

7 Transportprotokolle



- 7.1 Transmission Control Protocol (TCP)
- 7.2 User Datagram Protocol (UDP)
- 7.3 Ports

7.1 TCP (1)



- IP-Pakete (Datagramme) von A nach B transportieren reicht nicht
- interaktive Verbindungen sollen zeitnah laufen
- Gewisse Anwendungen benötigen zuverlässige Verbindungen

7.1.2 TCP (2)



- TCP stellt einen zuverlässigen Dienst über IP-Verbindungen her
- bringt seine Informationen in einem 20 Byte langen Header unter
- identifiziert die Endpunkte der Verbindung anhand der IP-Aufgabe der Transportschicht
- etabliert ein verbindungsorientiertes Protokoll auf dem verbindungslosen IP

7.1 TCP (3)



- nimmt Datenströme von Applikationen entgegen und teilt diese in max. 64kB große Blöcke auf, versieht diese mit einem Header und gibt sie an die IP Schicht weiter
- implementiert eine Zustellbestätigung und versendet bei Bedarf verlorengegangene oder beschädigte Pakete noch einmal
- stellt die richtige Reihenfolge der Pakete sicher

7.1 TCP (4)



- da mehrere Datenströme von einem Host möglich sein sollen erfolgt die Definition von speziellen Endpunkten: Sockets
- Jedes Socket hat eine Socketnummer, zusammengesetzt aus IP-Adresse und Portnummer
- Jedes Socket kann mehrere Verbindungen verwalten, solange der Socket der Gegenseite eindeutig ist

7.1 TCP (5)



- weitere Ein-Bit-Felder im Header für diverse Flags:
- Verbindungszustand
- Flusskontrolle
- Verbindungskontrolle

7.2 UDP (1)



- wesentlich weniger Overhead als TCP
- dafür keinen geordneten Verbindungsauf- und abbau
- Headergröße: 8Byte
- einfaches verbindungsloses Protokoll der Transportschicht

7.2 UDP (2)



- verwendet ebenfalls Ports
- jedes Paket wird mit einer Prüfsumme versehen
- keine Bestätigungsmeldung und keine Prüfung der Reihenfolge
- Hauptanwendung in lokalen Netzen, die sich i.d.R. durch geringe Fehlerraten auszeichnen

7.3 Ports (1)



- weiteres Adressierungsmerkmal neben IP-Adresse
- durchnummertiert mit 16 Bit, d.h. 65536 Ports
- bestimmte Dienste verwenden feste Ports (“well-known ports”)

7.3 Ports (2)



- Adressen sind Teile von IP
- Ports sind Teile von TCP und UDP
- es gibt also keine IP-Ports

8 Linux im Netzwerk



- 8.1 Einleitung
- 8.2 Manuelle IP-Konfiguration unter Linux
- 8.3 Überprüfung der Konfiguration

8.1 Einleitung



- Linux kann in so ziemlich jedem Netzwerk betrieben werden:
 - AppleTalk, IPX/SPX, Decnet ...
- Im folgenden liegt der Schwerpunkt bei IP

8.2 Manuelle IP-Konfiguration



- Konfiguration mittels
 - `/sbin/ifconfig`
 - `/sbin/route`

8.2.1 /sbin/ifconfig (1)



- `ifconfig` ohne Parameter liefert eine Liste der konfigurierten Interfaces
- die Ausgabe sollte selbsterklärend sein

8.2.1 /sbin/ifconfig (2)



- Konfiguration (Beispiel):

```
ifconfig eth0 134.76.82.141 netmask  
255.255.255.0
```

- erste Netzwerkkarte (eth0)
- IP Adresse 134.76.82.141
- Netzmaske 255.255.255.0
- weitere Optionen in der man-page

8.2.2 /sbin/route (1)



- Kernel unterhält eine Routing-Tabelle
- Abfrage erfolgt mit `route` ohne Parameter

```
mathpc00:~/> /sbin/route
```

```
Kernel IP routing table
```

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
134.76.82.0	*	255.255.255.0	U	0	0	0	eth0
default	mail.uni	0.0.0.0	UG	0	0	0	eth0

- der Parameter `-n` unterbindet die Namensauflösung

8.2.2 /sbin/route (2)



- route zum Ändern der IP Routing Tabelle des Kernels
- route add Ziel ... zum hinzufügen von Routen
- route del Ziel ... zum löschen einer Route
- Beispiel: Default Gateway setzen:

```
route add default gw 134.76.82.10
```

8.3 Überprüfen der Konfig.



Der Erfolg der Konfiguration wird über die Erreichbarkeit anderer Netze geprüft:

- ping
- traceroute

8.3.1 /bin/ping



ping sendet im Sekundenabstand Pakete an einen anderen Host, der dann ein einfaches Antwortpaket zurücksendet.

- aber: kein ping heisst nicht, dass das Netz nicht funktioniert.
- Interessante Parameter:
 - -c Zahl Anzahl der zu sendenden Pakete
 - -s Zahl zu verwendende Paketgröße
 - -f Floodping

8.3.2 /sbin/traceroute



traceroute verfolgt den Weg von Paketen im Netz.

- traceroute host liefert eine Liste der Router, die das Paket der auf dem Weg zum Zielhost passiert

9 Netzwerkanwendungen



- 9.1 Motivation
- 9.2 DHCP
- 9.3 Linux als Router
- 9.4 Linux als Firewall

9.1 Motivation



- Administrationsaufwand betreiberseitig
- Navigationsaufwand benutzerseitig
- Notwendigkeit zur Kennzeichnung von Diensten und Maschinen
- Folge sind “Low-Level-Anwendungen” wie DHCP, DNS u.a.

9.2 DHCP



Dynamic Host Control Protocol

- 9.2.1 Automatische IP-Zuweisung
- 9.2.2 Implementation
- 9.2.3 DHCP Server
- 9.2.4 Benutzerdefinierte Optionen
- 9.2.5 Vendor Code Identifier
- 9.2.6 DHCP Client

9.2.1 Automatische IP-Zuweisung



- Netzwerkprotokoll der Anwendungsschicht
- übermittelt grundlegende Daten zur Client Konfiguration
- kann zur automatischen, dynamischen IP-Zuweisung verwendet werden.
- benutzt spezielle IP-Adressen:
0.0.0.0 (Host ohne IP-Information) und
255.255.255.255 (lokaler Broadcast)
- großer Vorteil: Konfiguration erfolgt zentral

9.2.2 Implementation (1)



- Weiterentwicklung von BOOTP
- Voraussetzung: Broadcastfähiges Netz (also z.B. Ethernet oder TokenRing)
- verwendet UDP
 - Port 67 zur Anfrage an Server
 - Port 68 zur Antwort an den Client
- wird von nahezu allen Clientsystemen unterstützt

9.2.2 Implementation (2)



Ablauf

- Client schickt DHCPDISCOVER an 255.255.255.255 Port 67
- Server antwortet mit einem Vorschlag mit Parametern als DHCPOFFER
- Client sucht sich aus evtl. mehreren Angeboten das passende aus und schickt einen DHCPREQUEST an den entsprechenden Server
- Server bestätigt mit DHCPACK oder lehnt ab mit NAK

9.2.3 DHCP-Server (1)



- Server unterhält eine Liste der zur Zeit von ihm vergebenen Adressen (Leases)
- im Folgenden: Beispielimplementation des ISC
- `/var/lib/dhcp/dhcp.leases`
- Konfiguration in `/etc/dhcpd.conf`
- `man dhcp-options`

9.2.3 DHCP-Server (2)



- Umfang von `dhcpd.conf` hängt vom Einsatzziel ab:
- von einfacher IP-Vergabe im einfachsten Fall
- bis zu kompletter Konfiguration

9.2.4 Benutzerdefinierte Optionen



- Möglichkeit, sog. Vendoroptionen aufzunehmen
 - eigene Optionen zu den bereits vorhandenen
 - Codenummern 128-255
 - bei umfangreichen Optionen sollte die Paketgröße erhöht werden

9.2.5 Vendor-Code-Identifizierer



- mit VCI lassen sich DHCP-Anfragen verschiedener Rechner voneinander differenzieren
- Vendor-Code-Identifizierer sind feste Optionen für DHCP:
 - vendor-class-identifizierer für die Identifizierung des Clients durch den Server
 - vendor-encapsulated-options zur Identifizierung des Servers durch den Client

9.2.6 DHCP-Client



- Konfiguration in `/etc/dhclient.conf`
- legt fest:
 - welche Informationen zwingend erforderlich sind
 - nach welchen Kriterien ein DHCP OFFER ausgewählt wird
 - welches Programm die erhaltenen Informationen weiterverarbeiten soll

9.3 Linux als Router



- Router zwischen verschiedenen Teilen des LAN oder als Gateway mit Modem-, DSL- oder ISDN Uplink
- Überlicherweise mehr als ein Interface
- Dem Kernel muss das Weiterleiten der Pakete (Forwarding) erlaubt werden:
 - Schnittstelle hierfür ist der Kernel Filesystem:
`/proc/sys/net/ipv4/conf/INTERFACE/forwarding`

9.4 Linux als Firewall



- Kernel bringt grundlegende Firewallfunktionalität mit:
 - ab Kernel 2.0: ipfwadm
 - ab Kernel 2.2: ipchains
 - ab Kernel 2.4: iptables
- Umfangreiches Thema \Rightarrow eigenes Kapitel