TCP/IP

Autor: Steffen Dettmer (*steffen@dett.de*)
Layout: Matthias Nuessler (*m.nuessler@web.de*)

TCP/IP ist das Internetprotokoll, das auch von Linux verwendet wird. Dieses Kapitel beschäftigt sich mit TCP/IP Grundlagen.

Inhaltsverzeichnis

1 Einleitung

"TCP/IP" ist einer dieser typischen Begriffe, die im Zusammenhang mit Netzwerken immer wieder auftauchen, mit denen man aber selten etwas anzufangen weiß. Diesen Begriff zu klären soll Ziel der folgenden Abschnitte sein. Hier wird mit "TCP/IP" immer die Protokollversion 4 gemeint.

Als erstes ist TCP/IP eine Abkürzung für Transmission Control Protocol / Internet Protocol, was die Deutung auch nicht unbedingt erleichtert. Allerdings scheint es sich hier um ein Protokoll zu handeln. Genauer gesagt handelt es sich nicht um ein Protokoll, sondern um eine ganze Gruppe von Netzwerk- und Transportprotokollen, die unter dem Begriff TCP/IP zusammengefaßt werden. Zu TCP/IP gehört so immer das Protokoll ICMP und auch UDP wird von IP Implementationen grundsätzlich angeboten.

Genau diese Gruppe von Protokollen bildet die Grundlage für alle Netzwerkaktivitäten unter Linux bzw. UNIX. Doch auch unter anderen Betriebssystemen findet es immer mehr Verbreitung (schließlich baut das gesamte Internet auf TCP/IP auf).

TCP/IP entstand mit dem Internet seit 1983 und hat mehrere Ziele:

* Hardwareunabhängigkeit
* keine zentrale Verwaltung, kein zentraler Knoten, dadurch höhere Ausfallsicherheit
* applikationsunabhängig, es kann also für verschiedenste Zwecke genutzt werden

Mit dem Begriff Internet wird das größte zusammenhängende IP Netzwerk bezeichnet. Das Internet ging aus einem ARPANET, später DARPANET genannten Forschungsnetz hervor, dessen Grundlagen 1969 erarbeitet wurden.

TCP/IP ist anders als das ISO/OSI-Schichtenmodell nicht offiziell standardisiert, Aufbau der Protokolle wird in den sogenannten RFC's beschrieben (Request for Comment). Verfügbar sind diese z.B. unter http://www.rfc-editor.org.

Da es von der Hardware unabhängig ist, kann es auf den verschiedensten Netzwerktypen eingesetzt werden. Doch bevor wir uns detailliert den einzelnen Schichten von TCP/IP zuwenden, wollen wir uns damit beschäftigen, wie sich die Rechner im TCP/IP-Netz gegenseitig ansprechen.

2 Adressierung im IP

Jeder Host im Netzwerk erhält eine eindeutige Adresse, die sogenannte IP-Adresse oder IP-Nummer. Das 'IP' in der Bezeichnung deutet darauf hin, daß hierfür das Internet Protocol zuständig ist.

In der aktuellen Version von TCP/IP wird jedem Rechner eine 32 Bit lange Adresse zugeordnet, die der besseren Lesbarkeit halber in der Form `xxx.xxx.xxx.xxx` geschrieben wird. Hierbei werden die einzelnen Bytes durch Punkte getrennt. So sind IP-Adressen von 0.0.0.0 bis 255.255.255.255 möglich. Für die Verwendung durch Hosts nicht alle Nummern nutzbar, einige z.B. sind für die Netzadresse oder sogenannte Broadcasts reserviert. Mehr dazu weiter unten.

Eine typische IP-Adresse könnte nun lauten 141.89.64.1 (ist in diesem Fall der Nameserver der Universität Potsdam). In Binärschreibweise wäre dies `10001101 1011001 1000000 00000001` Eine IP-Adresse gliedert sich nun in einen vorderen und einen hinteren Teil. Der vordere Teil stellt die Netzadresse dar, der hintere Teil die Host-Adresse. Die IP-Adresse bezeichnet das Netz, in dem sich ein Rechner befindet und seine Nummer in diesem Netz. Die Trennung dieser beiden Teile kann prinzipiell an jedem Bit der Adresse erfolgen. Dazu benötigt man aber noch die sogenannte Netzmaske. Diese Netzmaske ist genauso lang wie die IP-Adresse und wird bis zu einem bestimmten Bit mit Einsen gefüllt. Der Rest wird auf Null gesetzt. Alle Bits in der IP-Adresse, die in der Netzmaske belegt sind, zählen dann zum Netzanteil, der Rest zum Hostanteil.

Lassen wir der Theorie einige Beispiel folgen und zwar in der nächsten Abbildung:

Beispiel

```
[1]         Binär                                Dezimal
IP-Adresse 10001101 1011001 01000000 00000001 141.89.64.1
Netzmaske 11111111 1111111 00000000 00000000     255.255.0.0
                ---------------- ----------------      ------- ---
                    Netz-ID     Host-ID            Netz Host

[2]         Binär                                Dezimal
IP-Adresse 00111010 00010001 10000011 00101100 58.17.131.43
Netzmaske 11111111 11111111 11111111 00000000     255.255.255.0
                ----------------------- --------      ------- ---
                    Netz-ID          Host-ID  Netz Host

[3]         Binär                                Dezimal
IP-Adresse 01111011 00000101 01100100 00000010 123.5.100.2
Netzmaske 11111111 11111111 11110000 00000000     255.255.240.0
                ----------------------- --------      ------- ---
                    Netz-ID          Host-ID  Netz Host
```

Im ersten Beispiel verwenden wir als Netmask 255.255.0.0, d.h. die ersten 16 Bit der Adresse sind dem Netzteil vorbehalten, hier 141.89, die restlichen 16 Bit werden zur Adressierung des Hosts in diesem Netz verwendet, dazu bleiben 254*254, also gut 64.516 Möglichkeiten - praktisch können also über 64.000 Hosts in diesem Netz adressiert werden. Die Adresse 141.89.0.0 ist für das Netzwerk selbst reserviert, die Adresse 141.89.255.255 bezeichnet die sogenannte Broadcast-Adresse: Pakete an diese Adresse werden an alle Stationen im Netz verschickt.

Doch fahren wir mit unseren Beispielen fort. Beispiel Nummer zwei beschreibt den Host 58.17.131.43 mit der Netmask 255.255.255.0. Hier sind also 24 Bit für den Netz- und 8 Bit für den Hostanteil reserviert. Theoretisch kann es in diesem Netz maximal 254 Hosts geben. Adresse dieses Netzwerk ist 58.17.131.0, Broadcasts gehen an 58.17.131.255.

Das dritte Beispiel zeigt nun, daß man sich bei der Netzmaske keinesfalls auf ganze Bytes beschränken muß. Bei dieser Adresse macht der Netzanteil 20 Bit aus, der Hostanteil 12 Bit. Die Netzadresse ist hier 123.5.6.0 (bis zum 20. Bit), Broadcasts gehen an 123.5.112.255 (Alle Bits ab dem 21. sind auf 1 gesetzt).

Die verschiedenen Netzmasken lassen sich in verschiedene Klassen unterteilen: Class-A-Netz haben die Netmask 255.0.0.0, Class-B-Netze die Netmask 255.255.0.0 und Class-C-Netze die Netmask 255.255.255.0. Netze mit anderen Netmasks bezeichnet man als Subnetze.

Man sieht, daß eine Netzmaske mit "1" beginnen und mit "0" aufhören muß. Eine Netzmaske "11111111000000011111110000000" macht keinen Sinn, da der Netzwerkanteil hier nicht geschlossen auftauchen würde; es gibt ja zwei Grenzen zwischen Netz- und Hostanteil! Eigentlich beschreibt die Netzmaske also eine Grenze an einer bestimmten Bitposition. Diese Grenze kann man auch einfach als eine Zahl angeben. Dazu zählt man einfach die "1". Aus "11111111111111111111110000000" bzw. 255.255.255.0 wird dann kurz "24" und aus 255.255.0.0 wird dann "16". Üblich ist folgende Notation: man schreibt die kurze Netzmaske durch einen Schrägstrich getrennt hinter eine Netzwerkadresse. Aus 141.89.0.0, Netzmaske 255.255.0.0 wird dann kurz 141.89.0.0/16 oder noch kürzer 141.89/16 (durch die Netzmaske ist ja klar, daß hinter der Netzwerkadresse nur noch "0" kommen können).

Heute verwendet man jedoch weitgehend **CIDR**, Classless Internet Domain Routing. Das heißt, man ist nicht mehr auf Netzklassen A, B und C festgelegt, sondern kann auch kleinere Netze verwenden. Dadurch werden Netzmasken wie zum Beispiel 255.255.255.252, kurz 30 verwendet, die ein Netz mit lediglich 4 IP Adressen bezeichnet, das kleinste sinnvolle Netz (2 IP ist wenig sinnvoll, da ja die erste [kleinste] Adresse das Netzwerk, und die größte für Broadcast reserviert ist, so daß hier keine [0] Adresse nutzbar wäre). Bei Standleitungen bekommt man in Deutschland häufig 8 IP Adressen, Netzmaske ist demzufolge 255.255.255.248, kurz 29. Von diesen sind dann 6 nutzbar.

Das Kopfrechnen ist hier unübersichtlich. Bei der Netzmaske in Langform erhält man die Zahl der IP Adressen, in dem man schaut, wieviel zwischen Netzmaske und 255.255.255.255+1 übrig bleibt. Das + 1 kommt daher, daß die letzte Position selbst auch Teil des Netzes ist: 256-252=4, 256-248=8. Mit der kurzen Netzmaske ist das etwas anders. Man subtrahiert diese zunächst von 32 und erhält die Anzahl der Bits für den Hostanteil (also die möglichen Adressen): 32-30=2, 32-29=3. Um von Bits auf mögliche Werte zu kommen, muß man nun (wie immer) "zwei hoch Bit" rechnen, also 2^2=4, 2^3=8. Man kommt hier natürlich auf das gleiche Ergebnis, wie wenn man die Langform verwendet.

Bei der Kurzform kann man übrigens keine ungültigen Netzmasken angeben, bei der Langform hingegen schon. Man überlege sich die Netzmaske 255.255.255.127. Vielleicht ist das auf den ersten Blick nicht erkennbar, aber binär ist das ja "1111111 111111 111111 0111111". Man sieht eine "0" in der "Mitte". Diese Netzmaske kann man gar nicht in Kurzform schreiben. Genauer gesagt, ist diese Angabe überhaupt keine Netzmaske.

Natürlich kann man sich IP-Adressen nicht einfach ausdenken, denn die Netzwerke, mit denen man Daten austauschen möchte, müssen diese Adresse ja erreichen können, also den Weg zu dieser Adresse finden. Wählt man sich über einen Internet Provider in das Internet ein, so erhält man von diesem beispielsweise eine Adresse zugewiesen (man bekommt meistens bei jeder Anwahl eine neue und nennt diese Adressen daher "dynamisch"). Mietet man sich eine Standleitung, erhält man in der Regel gleich mehrere IP-Adressen. Diese Adressen "gehören" meistens weiterhin dem Provider, jedoch wird der Endkunde meistens in einer öffentlich zugänglichen Datenbank eingetragen.

2.1 Adressen für private Netze

Nun wollen wir mit unserem Netz aber gar nicht fest an das Internet angebunden sein. Dann steht uns die Wahl der Netzadressen im Grunde genommen frei, dies kann aber später zu Problemen führen. Stellen wir uns vor, wir möchten ein Netzwerk mit der Adresse 141.89.0.0 aufbauen. Soweit kein Problem, unsere Rechner bekommen IP-Nummern von 141.89.1.1 bis 141.90.254.254 mit Netzmaske 255.255.0.0 und alles sollte funktionieren. Wenn dieses Netzwerk allerdings irgendwann in das Internet eingebunden wird kommt es zwangsläufig zu Konflikten mit Hosts irgendwo da draußen.

Abhilfe schaffen hier die sogenannten privaten Adressräume. Hierbei handelt es sich um Bereiche von IP-Adressen die im Internet nicht geroutet werden (dürfen) und für LANs frei verwendet werden können. Diese werden in RFC 1918 definiert. Im einzelnen sind dies:

* 1 Class-A-Netz 10.0.0.0 ... 10.255.255.255
* 16 Class-B-Netze 172.16.0.0 ... 172.31.255.255
* 256 Class-C-Netze 192.168.0.0 ... 192.168.255.255

Da heute "classless" geroutet wird (CIDR), notiert man diese meistens:

* 10.0.0.0/8
* 172.16.0.0/12
* 192.168.0.0/16

Letzteres zum Beispiel kann man sich dann in 256 C-Netze oder in 512 Netze mit je 128 Adressen oder auch in verschiedene unterschiedlich große aufteilen.

3 Protokollschichten

Genauso wie das OSI-Schichtenmodell verfügt die TCP/IP-Protokollsuite über mehrere Schichten bzw. Layer, durch Zusammenfassung einzelner Ebenen allerdings im Gegensatz zu sieben Schichten im OSI-Modell nur über vier. Im einzelnen handelt es sich hier um den Link Layer, Network Layer, Transport Layer und Application Layer. Die folgende Abbildung zeigt diese Schichten und die darin angesiedelten Protokolle, welche wir im folgenden detailliert behandeln werden. Das TCP/IP Link Layer faßt OSI Physical Layer und OSI Data Link Layer zusammen, da aus Sicht von IP die darunter liegenen Protokolle uninteressant sind.

TCP/IP selbst definiert Network und Transport Layer. Es benötigt also ein darunter liegendes Protokoll (oder mehrere), die Link und Network Layer bereitstellen. TCP/IP stellt an dieses keine hohen Anforderungen. Es muß lediglich möglich sein, Blöcke einer bestimmten Größe (wenige hundert Byte reichen bereits für sinnvollen Betrieb) von einem Geräte zu einem anderen übertragen zu können. Protokolle wie zum Beispiel Ethernet definieren diese Schichten.

4 Protokolle im Link Layer

Der Link Layer faßt die Hardware-Schicht zusammen, hier tummeln sich alle Netzwerkmedien, über die man TCP/IP betreiben kann. Durch seine hohe Hardwareunabhängigkeit sind dies natürlich eine ganze Menge. Auf die verschieden Hardwareprotokolle soll an dieser Stelle nicht detailliert eingegangen werden. Häufig wird TCP/IP über ein Ethernetprotokoll gefahren. Interessant ist daher vielleicht in Verbindung mit dem Ethernet das ARP-Protokoll, dazu gleich mehr.

Aus Sicht von TCP/IP ist Ethernet das Link Layer. Aus Sicht des OSI Referenzmodelles sind das (etwa) Physical Layer und Data Link Layer.

Ethernet kann Pakete (die man hier meistens Frames, Rahmen nennt) von einem Host (die man im Ethernet Sprachgebrauch meistens als Stationen bezeichnet) zu einem unmittelbar angeschlossenen übertragen. Dieses Protokoll ist umständlich: die Paketgröße ist stark begrenzt (üblich sind 1500 Bytes), und alle Hosts müssen direkt aneinander geschlossen sein. Die maximale Paketgröße nennt man **MTU**: maximum transfer unit. Direkt angeschlossen bedeutet hier, das sie am gleichen Bus sind (Cheap Ethernet), über einen Hub oder Switch verbunden sind. Ein Hub ist im Prinzip nur ein Signalverstärker, der das Signal eines Frames einer Station an alle anderen angeschlossenen weitersendet. Einen Switch kann man sich als intelligenten Hub vorstellen, denn der sendet das Signal nur an die Station oder Stationen, für die es bestimmt ist. Ein Switch muß dazu im Gegensatz zu einem Hub das Ethernetprotokoll verstehen, denn er muß in die Frames hineinschauen und die Zieladresse verstehen und sich in Tabellen merken, an welchen Anschluß (Port) diese Station angeschlossen ist.

Damit das Ethernet weiß, an welche Station ein Paket gesendet wurde, werden die Daten des Frames um einen Header erweitert. Das kann man mit einem Briefumschlag vergleichen, der um ein Anschreiben herum "gepackt" wird. In diesem Header steht beispielsweise Absender- und Empfängeradresse.

4.1 ARP - Adress Resolution Protocol

Jede Ethernet-Karte besitzt eine eindeutige Hardware-Adresse, auch als **MAC-Adresse** bezeichnet. Jedem Host im Netz wurde aber nun auch noch eine IP-Adresse zugeordnet und damit hat das Ethernet ein Problem: Mit der IP-Adresse 192.168.23.1 weiß es nichts anzufangen, mit der MAC-Adresse `08:00:00:04:72:98` allerdings schon. Ein Übersetzer muß her: Das Adress Resolution Protocol (ARP) übersetzt die IP-Adressen in Ethernet-Adressen. So gesehen gehört es eigentlich auch nicht vollständig in den Link Layer, sondern ist irgendwo zwischen Link und Network Layer einzuordnen.

Möchte nun eine Station die Hardware-Adresse des Hosts 192.168.20.11 herausbekommen, schickt sie einen ARP-Request ins Kabel: "Wer hat hier IP 192.168.20.11?". Der betreffende Host antwortet: "Zu IP 192.168.20.11 gehört MAC-Adresse 08:00:a2:12:4d:aa". Damit weiß der sendende Rechner, wohin er sein Ethernet-Paket zu schicken

hat.

5 Protokolle im Network Layer

Diese Schicht ist fast ausschließlich dem Internet Protocol vorbehalten, zur Kontrolle des Datenflusses gibt es dann noch ein Internet Control Message Protocol.

5.1 IP - Das Internet Protocol

IP benötigt irgendein Link Layer Protokoll, zum Beispiel Ethernet. Das IP erweitert die Paketvermittlung. Ein IP-Paket wird in gegebenenfalls mehrere Teile aufgeteilt, von denen jedes klein genug ist, um über das Link Layer Protokoll übertragen werden zu können. Diese Teile nennt man Fragmente.

IP selbst fügt den Daten wiederum einen Header hinzu, diesmal einen IP-Header. Dieser enthält unter anderem Absender- und Empfängeradresse; jetzt sind dies jedoch IP Adressen. Er enthält auch etliche weitere Informationen.

IP fügt noch eine weitere wichtige Eigenschaft hinzu: es kann über verschiedene "Zwischenstationen" Pakete ausliefern. Als Beispiel denke man sich einen Host A, der mit einem Host B verbunden ist, und einen Host C, der mit Host B verbunden ist. Über Ethernet können A und C nicht kommunizieren. Über IP funktioniert das, da IP in diesem Fall ein an C adressiertes IP Paket in ein oder mehrere an B adressierte Ethernetframes einpackt.

Die Daten sind hier also doppelt verpackt: Um die Daten herrum sind die Zusätze des IP-Frames (also beispielsweise der IP-Header mit den IP-Adressen), und um das IP-Frame herum sind die Zusätze des Ethernet-Pakets (also beispielsweise die Ethernet-MAC-Adressen).

Diese Frames werden dann von B empfangen. Die IP Implementation von B wirft die Zusätze des Ethernetframes weg, nach dem diese verarbeitet wurden. Übrig bleibt das IP-Frame (und natürlich die Daten). Nun erkennt B an der IP Adresse, daß das Paket an C gesendet werden soll, und packt es daher wieder in ein Ethernetframe, daß diesmal jedoch an C adressiert ist (B kann ja C erreichen) und sendet es. Diesen Vorgang bezeichnet man als Routing.

Durch das Zusammenspiel von Fragmentierung und Routing kann ein IP Paket in Form von mehreren Fragmenten beim Empfänger eventuell in falscher Reihenfolge ankommen. Das IP setzt diese Fragmente wieder ordentlich zusammen, und es entsteht wieder das IP Paket.

Dabei arbeitet das Protokoll verbindungslos und ungesichert, d.h. vor dem Versenden der Pakete wird keine explizite Verbindung zum Empfänger hergestellt, sondern diese werden einfach abgeschickt. Es existieren auf IP-Ebene keine Sicherungsmechanismen, die sicher stellen, daß die Daten auch korrekt beim Empfänger ankommen, lediglich der Header ist prüfsummengeschützt, um Übertragungsfehler zu erkennen. Die Daten selbst sind jedoch nicht prüfsummengeschützt. Dies wird gegebenfalls von höheren Schichten erledigt.

Die weiter vorn beschriebene Adressierung in TCP/IP-Netzen wird ebenfalls durch das IP gehandhabt.

5.2 ICMP - Internet Control Message Protocol

Das ICMP dient zur Übermittlung von Fehler- und Statusinformationen. ICMP wird auch von IP selbst verwendet, um beispielsweise Fehler zu melden. ICMP ist daher zwingend in IP vorhanden; funktional gesehen ist es Teil von IP, technisch setzt es jedoch auf IP auf. ICMP kann aber auch direkt von Applikationen verwendet werden.

ICMP Pakete werden verwendet, wenn beispielsweise der Zielrechner oder das Zielnetzwerk nicht erreichbar ist. In diesem Fall gibt ein TCP/IP Router beispielsweise ein 'host unreachable' bzw. 'network unreachable' Paket zurück. ICMP ist im Network Layer angesiedelt, ist aber eigentlich kein wirklich eigenständiges Protokoll, da die Übermittlung von ICMP-Nachrichten durch IP-Pakete erfolgt und dazu dient, die Übertragung von den eigentlichen Daten zu steuert. ICMP ist damit kein Datenfluß sondern ein Kontrollfluß, welches den Datenfluß steuert.

Eine recht bekannte Anwendung von ICMP ist das Programm `ping`, mit dem man die Erreichbarkeit eines anderen

Rechners prüfen kann. Dazu wird ein ICMP echo request an den Zielrechner geschickt, dieser sendet ein ICMP echo reply zurück, wenn er denn verfügbar ist.

Hier wollen wir auch mal eine Theorie-Pause machen und ein bißchen pingen. Ping wird aufgerufen an der Shell mit dem Befehl ping und danach der anzupingenden Adresse. Abgebrochen wird die Pingerei mit `STRG + C`.

```
user@linux / # ping 192.168.102.1

PING 192.168.102.1 (192.168.102.1) from 192.168.102.1 : 56(84) bytes of data.
64 bytes from 192.168.102.1: icmp_seq=0 ttl=255 time=0.4 ms
64 bytes from 192.168.102.1: icmp_seq=1 ttl=255 time=0.2 ms
64 bytes from 192.168.102.1: icmp_seq=2 ttl=255 time=0.2 ms
--- 192.168.102.1 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 0.2/0.2/0.4 ms
```

Ist ein Rechner nicht erreichbar erscheinen keine Antwortmeldungen.

ICMP wird meistens von IP für den Benutzer unsichtbar verwendet, zum Beispiel um herauszubekommen, wie groß ein Paket sein kann, welches von Link Protokoll übertragen werden kann, oder um Fehler zu melden.

6 Protokolle im Transport Layer

6.1 UDP - User Datagram Protocol

Das UDP ist das simpelste Protokoll auf der Transportebene. Es stellt im Gegensatz zum noch folgenden TCP nicht sicher, ob die versendeten Pakete auch wirklich beim Empfänger ankommen, sondern sendet auf gut Glück, bzw. nach dem Motto 'fire and forget'. Dafür ist es sehr schnell und erzeugt nicht so eine hohe Netzlast wie sein großer Bruder TCP. Somit eignet es sich gut für Anwendungen, die kleine Datenmengen schnell austauschen, wie verteilte Dateisysteme (NFS) oder Nameserveranfragen (DNS).

Da über IP nur eine Station adressierbar ist, könnte man zunächst ja nur eine Verbindung von Station zu Station verwenden. UDP fügt daher sogenannte Portnummern ein. Diese unterscheiden verschiedene logische Kanäle, die wiederum Diensten oder Anwendungen zugeordnet werden können. UDP ermöglicht 65535 verschiedene Portnummern, also theoretisch 65535 unterscheidbare UDP Kanäle. Die meisten Hosts unterstützen jedoch weit weniger gleichzeitig, beispielsweise 1024 oder 4096. Diese Informationen (Absender- und Empfängerport) werden wieder in einem Header vermerkt, dem UDP-Header.

Zusätzlich zum reinen IP-Paket besitzt ein UDP-Paket also noch einige wenige zusätzliche Daten wie Portnummer und Prüfsumme über die Nutzdaten (und den Header).

6.2 TCP - Transmission Control Protocol

Das Transmission Control Protocol ist im Gegensatz zu den bisher beschriebenen Protokollen ähnlich wie das Telefonnetz verbindungsorientiert aufgebaut. Es verwendet IP, um Daten zu übertragen. Bevor man irgendwelche Daten per TCP an einen anderen Rechner senden kann, muß man eine Verbindung zu diesem aufgebaut haben. Damit hat TCP die Haupteigenschaft, nicht mehr einzelne Pakete, sondern Datenströme zu verarbeiten. Im Gegensatz zu Paketen sind die Datenlängen hier nicht mehr begrenzt. Über eine TCP Verbindung können Gigabytes von Daten übertragen werden, die natürlich in Millionen von IP Paketen aufgeteilt werden.

Da alle darunter liegenden Protokolle allerdings paketorientiert arbeiten, ist die Verbindung natürlich nur 'virtuell'. Durch diese festgelegte Verbindung können die zwei Verbindungspartner auch sicherstellen, daß die gesendeten Daten auch wirklich korrekt ankommen und zwar auch in der richtigen Reihenfolge und nicht doppelt. Dadurch ist es natürlich wesentlich komplexer aufgebaut als UDP. In der Regel gibt es in der Verbindung einen Client und einen Server. So verwenden auch die meisten Internetanwendungen TCP als Übertragungsprotokoll.

Eine typische TCP-Verbindung sieht so aus, daß der Client eine Verbindungsanforderung an den Server schickt. Dieser bestätigt diese Verbindung, wenn er dies möchte. Er Client bestätigt diese Antwort. Damit besteht eine feste Verbindung zwischen den beiden Partnern. Da drei Schritte erforderlich sind, um eine Verbindung aufzubauen, bezeichnet man diesen Vorgang auch als three-way-handshake (dreifach Händeschütteln). Nun können fleißig Daten übertragen werden bis entweder der Server oder der Client die Zeit gekommen sehen, die Verbindung zu trennen. Dazu sendet er einfach eine Ende-Anforderung an den anderen Rechner und nach Bestätigung durch diesen wird die Verbindung beendet.

TCP erstellt nun für eine bestimmte Menge an Daten des Datenstromes ein (IP) Paket und sendet dieses über IP. Ebenso wie UDP kennt TCP Portnummern und verwendet einen eigenen Header. Wie auch bei IP können die TCP Fragmente in falscher Reihenfolge ankommen. Diese werden von TCP sortiert und zusammengesetzt, ähnlich, wie das auch IP macht. TCP numeriert die Pakete (diese Nummer steht im TCP Header), um dies zu ermöglichen. Im TCP Header steht auch eine Header- und Datenprüfsumme.

Paketbestätigungen werden automatisch verschickt. Eine Paketbestätigung ist ein spezielles TCP Paket. Eine Bestätigung gilt immer für eine bestimmte Paketnummern. Kommt innerhalb einer bestimmten Zeit keine Bestätigung, so ist entweder das ursprüngliche Paket oder dessen Bestätigung verlorengegegangen. Verlorengegangene Pakete werden von der TCP Schicht automatisch wiederholt. Der Empfänger merkt sich die gefolgten Pakete, bis er die

fehlenden erhalten hat. Dann erst kann er den Datenstrom wieder korrekt zusammensetzen.

Wird TCP verwendet, wird also sichergestellt, daß die Daten so ankommen, wie sie gesendet werden. Ein TCP Dienst muß also keine Datenprüfungen durchführen.

Zum Verbindungsabbau werden wieder spezielle Pakete verwendet, analog zum Verbindungsaufbau.

6.3 Ports

Ports wurden schon kurz erwähnt, aber kaum erklärt. Dies soll an dieser Stelle nachgeholt werden.

Über das TCP und UDP Protokoll können Daten zu einem anderen Rechner übertragen werden. Nun hat man aber häufig mehrere Dienste auf einem Rechner, und möchte gleichzeitig mit mehreren Diensten kommunizieren können. Da also ein Rechner in der Regel mehr als einen Dienst anbietet (z.B. FTP, Telnet, POP, ...), muß man neben der IP-Adresse ein weiteres Adressierungsmerkmal finden. Dies sind die sogenannten Ports. So erreicht man z.B. den Dienst FTP auf einem Rechner in der Regel über TCP Port 21, Telnet läuft über TCP Port 23, DNS auf UDP Port 53. Hinter jedem Port steht auf dem Rechner ein Prozeß, der auf Anfragen wartet, hinter Port 21 entsprechend der FTP-Daemon. Solche üblichen und allgemein bekannten Ports nennt man "well-known ports".

Man sollte die Informationen nicht verwechseln. Adressen sind Teile von IP. Ports sind Teile von UDP und TCP. Es gibt damit also keine IP-Ports, sondern nur UDP Ports und TCP Ports.

In der Datei `/etc/services` sind etliche "well-known ports" aufgeführt.

7 Protokolle im Application Layer

Wie bereits angedeutet, ist der Begriff Application Layer vom gleichnamigen Begriff des OSI Referenzmodelles zu unterscheiden. Hiermit wird in etwa die Zusammenfassung der OSI Layer Session, Presentation und Application gemeint.

Viele Protokolle setzen auf TCP oder UDP auf. Beispiele für UDP basierte Protokolle sind zum Beispiel DNS (DomainNameService), verschiedene Windows-Protokolle und RIP (Routing Information Protocol). Beispiele für TCP basierte Protokolle sind HTTP (Hyper Text Transfer Protocol), FTP (File Transfer Protocol) und SSH (Secure Shell).

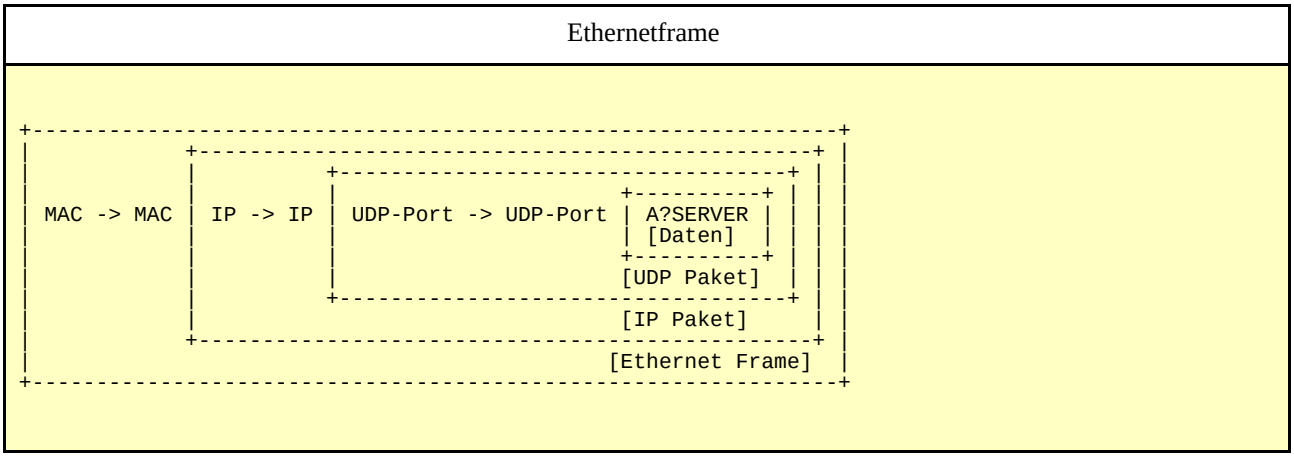
8 Beispiel

Zum Abschluß soll ganz kurz eine Verbindung erklärt werden. Angenommen, es gibt mehrere Ethernet-Netzwerke. Weiterhin einen Host CLIENT, der an einen IP-Router ROUTER1 angeschlossen ist. Dieser Router hat eine zweite Netzwerkkarte. In diesem zweiten Ethernet-Netzwerk steht ein DNS Server und ein Router ROUTER2. An dessen anderem Netz steht ein Server SERVER, der Webseiten über HTTP anbietet. Die folgende Beschreibung ist natürlich stark vereinfacht.

Auf CLIENT läuft eine Anwendung, die Webseiten von SERVER holen soll (es könnte sich um einen Browser handeln). Der CLIENT weiß, daß es dazu HTTP verwenden muß, und daß dieses Protokoll auf TCP aufsetzt. Er weiß weiterhin, daß er dazu die IP-Adresse von SERVER kennen muß, um das Paket adressieren zu können.

Zunächst muß CLIENT also die IP-Adresse von SERVER herauskriegen. Dazu kennt er das DNS-Protokoll, welches über UDP verwendet wird. CLIENT sieht in /etc/services nach, und erkennt, daß DNS (domain) Port 53 verwendet. Er sieht in /etc/resolv.conf nach, und kennt die IP-Adresse des Nameservers. Er sendet nun ein UDP Paket mit der Anfrage nach der Adresse von SERVER. Um das UDP Paket verschicken zu können, muß IP einen Router verwenden, da IP an Hand der IP Adresse erkennt, daß der DNS Server in einem anderen Netzwerk liegt. Hierzu schaut IP nun in der sogenannten Routingtabelle nach, und stellt fest, daß ROUTER1 verwendet werden muß, und das Netzwerk zu erreichen. CLIENT packt das IP Paket mit dem UDP Paket (welches wiederum die eigentliche Anfrage enthält) in ein Ethernetframe. Über ARP bekommt CLIENT heraus, wie die Ethernetadresse (MAC Adresse) von ROUTER1 ist (bisher kennt CLIENT ja aus der Routingtabelle nur dessen IP Adresse). Diese setzt er dann in das Ethernetframe ein. Das Frame sieht dann wie folgt aus:

Ethernetframe



In diesem Beispiel sieht man auch gut, wieviel zusätzliche Daten benötigt werden! Die hier dargestellten Kästchen entsprechen den Layern des TCP/IP Modells (Daten gehören zum Application Layer).

Der ROUTER1 empfängt dieses Paket, da er die MAC Zieladresse ist. Er entpackt das Ethernetframe und reicht dessen

Inhalt an IP weiter. IP erkennt, daß das Paket an DNS-Server adressiert ist. IP erkennt auch, daß DNS-Server im selben Netz liegt, und das kein Router verwendet werden muß. Über ARP wird die MAC Adresse vom DNS-Server bestimmt, und das IP Paket wird in ein an diese MAC Adresse adressiertes Ethernetframe verpackt und verschickt.

DNS-Server empfängt das Ethernetframe und packt es aus. IP erkennt, daß das Paket nicht weiterverschickt werden muß. Es packt es aus und reicht es an UDP weiter. UDP erkennt an der Portnummer, daß es an den DNS Prozeß weitergereicht werden muß. Der DNS Prozeß erkennt eine Adressanfrage für SERVER. Er schaut die IP Adresse in seiner Datenbank nach und erzeugt ein UDP-Antwort-Paket. Das IP erkennt nun, daß wieder ROUTER1 verwendet werden muß und schickt ein entsprechendes Ethernetframe. ROUTER1 sendet das UDP/IP Paket über ein neues Ethernetframe wiederum weiter zu CLIENT. Der empfängt dies und gibt die Antwort letztlich an die Anwendung weiter.

So, nun kennt die Anwendung die Adresse von SERVER. Über `/etc/services` findet es TCP Port 80 für HTTP. Sie schickt ein TCP-Verbindungsaufbaupaket an SERVER, Port 80. IP erkennt, das ein Router verwendet werden muß, und findet ROUTER1 als zuständig. Diesem wird also ein Ethernetframe geschickt. Das IP erkennt hier jedoch, daß die Zieladresse (immer noch) in einem anderen Netzwerk liegt. Anhand der Routingtabelle findet IP heraus, daß ROUTER2 verwendet werden muß und macht eine entsprechende ARP Anfrage. Über die gelernte MAC Adresse wird nun das Paket zu ROUTER2 geschickt, der wiederrum eine ARP Anfrage für die MAC Adresse von SERVER macht, und letztendlich das Paket an diese verschickt.

Endlich hat SERVER ein Paket empfangen. Hier wird geschaut, ob es einen Dienst für Port 80 bekannt ist. In diesem Fall wird eine Bestätigung geschickt (ansonsten eine entsprechende Meldung als Paket verschickt). Ist SERVER überhaupt nicht erreichbar (ROUTER2 bekommt zum Beispiel keine Antwort auf seine ARP Anfrage), so würde ein ICMP Paket von ROUTER2 an CLIENT geschickt werden. Diese Antwort wird über ROUTER2 und ROUTER1 entsprechend an CLIENT geschickt, der sie bestätigt (über ROUTER1 und ROUTER2). Nun kann CLIENT anfangen, TCP Nutzdaten zu schicken. Die Pakete gelangen dann zu SERVER und werden nun an die Anwendung weitergereicht.

In der Praxis hat man oft nicht nur zwei, sondern 10 oder manchmal 20 und mehr Router auf dem Weg! Die Beschreibung hier hat viele viele Details weggelassen; in der Praxis ist der Vorgang weitaus komplexer.

Mit diesem Wissen ärgert man sich vielleicht nicht mehr so sehr über langsamen Webseitenaufbau, da man bedenkt, welche komplexen Vorgänge hier ablaufen...