. Die Ausgabe von `ip link show` zeigt an, dass es sich bei dem Interface um ein Subinterface von `eth0` handelt.

So können auf einem physikalischen Interface mehrere Subinterfaces in verschiedenen VLANs konfiguriert werden. Mit dem Befehl `ip link del` lassen sich diese wieder löschen.

## 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



In diesem Beispiel ist das Subinterface `vlan5` über `/etc/network/interfaces` konfiguriert worden. Es wurde auch eine IP-Adresse auf dem Subinterface konfiguriert.

Um VLANs über `/etc/network/interfaces` konfigurieren zu können, muss unter Debian das Paket `vlan` 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



Mit Port-Aggregation können mehrere physikalische Interfaces zu einem logischen Interface zusammengefasst werden. Dadurch kann eine höhere Ausfallsicherheit und Performance erreicht werden.

Die Linux-Bonding-Implementierung unterstützt auch das Link Aggregation Control Protocol, das zur Verbindung mit Nicht-Linux-Systemen, z. B. Switchen benötigt wird.

## 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



Es werden verschiedene Möglichkeiten zur Lastverteilung unterstützt. Die richtige Einstellung hängt von den Anforderungen an Performance und Ausfallsicherheit, sowie dem Gerät auf der Gegenseite ab.

So wird mit Round Robin zwar eine gute Last-Verteilung erreicht, da beide Interfaces gleichmäßig verwendet werden, allerdings steht nach dem Ausfall eines Interfaces nur noch die halbe Bandbreite zur Verfügung, was je nach Umgebung zu Problemen führen kann. Außerdem kann Round Robin zu Problemen mit TCP-Verbindungen führen, da sich die Reihenfolge der Pakete ändern kann, was bei TCP erhebliche Performance-Einbußen bewirkt.

Im Gegensatz dazu steht der Active-Backup-Modus. Dieser verwendet stets nur ein Interface. Fällt dieses aus, wird eines der Ersatz-Interfaces verwendet. Dadurch wird zwar nur die Bandbreite eines Interfaces genutzt, diese ist aber stets konstant.

Die primitivste Variante der Last-Verteilung ist Broadcast, bei der Frames über alle Interfaces gleichzeitig versendet werden. Es werden zwar alle Interfaces verwendet, aber es steht trotzdem nur die Bandbreite eines Interfaces zur Verfügung.

Active-Backup und Broadcast können auch ohne Unterstützung durch den Switch verwendet werden.

Des weiteren stehen noch diverse Möglichkeiten zur Verfügung, die Last anhand der MAC- oder IP-Adressen (oder der Port-Nummern) über Hashing-Verfahren aufzuteilen. Dazu wird eine geeignete Gegenstelle benötigt. Hier empfiehlt sich die Verwendung von LACP.

Weitere unterstützte Verfahren sind in `/usr/share/doc/linux-doc-3.2/Documentation/networking/bonding.txt.gz` beschrieben. Diese Datei ist im Paket `linux-doc-3.2` enthalten.

## ifenslave – Beispiel

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



In diesem Beispiel werden die Interfaces `eth0` und `eth1` zum logischen Interface `bond0` zusammengefasst. Dazu muss zuerst das Kernel-Modul `bonding` geladen werden.

Weitere Einstellungen müssen über Dateien in `/sys/class/net/bond0/bonding/` vorgenommen werden.

Standardmäßig wird der Round-Robin-Modus zur Lastverteilung verwendet.

## Port-Aggregation – /etc/network/interfaces

```
1 auto bond0
2 iface bond0 inet dhcp
3   bond_slaves eth0 eth1
4   bond_mode 802.3ad
5   bond_miimon 100
6   bond_xmit_hash_policy layer3+4
7   bond_lacp_rate slow
```



Das Beispiel zeigt, wie ein Bonding-Interface in `/etc/network/interfaces` konfiguriert wird. Dies ist deutlich bequemer als über `ifenslave` und `/sys/class/net/bond0/bonding/`. `ifenslave` muss trotzdem installiert sein, da es im Hintergrund verwendet wird.

Im Beispiel wurde LACP verwendet und die Lastverteilung über IP-Adressen und UDP- und TCP-Port-Nummern realisiert. Die Lastverteilung über Port-Nummern entspricht nicht vollständig dem LACP-Standard. Bei Problemen mit inkompatiblen Gegenstellen sollte daher darauf verzichtet werden (`bond_xmit_hash_policy layer3`).