4.4 Der Rechner im Netzwerk

Referenz zu den LPI-Lernzielen

Linux Essentials version 1.6, Exam 010, Objective 4.4

Gewichtung

2

Hauptwissensgebiete

- Internet, Netzwerk, Router
- Abfragen der DNS-Client-Konfiguration
- Abfragen der Netzwerk-Konfiguration

Auszugsweise Liste der verwendeten Dateien, Begriffe und Hilfsprogramme

- `route`, `ip route show`
- `ifconfig`, `ip addr show`
- `netstat`, `ss`
- `/etc/resolv.conf`, `/etc/hosts`
- `IPv4`, `IPv6`
- `ping`
- `host`

4.4 Lektion 1

Zertifikat:	Linux Essentials
Version:	1.6
Thema:	4 Das Linux-Betriebssystem
Lernziel:	4.4 Der Rechner im Netzwerk
Lektion:	1 von 1

Einführung

In der heutigen Welt tauschen Computer jeglicher Art Informationen über Netzwerke aus. Grundlage jedes Computernetzwerks sind die physikalischen Verbindungen zwischen den Geräten, den sog. *Peers*. Die Verbindungen bezeichnet man als *Links*; sie bilden die grundlegendste Verbindung zwischen zwei Geräten. Links können über verschiedene Medien wie Kupferkabel, Glasfaser, Radiowellen oder Laser hergestellt werden.

Jede Verbindung erfolgt über eine Schnittstelle im Gerät. Jedes Gerät kann mehrere Schnittstellen haben und so über mehrere Links verbunden sein. Über diese Links bilden die Computer ein Netzwerk, eine kleine Gruppe von Geräten, die sich direkt miteinander verbinden können. Es gibt weltweit zahlreiche solcher Netzwerke. Um aber über ein solches *Link-Layer-Netzwerk* hinaus zu kommunizieren, benötigen die Geräte sog. *Router*. Stellen Sie sich Link-Layer-Netzwerke als Inseln vor, die durch Router verbunden sind — wie Brücken, über die Informationen transportiert werden müssen, um ein Gerät auf einer anderen Insel zu erreichen.

Dieses Modell führt zu verschiedenen Schichten der Vernetzung:

Link Layer (Verbindungsschicht)

Steuert die Kommunikation zwischen direkt angeschlossenen Geräten.

Network Layer (Netzwerkschicht)

Steuert das Routing außerhalb einzelner Netzwerke und die eindeutige Adressierung von Geräten über ein einzelnes Link-Layer-Netzwerk hinaus.

Application Layer (Anwendungsschicht)

Ermöglicht es einzelnen Programmen, sich miteinander zu verbinden.

Bei ihrer Erfindung verwendeten Computernetzwerke die gleichen Kommunikationsmethoden wie Telefone, waren also leitungsvermittelt. Es musste also eine dedizierte und direkte Verbindung zwischen zwei Knoten für deren Kommunikation hergestellt werden. Diese Methode funktionierte gut, benötigte aber die gesamte Kapazität der jeweiligen Verbindung, damit nur zwei Hosts kommunizieren konnten.

Irgendwann gingen Computernetzwerke zur sogenannten *Paketvermittlung* (*Packet Switching*) über. Hier werden die Daten mit einem Header versehen, der Informationen darüber enthält, woher die Informationen kommen und wohin sie gehen. Die eigentlichen Inhaltsinformationen befinden sich in diesem Frame und werden über die Verbindung an den im Header des Frames angegebenen Empfänger gesendet. Das bietet die Möglichkeit, dass sich mehrere Geräte einen einzigen Link teilen und fast gleichzeitig kommunizieren können.

Die Verbindungsschicht (Link Layer)

Die Aufgabe eines jeden Pakets ist es, Informationen von der Quelle zum Ziel über eine Verbindung zwischen beiden Geräten zu übertragen. Diese Geräte benötigen eine Möglichkeit, sich gegenseitig zu identifizieren, und das ist der Zweck einer *Link-Layer-Adresse*. In einem Ethernet-Netzwerk werden *Media Access Control* (MAC) Adressen zur Identifizierung einzelner Geräte verwendet. Eine MAC-Adresse besteht aus 48 Bit. Sie sind nicht global eindeutig und darum ungeeignet, um Peers außerhalb des aktuellen Links anzusprechen. Daher können diese Adressen nicht verwendet werden, um Pakete an andere Links weiterzuleiten. Der Empfänger eines Pakets prüft, ob die Zieladresse mit seiner eigenen Verbindungsschicht übereinstimmt, und, falls ja, verarbeitet er das Paket. Andernfalls wird das Paket verworfen. Die Ausnahme von dieser Regel sind *Broadcast Packets* (also solche, die an alle Teilnehmer in einem bestimmten lokalen Netzwerk gesendet werden), die immer akzeptiert werden.

Der Befehl `ip link show` zeigt eine Liste aller verfügbaren Netzwerkschnittstellen, deren Link-Layer-Adressen sowie einige weitere Informationen über die maximale Paketgröße:

```
$ ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group
default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode DEFAULT
group default qlen 1000
    link/ether 00:0c:29:33:3b:25 brd ff:ff:ff:ff:ff:ff
```

Die obige Ausgabe zeigt, dass das Gerät zwei Schnittstellen hat, `lo` und `ens33`. `lo` ist das *Loopback Device* mit der MAC-Adresse `00:00:00:00:00:00`, während `ens33` eine Ethernet-Schnittstelle mit der MAC-Adresse `00:0c:29:33:3b:25` ist.

IPv4-Netzwerke

Um Websites wie Google oder Twitter zu besuchen, E-Mails abzurufen oder Unternehmen die Möglichkeit zu geben, sich miteinander zu verbinden, müssen sich Pakete von einem Link-Layer-Netzwerk zum anderen bewegen können. Oft sind diese Netzwerke nur indirekt über weitere Link-Layer-Netzwerke verbunden, die die Pakete durchqueren müssen, um ihr eigentliches Ziel zu erreichen.

Die Adressen der Verbindungsschicht einer Netzwerkschnittstelle können nicht außerhalb dieses spezifischen Link-Layer-Netzwerks verwendet werden. Da diese Adresse für Geräte in anderen Link-Layer-Netzwerken keine Bedeutung hat, werden für die Implementierung des Routings global eindeutige Adressen benötigt. Dieses Adressierungsschema wird ebenso wie das Gesamtkonzept des Routings durch das *Internet Protocol* (IP) implementiert.

NOTE

Ein *Protokoll* beschreibt eine Reihe von Verfahren und Abläufen, so dass alle dem Protokoll folgenden Teilnehmer zueinander kompatibel sind. Ein Protokoll ist also eine Art Definition einer Norm. Bei der Datenverarbeitung ist das Internet Protocol ein allgemein vereinbarter Standard, so dass verschiedene Geräte verschiedener Hersteller miteinander kommunizieren können.

IPv4-Adressen

IP-Adressen sind, ähnlich wie MAC-Adressen, eine Möglichkeit anzugeben, woher ein Datenpaket kommt und wohin es geht. IPv4 war das ursprüngliche Protokoll. IPv4-Adressen sind 32 Bit lang und ergeben eine theoretische maximale Anzahl von 4.294.967.296 Adressen. Die Anzahl der von den Geräten nutzbaren Adressen ist jedoch viel geringer, da einige Bereiche für spezielle Anwendungsfälle reserviert sind, wie z.B. Broadcast-Adressen (mit denen alle Teilnehmer eines bestimmten Netzwerks erreicht werden), Multicast-Adressen (ähnlich wie Broadcast-Adressen,

aber ein Gerät muss sich wie ein Radio einschalten) oder solche, die für den privaten Gebrauch reserviert sind.

In ihrem menschenlesbaren Format werden IPv4-Adressen als vier jeweils durch einen Punkt getrennte Zahlen geschrieben, wobei jede Zahl zwischen 0 und 255 liegt, z.B.:

```
192.168.0.1
```

Technisch gesehen stellt jede dieser Zahlen acht einzelne Bits dar, so dass diese Adresse auch so geschrieben werden könnte:

```
11000000.10101000.00000000.00000001
```

In der Praxis wird die dezimale Notation, wie oben beschrieben, verwendet, jedoch ist die bitweise Darstellung wichtig, um das sog. *Subnetting* zu verstehen.

IPv4-Subnetze

Um Routing zu unterstützen, können IP-Adressen in zwei Teile aufgeteilt werden: den Netzwerk- und den Host-Teil. Der Netzwerkteil identifiziert das Netzwerk, in dem sich das Gerät befindet, und dient dazu, Pakete an dieses Netzwerk weiterzuleiten. Der Host-Teil dient dazu, eine bestimmtes Gerät in einem Netzwerk zu identifizieren und das Paket an seinen spezifischen Empfänger weiterzugeben, sobald es dessen Verbindungsschichtnetzwerk erreicht hat.

IP-Adressen können jederzeit in Subnetz- und Host-Teile zerlegt werden, wobei die sogenannte *Subnetzmaske* definiert, wo diese Aufteilung erfolgt. Betrachten wir die binäre Darstellung der IP-Adresse aus dem vorherigen Beispiel:

```
11000000.10101000.00000000.00000001
```

Nun setzt die Subnetzmaske für diese IP-Adresse jedes Bit, das zum Netzwerkteil gehört, auf `1` und jedes Bit, das zum Host-Teil gehört, auf `0`:

```
11111111.11111111.11111111.00000000
```

In der Praxis wird die Netzmaske in dezimaler Schreibweise notiert:

```
255.255.255.0
```

Das bedeutet, dass dieses Netzwerk von `192.168.0.0` bis `192.168.0.255` reicht. Beachten Sie, dass die ersten drei Zahlen, bei denen alle Bits in der Netzmaske gesetzt sind, in den IP-Adressen unverändert bleiben.

Schließlich gibt es noch eine alternative Notation für die Subnetzmaske, die *Classless Inter-Domain Routing* (CIDR) genannt wird. Sie gibt lediglich an, wie viele Bits in der Subnetzmaske gesetzt sind, und fügt diese Zahl zur IP-Adresse hinzu. 24 von 32 Bits sind im Beispiel in der Subnetzmaske auf `1` gesetzt, was in CIDR-Notation als `192.168.0.1/24` ausgedrückt wird.

Private IPv4-Adressen

Wie bereits erwähnt, sind bestimmte Abschnitte des IPv4-Adressraums für spezielle Anwendungsfälle reserviert, darunter die private Adresszuweisung. Die folgenden Subnetze sind für die private Adressierung reserviert:

- `10.0.0.0/8`
- `172.16.0.0/12`
- `192.168.0.0/16`

Jedermann kann Adressen aus diesen Subnetzen nutzen, aber sie werden im öffentlichen Internet nicht geroutet werden, da potentiell zahlreiche Netzwerke sie gleichzeitig nutzen.

Heute verwenden die meisten Netzwerke diese internen Adressen. Sie ermöglichen die interne Kommunikation ohne externe Adresszuweisung. Die meisten Internetverbindungen haben heute nur noch eine einzige externe IPv4-Adresse. Router bilden beim Weiterleiten von Paketen ins Internet alle internen Adressen auf diese einzelne externe IP-Adresse ab, was als *Network Address Translation* (NAT) bezeichnet wird. Der Sonderfall von NAT, bei dem ein Router interne Adressen auf eine einzige externe IP-Adresse abbildet, wird manchmal als *Masquerading* bezeichnet. Dies ermöglicht es jedem Gerät im internen Netzwerk, neue Verbindungen mit jeder globalen IP-Adresse im Internet herzustellen.

NOTE

Beim Masquerading können die internen Geräte nicht aus dem Internet referenziert werden, da sie keine global gültige Adresse haben. Dies ist jedoch kein Sicherheitsmerkmal. Selbst bei der Verwendung von Masquerading ist eine Firewall erforderlich.

IPv4-Adresskonfiguration

Es gibt zwei Möglichkeiten, IPv4-Adressen auf einem Computer zu konfigurieren: Entweder durch manuelle Adresszuweisung oder durch Verwendung des *Dynamic Host Configuration Protocol* (DHCP) für die automatische Konfiguration.

Beim Einsatz von DHCP steuert ein zentraler Server, welche Adressen an welche Geräte übergeben werden. Der Server versorgt Geräte auch mit anderen Netzwerkinformationen, z.B. zu IP-Adressen von DNS-Servern, zur IP-Adresse des Standard-Routers oder — bei komplexeren Setups — zum Start eines Betriebssystems aus dem Netzwerk. DHCP ist auf vielen Systemen standardmäßig aktiviert, so dass Sie wahrscheinlich bereits eine IP-Adresse haben, wenn Sie mit einem Netzwerk verbunden sind.

IP-Adressen lassen sich auch manuell mit dem Befehl `ip addr add` einer Schnittstelle zuweisen. Hier geben wir die Adresse `192.168.0.5` der Schnittstelle `ens33`. Das Netzwerk verwendet die Netzmaske `255.255.255.0`, was `/24` in CIDR-Notation entspricht:

```
$ sudo ip addr add 192.168.0.5/255.255.255.0 dev ens33
```

Jetzt prüfen wir mit dem Befehl `ip addr show`, ob die Adresse zugewiesen wurde:

```
$ ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
25: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen
1000
    link/ether 00:0c:29:33:3b:25 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.5/24 192.168.0.255 scope global ens33
       valid_lft forever preferred_lft forever
    inet6 fe80::010c:29ff:fe33:3b25/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

Die Ausgabe zeigt sowohl die `lo`- als auch die `ens33`-Schnittstelle mit der oben zugewiesenen Adresse.

Um die Erreichbarkeit eines Geräts zu prüfen, dient der Befehl `ping`. Er sendet eine spezielle Nachricht namens *Echo Request*, mit der der Absender den Empfänger um eine Antwort bittet. Kann die Verbindung zwischen den beiden Geräten erfolgreich hergestellt werden, sendet der Empfänger ein *Echo Reply* zurück und bestätigt damit die Verbindung:

```
$ ping -c 3 192.168.0.1
PING 192.168.0.1 (192.168.0.1) 56(84) bytes of data.
64 bytes from 192.168.0.1: icmp_seq=1 ttl=64 time=2.16 ms
```

```
64 bytes from 192.168.0.1: icmp_seq=2 ttl=64 time=1.85 ms
64 bytes from 192.168.0.1: icmp_seq=3 ttl=64 time=3.41 ms

--- 192.168.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 5ms
rtt min/avg/max/mdev = 1.849/2.473/3.410/0.674 ms
```

Der Parameter `-c 3` bewirkt, dass `ping` nach dem Senden von drei Echo Requests stoppt. Andernfalls läuft `ping` immer weiter und muss durch `Ctrl` + `C` gestoppt werden.

IPv4-Routing

Routing ist der Prozess, bei dem ein Paket vom Quellnetzwerk zum Zielnetzwerk gelangt. Jedes Gerät hält eine Routingtabelle vor, die Informationen darüber enthält, welches IP-Netzwerk über den Anschluss an das Verbindungsschichtnetzwerk direkt erreichbar ist und welches IP-Netzwerk durch Weiterleitung von Paketen an einen Router erreicht werden kann. Schließlich definiert eine *Default Route* einen Router, der alle Pakete empfängt, die keiner anderen Route entsprechen.

Beim Verbindungsaufbau sucht das Gerät die IP-Adresse des Ziels in seiner Routingtabelle. Stimmt ein Eintrag mit der Adresse überein, wird das Paket entweder an das jeweilige Verbindungsschichtnetzwerk gesendet oder an den in der Routingtabelle angegebenen Router weitergeleitet.

Router selbst pflegen ebenfalls Routingtabellen. Empfängt er ein Paket, sucht ein Router die Zieladresse in seiner eigenen Routingtabelle und sendet das Paket an den nächsten Router, was sich wiederholt, bis das Paket beim Router im Zielnetzwerk ankommt. Jeder an dieser Reise beteiligte Router wird als *Hop* bezeichnet. Der letzte Router findet eine direkte Verbindung zur Zieladresse in seiner Routingtabelle und sendet die Pakete zur Zielschnittstelle.

Die meisten Heimnetzwerke haben nur einen Weg nach draußen, einen einzigen Router vom *Internet Service Provider* (ISP). In diesem Fall leitet ein Gerät einfach alle Pakete, die nicht für das interne Netzwerk bestimmt sind, direkt an den Heimrouter weiter, der sie dann zur Weiterleitung an den Router des Providers sendet. Dies ist ein Beispiel für eine Standardroute.

Der Befehl `ip route show` listet die aktuelle IPv4-Routingtabelle auf:

```
$ ip route show
127.0.0.0/8 via 127.0.0.1 dev lo0
192.168.0.0/24 dev ens33 scope link
```

Um eine Standardroute hinzuzufügen, benötigen Sie lediglich die interne Adresse des Routers, der

das Standard-Gateway sein wird. Wenn der Router beispielsweise die Adresse `192.168.0.1` hat, dann wird er mit dem folgenden Befehl als Standardroute eingerichtet:

```
$ sudo ip route add default via 192.168.0.1
```

Um dies zu überprüfen, führen Sie `ip route show` erneut aus:

```
$ ip route show
default via 192.168.0.1 dev ens33
127.0.0.0/8 via 127.0.0.1 dev lo0
192.168.0.0/24 dev ens33 scope link
```

IPv6-Netzwerke

IPv6 wurde entwickelt, um die Beschränkungen von IPv4 zu überwinden, vor allem den Mangel an Adressen, da immer mehr Geräte online gehen. IPv6 umfasst aber auch andere Funktionen wie z.B. neue Protokolle zur automatischen Netzwerkkonfiguration. Anstelle von 32 Bit pro Adresse verwendet IPv6 128. Dies ermöglicht ca. 2^{128} Adressen. Wie bei IPv4 ist die Anzahl der weltweit eindeutigen nutzbaren Adressen jedoch deutlich geringer, da Teile der Zuordnung für andere Nutzungen reserviert sind. Diese große Anzahl von Adressen bedeutet, dass es mehr als genug öffentliche Adressen für jedes Gerät gibt, das derzeit mit dem Internet verbunden ist, und für viele weitere, die kommen werden. Das wiederum verringert die Notwendigkeit von Masquerading und umgeht die damit verbundenen Probleme wie die Verzögerung bei der Übersetzung und die Beschränkung, sich nicht direkt mit maskierten Geräten verbinden zu können.

IPv6-Adressen

IPv6-Adressen bestehen aus 8 Gruppen von 4 hexadezimalen Ziffern, die jeweils durch einen Doppelpunkt getrennt sind:

```
2001:0db8:0000:abcd:0000:0000:0000:7334
```

NOTE Hexadezimale Ziffern reichen von `0` bis `f`, so dass jede Ziffer einen von 16 verschiedenen Werten haben kann.

Um es einfacher zu machen, können führende Nullen aus jeder Gruppe bei der Notierung entfernt werden, jedoch muss jede Gruppe mindestens eine Ziffer enthalten:

```
2001:db8:0:abcd:0:0:0:7334
```

Wenn mehrere Gruppen, die nur Nullen enthalten, unmittelbar hintereinander folgen, können sie vollständig durch `::` ersetzt werden:

```
2001:db8:0:abcd::7334
```

Das ist jedoch nur einmal in jeder Adresse erlaubt.

IPv6-Präfix

Die ersten 64 Bit einer IPv6-Adresse werden als *Routing-Präfix* bezeichnet und von Routern dazu verwendet, um zu bestimmen, zu welchem Netzwerk ein Gerät gehört und somit auf welchem Pfad die Daten gesendet werden müssen. Subnetting findet immer innerhalb des Präfix statt. ISPs vergeben in der Regel /48 oder /58 Präfixe an ihre Kunden, so dass sie 16 oder 8 Bit für ihr internes Subnetting übrig haben.

Es gibt drei wichtige Präfix-Typen in IPv6:

Global Unique Address

Das Präfix wird aus den für globale Adressen reservierten Blöcken vergeben, die im gesamten Internet gültig sind.

Unique Local Address

Dürfen nicht im Internet, können aber intern innerhalb einer Organisation geroutet werden. Diese Adressen werden innerhalb eines Netzwerks verwendet, um sicherzustellen, dass die Geräte auch ohne Internetverbindung eine Adresse haben. Sie entsprechen den privaten Adressbereichen von IPv4. Die ersten 8 Bits sind immer `fc` oder `fd`, gefolgt von 40 zufällig erzeugten Bits.

Link Local Address

Sind nur für einen bestimmten Link gültig. Jede IPv6-fähige Netzwerkschnittstelle hat eine solche Adresse, beginnend mit `fe80`. Diese Adressen werden intern von IPv6 verwendet, um zusätzliche Adressen über die automatische Konfiguration anzufordern und andere Computer im Netzwerk über das Neighbor Discovery Protocol zu finden.

IPv6 Interface Identifier

Während das Präfix bestimmt, in welchem Netzwerk sich ein Gerät befindet, wird die

Schnittstellenkennung verwendet, um die Geräte innerhalb eines Netzwerks zu listen. Die letzten 64 Bit einer IPv6-Adresse bilden die Schnittstellenkennung, genau wie die letzten Bit einer IPv4-Adresse.

Bei der manuellen Vergabe einer IPv6-Adresse wird die Schnittstellenkennung als Teil der Adresse eingestellt. Bei der automatischen Adresskonfiguration wird die Schnittstellenkennung entweder zufällig oder aus der Link-Layer-Adresse des Geräts abgeleitet, so dass eine Variation der Link-Layer-Adresse innerhalb der IPv6-Adresse erscheint.

IPv6-Adresskonfiguration

Wie eine IPv4- kann auch eine IPv6-Adresse entweder manuell oder automatisch zugewiesen werden, wobei IPv6 zwei verschiedene Arten der automatisierten Konfiguration bietet: DHCPv6 und *Stateless Address Autoconfiguration* (SLAAC).

SLAAC ist das einfachere der beiden automatisierten Verfahren und direkt in den IPv6-Standard integriert. Die Nachrichten verwenden das neue *Neighbor Discovery Protocol*, das es Geräten ermöglicht, sich gegenseitig zu finden und Informationen über ein Netzwerk abzufragen. Diese Informationen werden von Routern gesendet und können IPv6-Präfixe enthalten, die die Geräte verwenden können, indem sie diese mit einer Schnittstellenkennung ihrer Wahl kombinieren, solange die resultierende Adresse sie noch nicht verwendet. Die Geräte geben dem Router keine Rückmeldung über die tatsächlich erstellten Adressen.

DHCPv6 hingegen ist das aktualisierte DHCP, das mit den Änderungen von IPv6 kompatibel ist und eine feinere Kontrolle über die an die Clients übergebenen Informationen ermöglicht, wie z.B. die Möglichkeit, dass immer dieselbe Adresse an denselben Client vergeben wird, und die Übertragung von mehr Optionen an den Client als SLAAC. Bei DHCPv6 müssen Clients die ausdrückliche Zustimmung eines DHCP-Servers einholen, um eine Adresse zu verwenden.

Das Verfahren zur manuellen Zuweisung einer IPv6-Adresse zu einer Schnittstelle ist das gleiche wie bei IPv4:

```
$ sudo ip addr add 2001:db8:0:abcd:0:0:0:7334/64 dev ens33
```

Um sicherzustellen, dass die Zuweisung erfolgreich war, wird derselbe Befehl `ip addr show` verwendet:

```
$ ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
```

```
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
25: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen
1000
    link/ether 00:0c:29:33:3b:25 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.5/24 192.168.0.255 scope global ens33
       valid_lft forever preferred_lft forever
    inet6 fe80::010c:29ff:fe33:3b25/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
    inet6 2001:db8:0:abcd::7334/64 scope global
       valid_lft forever preferred_lft forever
```

Hier sehen wir auch die Link-Local-Adresse `fe80::010c:29ff:fe33:3b25/64`.

Wie bei IPv4 kann auch der Befehl `ping` verwendet werden, um die Erreichbarkeit von Geräten über IPv6 zu bestätigen:

```
$ ping 2001:db8:0:abcd::1
PING 2001:db8:0:abcd::1(2001:db8:0:abcd::1) 56 data bytes
64 bytes from 2001:db8:0:abcd::1: icmp_seq=1 ttl=64 time=0.030 ms
64 bytes from 2001:db8:0:abcd::1: icmp_seq=2 ttl=64 time=0.040 ms
64 bytes from 2001:db8:0:abcd::1: icmp_seq=3 ttl=64 time=0.072 ms

--- 2001:db8:0:abcd::1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 43ms
rtt min/avg/max/mdev = 0.030/0.047/0.072/0.018 ms
```

NOTE Auf einigen Linux-Systemen unterstützt `ping` kein IPv6. Diese Systeme bieten stattdessen einen dedizierten `ping6`-Befehl.

Um die Link-Local-Adresse erneut zu überprüfen, verwenden Sie erneut `ping`. Da jedoch alle Interfaces das Präfix `fe80::/64` verwenden, muss neben der Adresse auch das richtige Interface angegeben werden:

```
$ ping6 -c 1 fe80::010c:29ff:fe33:3b25%ens33
PING fe80::010c:29ff:fe33:3b25(fe80::010c:29ff:fe33:3b25) 56 data bytes
64 bytes from fe80::010c:29ff:fe33:3b25%ens33: icmp_seq=1 ttl=64 time=0.049 ms

--- fe80::010c:29ff:fe33:3b25 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
```

```
rtt min/avg/max/mdev = 0.049/0.049/0.049/0.000 ms
```

DNS

IP-Adressen sind schwer zu merken und haben nicht gerade einen hohen Coolness-Faktor, wenn Sie versuchen, einen Dienst oder ein Produkt zu vermarkten. Hier kommt das *Domain Name System* ins Spiel: In seiner einfachsten Form ist DNS ein verteiltes Telefonbuch, das einprägsame Domainnamen wie `example.com` auf IP-Adressen abbildet. Wenn ein Benutzer beispielsweise zu einer Website navigiert, gibt er den DNS-Hostnamen als Teil der URL ein. Der Webbrowser sendet dann den DNS-Namen an den konfigurierten DNS-Resolver. Dieser DNS-Resolver findet wiederum die Adresse heraus, die mit der Domain korreliert. Der Resolver antwortet dann mit dieser Adresse und der Webbrowser versucht, den Webserver unter dieser IP-Adresse zu erreichen.

Die Resolver, die Linux verwendet, um DNS-Daten abzurufen, werden in der Datei `/etc/resolv.conf` konfiguriert:

```
$ cat /etc/resolv.conf
search lpi
nameserver 192.168.0.1
```

Wenn der Resolver eine Namenssuche durchführt, überprüft er zunächst die Datei `/etc/hosts`, um zu sehen, ob sie eine Adresse für den angeforderten Namen enthält. Wenn ja, gibt er diese Adresse zurück und kontaktiert das DNS nicht, so dass Netzwerkadministratoren eine Namensauflösung bereitstellen können, ohne einen kompletten DNS-Server konfigurieren zu müssen. Jede Zeile in dieser Datei enthält eine IP-Adresse, gefolgt von einem oder mehreren Namen:

```
127.0.0.1          localhost.localdomain   localhost
::1                localhost.localdomain   localhost
192.168.0.10       server
2001:db8:0:abcd::f server
```

Um einen Lookup im DNS durchzuführen, verwenden Sie den Befehl `host`:

```
$ host learning.lpi.org
learning.lpi.org has address 208.94.166.198
```

Detailliertere Informationen liefert der Befehl `dig`:

```
$ dig learning.lpi.org
; <<>> DiG 9.14.3 <<>> learning.lpi.org
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 21525
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: 2ac55879b1adef30a93013705d3306d2128571347df8eadf (bad)
;; QUESTION SECTION:
;learning.lpi.org.      IN  A

;; ANSWER SECTION:
learning.lpi.org.   550 IN  A   208.94.166.198

;; Query time: 3 msec
;; SERVER: 192.168.0.1#53(192.168.0.1)
;; WHEN: Sat Jul 20 14:20:21 EST 2019
;; MSG SIZE  rcvd: 89
```

Hier sehen wir auch den Namen der DNS-Record-Typs, in diesem Fall `A` für IPv4.

Sockets

Ein *Socket* ist ein Kommunikationsendpunkt für zwei Programme, die miteinander kommunizieren. Wenn der Socket über ein Netzwerk verbunden ist, können die Programme auf verschiedenen Geräten ausgeführt werden, wie z.B. ein Webbrowser auf dem Laptop eines Benutzers und ein Webserver im Rechenzentrum eines Unternehmens.

Es gibt drei Haupttypen von Sockets:

Unix Sockets
: verbinden Prozesse, die auf demselben Gerät laufen.

UDP (User Datagram Protocol) Sockets
: verbinden Anwendungen über ein Protokoll, das schnell, aber nicht belastbar ist.

TCP (Transmission Control Protocol) Sockets
: sind zuverlässiger als UDP Sockets und bestätigen beispielsweise den Empfang von Daten.

Unix Sockets können nur Anwendungen verbinden, die auf demselben Gerät laufen. TCP und UDP

Sockets können sich hingegen über ein Netzwerk verbinden. TCP ermöglicht einen Datenstrom, der immer in der exakten Reihenfolge ankommt, in der er gesendet wurde. UDP folgt mehr dem Prinzip “feuern und vergessen”: das Paket wird gesendet, aber seine Zustellung am anderen Ende ist nicht garantiert. UDP hat jedoch nicht den Overhead von TCP, was es perfekt für Anwendungen mit niedriger Latenz wie Online-Videospiele macht.

TCP und UDP verwenden Ports, um mehrere Sockets an derselben IP-Adresse zu adressieren. Während der Quellport für eine Verbindung in der Regel zufällig ist, sind die Zielports für einen bestimmten Dienst standardisiert. So wird HTTP normalerweise an Port 80 gehostet, HTTPS an Port 443. SSH, ein Protokoll zur sicheren Anmeldung an einem entfernten Linux-System, lauscht an Port 22.

Der Befehl `ss` ermöglicht es einem Administrator, alle Sockets auf einem Linux-Rechner zu untersuchen. Es zeigt alles von der Quelladresse über die Zieladresse bis zum Typ. Ein herausragendes Feature sind Filter, so dass ein Benutzer die Sockets in jedem gewünschten Verbindungszustand überwachen kann. `ss` kann mit einer Reihe von Optionen sowie einem Filterausdruck ausgeführt werden, um die angezeigten Informationen einzuschränken.

Wird der Befehl ohne Optionen ausgeführt, zeigt er eine Liste aller eingerichteten Sockets an. Die Option `-p` liefert Informationen über den Prozess, der einen Socket nutzt. `-s` zeigt eine Zusammenfassung der Sockets an. Das Tool bietet zahlreiche Optionen, von denen `-4` und `-6` zu den wichtigsten gehören, da sie das IP-Protokoll auf IPv4 bzw. IPv6 eingrenzen. `-t` und `-u` erlauben es dem Administrator, TCP oder UDP Sockets auszuwählen, und `-l` zeigt nur Sockets an, die auf neue Verbindungen warten.

Der folgende Befehl listet beispielsweise alle derzeit verwendeten TCP Sockets auf:

```
$ ss -t
State       Recv-Q  Send-Q    Local Address:Port      Peer Address:Port
ESTAB       0       0           192.168.0.5:49412      192.168.0.1:https
ESTAB       0       0           192.168.0.5:37616      192.168.0.1:https
ESTAB       0       0           192.168.0.5:40114      192.168.0.1:https
ESTAB       0       0           192.168.0.5:54948      192.168.0.1:imap
...
```

Geführte Übungen

1. Ein Netzwerkingenieur wird gebeten, der `ens33`-Schnittstelle eines Hosts zwei IP-Adressen zuzuweisen, eine IPv4-Adresse (`192.168.10.10/24`) und eine IPv6-Adresse (`2001:0:0:0:abcd:0:8a2e:0370:7334/64`). Welche Befehle müssen Sie eingeben, um dies zu erreichen?

2. Welche Adressen aus der untenstehenden Liste sind privat?

`192.168.10.1`	
`120.56.78.35`	
`172.16.57.47`	
`10.100.49.162`	
`200.120.42.6`	

3. Welchen Eintrag würden Sie in der Datei `hosts` hinzufügen, um `192.168.0.15` `example.com` zuzuordnen?

4. Was bewirkt der folgende Befehl?

```
sudo ip -6 route add default via 2001:db8:0:abcd::1
```

Offene Übungen

1. Benennen Sie den DNS-Record-Typ, der folgende Anfragen bedient:
   - Text-Daten
   - Reverse Lookup nach IP-Adressen
   - Eine Domain, die keine eigene Adresse hat und für diese Informationen auf eine andere Domain angewiesen ist.
   - Mailserver
2. Linux hat eine Funktion namens Bridging, was macht sie und wo ist sie nützlich?
3. Mit welcher Option muss der Befehl `ss` aufgerufen werden, um alle eingerichteten UDP Sockets anzuzeigen?
4. Welcher Befehl zeigt eine Zusammenfassung aller Sockets, die auf einem Linux-Gerät laufen?
5. Die folgende Ausgabe wird durch den Befehl aus der vorherigen Übung erzeugt. Wie viele TCP und UDP Sockets sind aktiv?

```
Total: 978 (kernel 0)
TCP:   4 (estab 0, closed 0, orphaned 0, synrecv 0, timewait 0/0), port

Transport Total     IP        IPv6
*     0         -         -
RAW   1         0         1
UDP   7         5         2
TCP   4         3         1
INET      12        8         4
FRAG      0         0         0
```

Zusammenfassung

In dieser Lektion ging es um die Vernetzung Ihres Linux-Computers. Zuerst haben wir die verschiedenen Ebenen der Vernetzung betrachtet:

- Die Verbindungsschicht, die Geräte direkt miteinander verbindet.
- Die Netzwerkschicht, die das Routing zwischen Netzwerken und einem globalen Adressraum ermöglicht.
- Die Anwendungsschicht, in der Anwendungen miteinander verbunden sind.

Wir haben gesehen, wie IPv4 und IPv6 verwendet werden, um einzelne Computer anzusprechen, und wie TCP und UDP Sockets nummerieren, die von Anwendungen verwendet werden, um sich miteinander zu verbinden. Zudem haben wir betrachtet, wie DNS verwendet wird, um Namen in IP-Adressen aufzulösen.

Befehle, die in den Übungen verwendet werden:

`dig`
: Abfrage von DNS-Informationen und Bereitstellung umfangreicher Informationen über die DNS-Abfragen und -Antworten.

`host`
: Abfrage von DNS-Informationen und Bereitstellung einer komprimierten Ausgabe.

`ip`
: Konfiguriert Netzwerke unter Linux, einschließlich Netzwerkschnittstellen, Adressen und Routing.

`ping`
: Testet die Verbindung zu einem entfernten Gerät.

`ss`
: Zeigt Informationen zu Sockets an.

Lösungen zu den geführten Übungen

1. Ein Netzwerkingenieur wird gebeten, der `ens33`-Schnittstelle eines Hosts zwei IP-Adressen zuzuweisen, eine IPv4-Adresse (`192.168.10.10/24`) und eine IPv6-Adresse (`2001:0:0:0:abcd:0:8a2e:0370:7334/64`). Welche Befehle müssen Sie eingeben, um dies zu erreichen?

```
sudo ip addr add 192.168.10.10/24 dev ens33
sudo ip addr add 2001:0:0:abcd:0:8a2e:0370:7334/64 dev ens33
```

2. Welche Adressen aus der untenstehenden Liste sind privat?

`192.168.10.1`	X
`120.56.78.35`	
`172.16.57.47`	X
`10.100.49.162`	X
`200.120.42.6`	

3. Welchen Eintrag würden Sie in der Datei `hosts` hinzufügen, um `192.168.0.15` `example.com` zuzuordnen?

```
192.168.0.15  example.com
```

4. Was bewirkt der folgende Befehl?

```
sudo ip -6 route add default via 2001:db8:0:abcd::1
```

   Er fügt eine Standardroute in die Tabelle ein, die den gesamten IPv6-Verkehr an den Router mit der internen Adresse `2001:db8:0:abcd::1` sendet.

Lösungen zu den offenen Übungen

1. Benennen Sie den DNS-Record-Typ, der folgende Anfragen bedient:
   - Text-Daten

     TXT

   - Reverse Lookup nach IP-Adressen

     PTR

   - Eine Domain, die keine eigene Adresse hat und für diese Informationen auf eine andere Domain angewiesen ist.

     CNAME

   - Mailserver

     MX

2. Linux hat eine Funktion namens Bridging, was macht sie und wo ist sie nützlich?

   Eine Bridge verbindet mehrere Netzwerkschnittstellen. Alle an eine Bridge angeschlossenen Schnittstellen können so kommunizieren, als seien sie mit demselben Verbindungsschichtnetzwerk verbunden. Alle Geräte verwenden IP-Adressen aus demselben Subnetz und benötigen keinen Router, um sich miteinander zu verbinden.

3. Mit welcher Option muss der Befehl `ss` aufgerufen werden, um alle eingerichteten UDP Sockets anzuzeigen?

   Die Option `-u` zeigt alle eingerichteten UDP Sockets.

4. Welcher Befehl zeigt eine Zusammenfassung aller Sockets, die auf einem Linux-Gerät laufen?

   Der Befehl `ss -s` zeigt eine Zusammenfassung aller Sockets.

5. Die folgende Ausgabe wird durch den Befehl aus der vorherigen Übung erzeugt. Wie viele TCP und UDP Sockets sind aktiv?

```
Total: 978 (kernel 0)
TCP:   4 (estab 0, closed 0, orphaned 0, synrecv 0, timewait 0/0), port

Transport Total     IP        IPv6
```

```
*     0         -         -
RAW   1         0         1
UDP   7         5         2
TCP   4         3         1
INET      12        8         4
FRAG      0         0         0
```

11 TCP und UDP Sockets sind aktiv.