IP-Netze

Michael Dienert

23. April 2020

Inhaltsverzeichnis

1 Begriffe Netz und Subnetz

Aus historischen Gründen gibt es den Begriff *Subnetz*. Er stammt aus der Zeit, in der das Internet in Netze der Grössen 1 677 721, 65 536 und 256 (Klasse A-, B-, C-Netze) aufgeteilt wurde. Diese Netze konnte man weiter unterteilen und gelangte damit eben zu *Sub*-Netzen.

Heute hat man diese Klasseneinteilung aufgegeben und man kann die Begriffe Subnetz und Netz synonym verwenden.

2 IP-Adressen

Eine IPv4-Adresse ist 32 bit = 4 Oktette lang und wird gewöhnlich in der *dotted decimal notation* geschrieben. Z.B.:

129.143.14.155

Dabei wird jedes Oktett der Adresse als *Dezimalzahl* angegeben und durch einen Punkt getrennt. Im Internet muss eine IP-Adresse eindeutig sein.

Da sich bei 32 bit $2^{32} = 4294967296$ Adressen ergeben, muss dieser Adressvorrat in kleinere Netze aufgeteilt werden.

Dazu wird die IP-Adresse zweigeteilt: der vordere Teil wird als Netzwerk-ID, der hintere Teil als Host-ID betrachtet (Abb. 1).

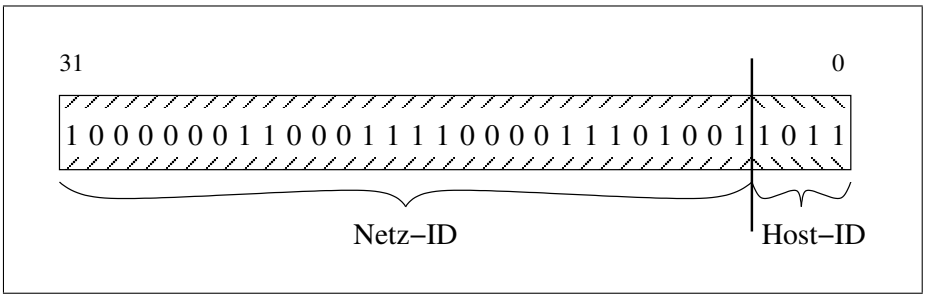

Abbildung 1: Netz- und Host-ID einer IP-Adresse

Mit Hilfe der *Netzmaske* wird festgelegt, wo diese Trennung verläuft, also wieviele Bits von vorne die *Netzwerk-ID* darstellen und aus wievielen Bits der *Host-Anteil* besteht.

Die Netzmaske ist ebenfalls 32 bit lang. Die höherwertigen Bits enthalten nur Einsen, die niederwertigen Bits nur Nullen. Z.B.:

1111 1111 . 1111 1111 . 1111 1111 . 1111 0000
255 . 255 . 255 . 240

Mit Hilfe der Netzmaske kann man zu jeder IP-Adresse die zugehörige Netzadresse berechnen. Dazu wird IP-Adresse und Netzmaske **bitweise UND-Verknüpft**. Das Ergebnis ist die Netzadresse:

129	**143**	**14**	**155**
1000 0001	**1000 1111**	**0000 1110**	**1001 1011**
1111 1111	**1111 1111**	**1111 1111**	**1111 0000**
1000 0001	**1000 1111**	**0000 1110**	**1001 0000**
129	**143**	**14**	**144**

In diesem Beispiel liegt die IP-Adresse also im Netz mit der Netzadresse 129.143.14.144.
Für IP-(Sub-)Netze gelten somit folgende wichtige Regeln:

Zwei IP-Adressen sind im gleichen Subnetz, wenn sie bezüglich derselben Netzmaske die gleiche Netzadresse haben.

Ein IP-Netz ist eindeutig durch die Angabe der Netzadresse ***und*** die Netzmaske bzw. den Präfix festgelegt.

2.1 Vereinfachte Adressberechnung

Um die Netzadresse auszurechnen muss man bei Netzen, deren Host-ID *maximal* 8 Bit lang ist nur das *letze* Oktett in Binärdarstellung umrechnen und mit der Maske UND-verknüpfen. Die oberen 3 Oktette sind in Netz- und IP-Adresse identisch.
Allgemein lässt sich eine Regel zur vereinfachten Berechung so formulieren:

- Das Oktett, innerhalb dem die Trennung verläuft muss in Binärdarstellung mit dem entsprechenden Oktett der Netzmaske UND-Verknüpft werden.
- Das oder die höherwertigen Oktette bleiben unverändert.
- Das oder die niederwertigen Oktette erhalten den Wert '**0**'.

Ein Beispiel: In welchem Netz liegt die Adresse 129.143.110.155, wenn die ersten 19 Bit die Adress-ID darstellen? In Kurzschreibweise (siehe Kap. 2.2) 129.143.110.155/19:

2.2 Adresspräfix und CIDR-Schreibweise

Das Einzige was die Netzmaske festlegt, ist die Stelle der Trennung zwischen Netz- und Host-ID innerhalb der IP-Adresse. Das kann man aber auch viel einfacher durch eine einfache Zahl, den sog. **Präfix**, angeben.

Der Präfix gibt an, aus wievielen Bits die Netz-ID besteht. Das ist gleichbedeutend mit der Aussage: Der Präfix gibt die Anzahl der Einsen in der Subnetzmaske an.

Der Präfix wird in der CIDR-Schreibweise [1] durch einen Schrägstrich getrennt hinter die IP-Adresse geschrieben. Z.B.:

129.143.14.155/28

[1] Classless Internet Domain Routing

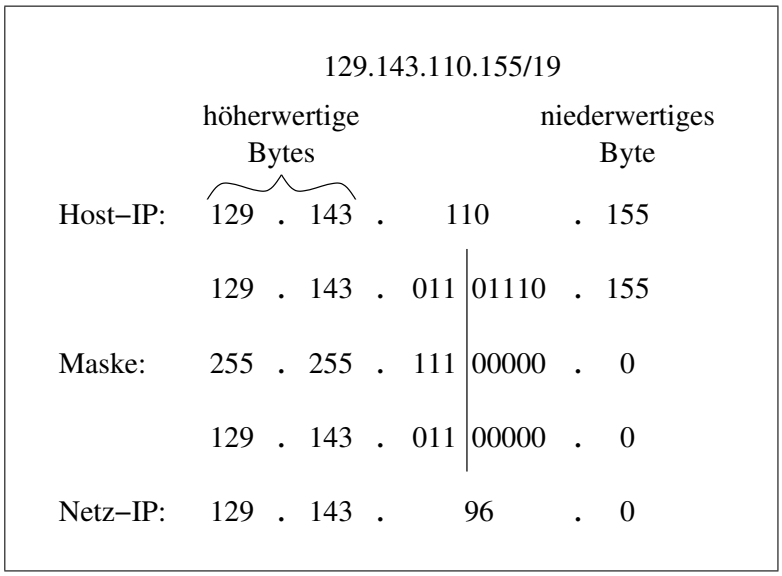

Abbildung 2: Vereinfachte Berechnung einer Netzadresse

Da die Länge einer IP Adresse konstant 32 bit ist, ist mit dem Präfix natürlich auch die Anzahl der Host-ID-Bits gegeben:

$$\text{Anzahl Bits der Host-ID} = 32 - \text{Präfix}$$

Daraus folgt für die Anzahl n der Host-Adresssen:

$$n = 2^{(32-\text{Präfix})}$$

2.3 Aufteilung eines Adressbereichs in Subnetze

Ein gegebener Adressbereich, z.b. 129.143.14.155/26 kann in kleinere Netze aufgeteilt werden. Dabei müssen folgende Regeln beachtet werden:

- Die Anzahl der Adressen eines Bereichs ist immer eine Zweierpotenz. Sie wird duch die Anzahl der Host-ID-Bits bestimmt (s.o.).
- Die Netzadresse ist die Adresse, bei der alle Host-ID-Bits **0** sind. Sie darf nicht als IP-Adresse für eine Arbeitsstation verwendet werden.
- Die Adresse, bei der alle Host-ID-Bits **1** sind, ist die Broadcast-Adresse des (Sub-)-Netzes. Diese darf ebenfalls nicht als Host-Adresse verwendet werden.
- Die Netzadresse muss ein Vielfaches der Anzahl der Adressen (einschliesslich Netz- und Broadcastadresse) im zugehörigen Netz sein.
- Es gehen pro Subnetz immer 2 Adressen für die Netzadresse und die Broadcastadresse verloren.

Soll der Bereich 129.143.14.128/26 in zwei Netze mit je 4 und 25 Arbeitsstationen

aufgeteilt werden, könnte eine mögliche Aufteilung so aussehen:

Netz1: Benötigt werden 4 Hostadressen. Da noch je eine Adresse für Broadcast und Netz dazukommen, muss ein Subnetz mit 8 Adressen verwendet werden. Dazu sind 3 Host-ID-Bits nötig, der Adresspräfix ist also $32 - 3 = 29$. Eine mögliche Netzadresse wäre z.B. **129.143.14.160/29**

Netz2: Für insgesamt $25 + 2 = 27$ Adressen wählen wir als Netzgrösse die nächste Zweierpotenz: 32. Benötigt werden also 5 Host-ID-Bits. Damit ergeben sich 2 mögliche Netzadressen: **129.143.14.128/27** oder **129.143.14.160/27** Das Netz1 kann dann an jeder Adresse im jeweiligen freien Bereich beginnen, die ein Vielfaches von 8 ist.

3 Die früher verwendeten IP-Netzklassen

Wie in der Einleitung bereits erwähnt, wurde das Internet früher in Netze aufgeteilt, bei denen die Netz-ID genau ein, zwei oder drei Oktette lang war. Diese Netze wurden entsprechend Klasse C, Klasse B und Klasse A Netze genannt.

Daneben gibt es noch zwei Netzklassen, die für spezielle Anwendungen gedacht sind: Klasse D ist für Multicast-Anwendungen reserviert. Hier gibt es keine Netz-ID sondern die letzten 28 Bit der Adresse werden als Multicast Adresse verwendet. Klasse E ist für zukünftige Anwendungen vorgesehen. Klasse D und E haben keine Netzmaske. Bei Klasse D Netzen stellen z.b. die 28 wählbaren Bits die Multicast-Adresse dar.

Die genaueren Adressbereiche sind:

Netzwerkklasse	**Adressbereich**		**Netzmaske**
öffentliche Netzadressen			
Klasse A	1.0.0.0	bis 127.0.0.0	255.0.0.0
Klasse B	128.0.0.0	bis 191.255.0.0	255.255.0.0
Klasse C	192.0.0.0	bis 223.255.255.0	255.255.255.0
Klasse D	224.0.0.0	bis 239.255.255.255	Multicast Adr.
Klasse E	240.0.0.0	bis 255.255.255.255	zukünft. Anw.
private Netzadressen			
Klasse A	10.0.0.0	bis 10.255.255.255	255.0.0.0
Klasse B	172.16.0.0	bis 172.31.0.0	255.255.0.0
Klasse C	192.168.0.0	bis 192.168.255.0	255.255.255.0

Tabelle 1: Die Adressbereiche der alten Netzwerkklassen

Schaut man sich die Adress-IDs etwas genauer an, kann man die Netze auch so klassifizieren:

4 Die beiden Schichtenmodelle

Nach dieser detaillierten Beschreibung der IP-Adressen, wird es jetzt Zeit, die restlichen Strukturen eines Netzwerkes der Reihe nach zu behandeln. Dazu ist es unverzichtbar, ein Modell zu verwenden. Dieses Modell soll helfen zu verstehen, wie die

Netzwerkklasse	Bits 31-0 mit Netz-ID und Host-ID			
Klasse A	**0nnn nnnn** . hhhh hhhh	. hhhh hhhh	. hhhh hhhh	
Klasse B	**10nn nnnn** . **nnnn nnnn**	. hhhh hhhh	. hhhh hhhh	
Klasse C	**110n nnnn** . **nnnn nnnn**	. **nnnn nnnn**	. hhhh hhhh	
Klasse D	**1110**mmmm . mmmm mmmm	. mmmm mmmm	. mmmm mmmm	
Klasse E	**11110**zzz . zzzz zzzz	. zzzz zzzz	. zzzz zzzz	

Tabelle 2: Netzwerkklassen in Binärdarstellung

einzelnen Teilaufgaben innerhalb eines Netzwerks verteilt sind und aufeinander aufbauen.

Abb. 3 zeigt die beiden Modelle nach DOD und OSI. Das DOD Modell wurde vor über 30 Jahren von der *Defense Advanced Research Projects Agency* (DARPA), einer Agentur des amerikanischen Verteidigungsministeriums (*Department of Defense*, **DOD**) vorgestellt.

Seine praktische Anwendung ist das heutige Internet.

Das *Open Systems Interconnection Basic Reference Model* (OSI-Model) wurde später entwickelt. Es dient der Veranschaulichung von Netzwerktechnologien und es gibt keine existierende Technologie, bei der das OSI-Modell genau 1:1 umgesetzt wurde. [2]

Für alle hardwareabhängigen Schichten ist das IEEE (*Institute of Electrical and Electronics Engineers* verantwortlich. Alle Schichten darüber werden von der IETF (*Internet Engineering Task Force*) definiert. Dazu werden sog. RFCs (*Request For Comments*) verwendet.

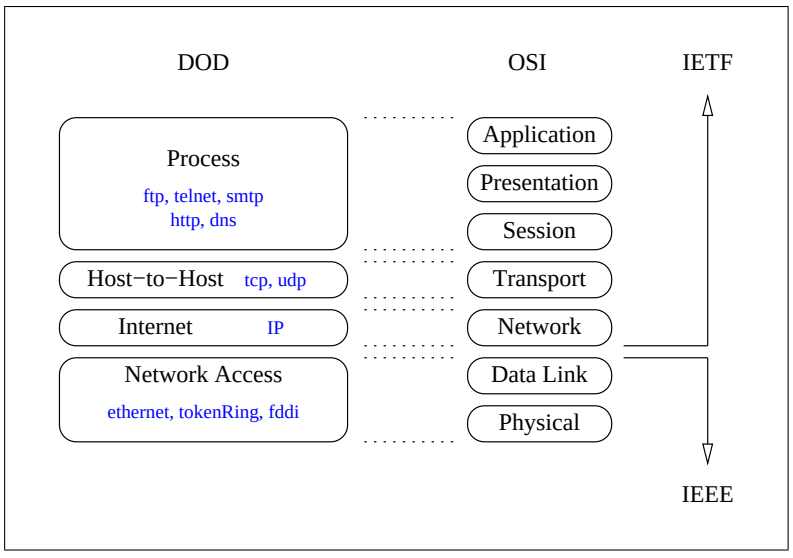

Abbildung 3: Gegenüberstellung von DOD- und Osi-Modell

[2] Am nächsten von allen realen Netzwerktechnologien kommt dem OSI-Modell -wie kann es anders sein- Apple Talk. allerdings ist Apple Talk inzwischen auch auf Macs durch Ethernet und TCP/IP abgelöst worden.

4.1 Encapsulation

Die nächste Abbildung (Abb. 4) zeigt am Beispiel einer Dateiübertragung mit FTP (File Transfer Protocol), wie die Anwendungsdaten zunächst in kleine Einheiten unterteilt und dann von einer Schicht zur nächsten nach unten weitergereicht werden.

Zunächst werden die Datenstückchen in ein TCP-Segment *eingepackt*.

Den TCP-Header kann man sich nun als *Beschriftung* des Päckchens vorstellen. In diesem Fall trägt die Beschriftung Informationen darüber, an welcher Stelle am Empfangsort der Inhalt des Päckchens wieder in den Datenstrom eingeordnet werden soll und welcher *Dienst*, identifiziert über die Portnummmer, der Empfänger ist. Im Beispiel der Abb. 4 wäre das Nummer 21, FTP.

Dieses Päckchen wird dann wiederum eingewickelt. Diesmal enthält die Beschriftung u.a. die IP-Adresse, also die Information, an welches *Gerät* (Host, Drucker, ...) das Paket geschickt werden soll.

Und zum Schluss nochmals eine Verpackung: das so entstandene Gebilde ist der *Ethernet-Frame*. Hier steht im Beschriftungstext endlich die Harware-Adresse des Empfängers, sowie eine Prüfsumme.

Dieses mehrfache Verpacken der Daten wird **Encapsulation** genannt.

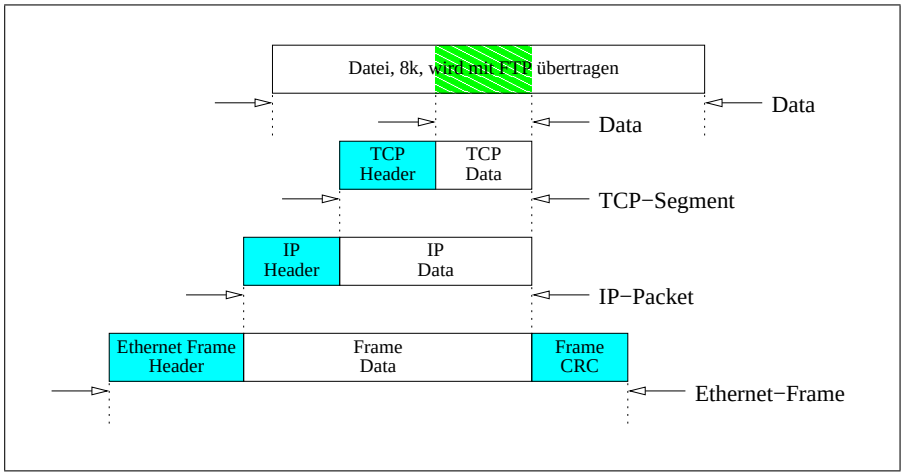

Abbildung 4: Die Ineinanderschachtelung (Encapsulation) der Daten

4.2 Die Protocol-Data-Units

Die Namen der Verpackungen und Unterverpackungen sind nicht zwingend normiert, können aber nach folgenden Regeln benannt werden. Im Englischen wird für (Protokoll)-Dateneinheit der Begriff *Protocol Data Unit*, **PDU** verwendet.

- Die Dateneinheit der obersten Schicht (**Application**) heist einfach: **Data**.
- Die Dateneinheit der **Transport**-Schicht (Host-To-Host) ist das TCP-**Segment** oder das UDP-**Datagramm**.
- In der **Internetworking**-Schicht heist die Dateneinheit **Packet**
- Und auf der **Network-Access**-Schicht spricht man von **Frames**

5 Ethernet II

Dieses Kapitel enthält einen kurzen Ausflug in die hardwareabhängigen Netzwerkschichten. Da die mit Abstand wichtigste Netzwerktechnologie bei lokalen Netzwerken Ethernet ist, wollen wir uns auch nur auf diese beschränken.

Ethernet wurde von Intel, DEC und Xerox erfunden und ist ein Bussystem, das heisst, dass alle Teilnehmer parallel geschaltet sind und durch eine [3] Leitung verbunden sind.

Die Daten über diese Leitung werden in Einheiten gesendet. Diese Einheiten nennt man (wie im vorigen Kapitel schon eingeführt): **Frames**.

Die Abb. 5 zeigt einen Ethernet-II-Frame. Es gibt noch 3 weitere Ethernet-Frames, die sich um ein einige zusätzliche Kontroll-Oktette vom Ethernet-II-Frame unterscheiden.

Wichtigstes gemeinsames Merkmal der Ethernet-Frames sind aber die Quell- und die Zieladresse. Beide sind je 6 Oktette (48bit) lang und werden als *Hardwareadresse*, meistens jedoch als *MAC-Adresse* bezeichnet.

MAC steht hier für Medium Access Control. Das ist eine Schicht im Ethernet Datenmodell, die direkt mit der darunterliegenden Hardware zusammenarbeitet und die u.A. die Frames zusammenbaut und hierzu die Prüfsumme berechnet und auswertet.

5.1 Zugriffsteuerung CSMA/CD

CSMA/CD steht für *Carrier Sense Multiple Access / Collision Detection*:

Bei Ethernet als Bussystem sind alle Stationen parallel an einer Leitung angeschlossen. Dabei gibt es keinen Busmaster, d.h. alle Stationen sind gleichberechtigt: **Multiple Access**

Das Verfahren CSMA/CD sorgt nun dafür, dass immer nur eine Station Daten senden kann. CSMA/CD kommt dabei ohne zusätzliche Kommunikation zwischen den Stationen aus.

Das Prinzip: eine Station, die senden möchte, hört zunächst den Bus ab um festzustellen, ob dieser frei ist. Dazu wird einfach gemessen, ob ein Trägersignal auf dem Bus anliegt: **Carrier Sense** [4] Nun kann der Fall eintreten, dass zwei Stationen fast zeitgleich beginnen, ein Ethernet-Paket zu senden. Da sich die Signale auf dem Bus nur mit ca. $1.8 \cdot 10^8 m/s$ ausbreiten, werden sie sich nach einer bestimmten Laufzeit vermischen, wobei die Daten natürlich komplett verfälscht und damit unbrauchbar werden.

[3] Ursprünglich war es eine Koaxleitung, heute sind es meistens 2 Twisted-Pair-Leitungen, eine für Sende- und eine für Empfangsbetrieb.

[4] Dies war beim ursprünglichen Ethernet technisch einfach zu lösen, da zusammen mit den Daten immer der Bittakt von 10MHz mitgesendet wurde. Erreicht hat man dies durch Verwendung des **Manchester-Codes**, bei dem immer zur Bitmitte ein Pegelwechsel auf der Leitung auftritt.

So etwas wird als *Kollision* bezeichnet. Tritt so eine Kollision auf, wird sie von den sendenden Stationen erkannt: **Collision Detection**.

Erkennt eine Station eine Kollision, hört sie sofort mit der Sendung auf. Damit dann, wenn der Bus wieder frei ist nicht wieder die beiden Sender gleichzeitig beginnen, hat jeder Teilnehmer einen Zufallsgenerator, der eine zufällige Wartezeit bis zur nächsten Sendung erzeugt.

5.1.1 Kollisionsdomänen und Slot-Time

Damit Kollisionen von den Sendern sicher erkannt werden, muss der durch die Kollision entstehende Datenmüll bei jedem Sender eintreffen, **solange dieser noch sendet**!

Der Grund ist einfach: der Sender erkennt eine Kollision durch Vergleich der Daten, die er sendet mit denen, die auf der Leitung sind. Bei einer Kollision sind diese beiden sicher nicht identisch.

Leider breitet sich der Datenmüll (=Kollision) auf der Leitung aber nicht unendlich schnell aus.

Ein Ethernet-Paket muss minimal 64 Bytes = 512 bit lang sein. In diesem Beispiel soll es mit 100 MBit/s gesendet werden. Dann dauert die Aussendung des gesamten Pakets $t = 5.12\mu s$. Mit der Ausbreitungsgeschwindigkeit von $v = 1.8 \cdot 10^8 m/s$ ist das erste vom letzten Bit

$$s = t \cdot v = 921m$$

entfernt.

Im schlimmsten Fall (worst case) tritt die Kollision dicht bei einer weit entfernten Staion auf. Die Zeit, die ein Signal braucht bis es dorthin und als Kollisionssignal wieder zurück läuft, nennt man *Round Trip Delay*. Dieser Round Trip Delay muss also immer kleiner sein als die Zeit $t = 5.12\mu s$, damit eine Kollision auch im schlechtesten Fall erkannt wird.

Oder anders ausgedrückt: die Stationen dürfen nicht weiter als $921m/2 = 460m$ voneinander entfernt ein.

In der Praxis hat man noch ca. den Faktor 5 als Sicherheit eingebaut: bei 100BaseT ist die **maximale Länge eines Segments 100 m**. Unter Segment versteht man hier den Teil eines Netzes, in dem Kollisionen erkannt werden müssen. Man nennt diesen Teil auch *Kollisionsdomäne*.

Z.B. bilden alle Ports eines Hubs eine Kollisionsdomäne. Bei einem Switch ist das anders: dieser trennt Kollisionsdomänen voneinander indem er die Pakete analysiert. Dazu später mehr.

6 IP-Pakete im Ethernet-Frame

Im folgenden Kapitel wird dargestellt, wie ein IP-Paket mit Hilfe von Ethernet als Transportmedium versandt wird. Dazu betrachten wir zunächst den Kopfteil eines IP-Paketes (Abb. 6).

Abb. 6 zeigt den Aufbau eines IP-Kopfs (Header).

Die einzelnen Teile haben folgende Bedeutung:

Version IP-Version.

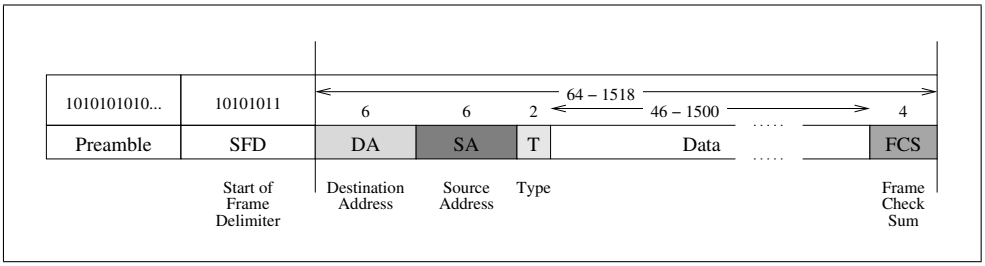

Abbildung 5: Ein Ethernet-II-Frame

Version 4	HLEN 4	Type of Service 8	Total Length 16	
Identification 16			Flags 3	Fragment Offset 13
Time to Live 8		Protocol 8	Header Checksum 16	
Source Address 32				
Destination Address 32				
Options variabel				Padding n * 8

Abbildung 6: Kopfteil eines IP-Pakets

HLEN Länge des Headers.

Type of Service

Total Length Gesamtlänge des Datagramms. Bis zu 64 kBytes.

Identification zu welchem IP-Datagramm gehört das Fragment?

Flags sind bereits alle Fragmente eines Datagramms empfangen?

Fragment Offset wo gehört das Fragment hin? Erklärung siehe unten.

Time to Live Nach wieviel Router-Hops soll das Paket fallengelassen werden?

Protocol ICMP / TCP / UDP und ein paar weitere.

Header Checksum 16-bit Paritäts-Summe. Wird nur über den Header gebildet!

Source Address 32-bit IP Adresse des Absenders.

Destination Address 32-bit IP Adresse des Ziels.

Options Für Erweiterungen.

Padding Füllbits auf ganze 32 bit - Worte

Abb. 7 zeigt, wie die IP-Pakete im Datenteil eines Ethernet-Rahmens eingebettet werden.

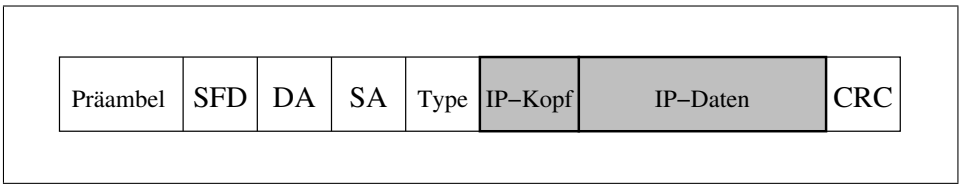

Abbildung 7: Ein IP-Paket im Datenteil eines Ethernet-Rahmens

Da ein IP-Datagramm bis zu 64 kBytes lang sein kann, das Datenfeld eines Ethernet-Frames aber maximal 1500 Bytes lang ist, muss das IP-Datagramm unter Umständen auf mehrere Ethernet-Rahmen verteilt werden. Damit am Empfangsort alles wieder in der richtigen Reihenfolge zusammengesetzt werden kann, enthält jeder IP-Kopf eine Schlüsselnummer die bestimmt, welche Fragmente zusammengehören und einen Frament Offset der angibt, an welche Stelle im IP-Datenfeld dieses Fragment hingehört.

Geht ein Fragment unterwegs verloren, muss das gesamte Datagramm wiederholt werden. Datagramme, die kürzer als 576 Bytes sind, sollen nur unfragmentiert übertragen werden.

6.1 Adressauflösung mit dem Address Resolution Protocol ARP

Da das Internet Protocol hardwareunabhängig ist, muss es mit jeder Netzwerktechnologie funktionieren. Da als Netzwerk meistens Ethernet verwendet wird, wird die Adressauflösung an diesem Beispiel besprochen.

Ethernet benutzt eine 6 Oktette Adresse, um Datenpakete zu adressieren. Im folgenden Beispiel möchte ein Rechner in einem IP-Netz einer Station ein IP-Paket senden. Das IP-Paket wird dazu in das Datenfeld eines Ethernet Rahmens eingestellt. Die Quell- und Ziel IP-Adresse steht im IP-Kopf. Sie kann aber nicht für die Adressierung des Pakets verwendet werden, da ein Ethernet-Paket eine andere, eben die MAC-Adresse hat. Siehe dazu die Abbildungen 7 und 6.

Um das IP-Paket im Ethernetrahmen nun richtig adressieren zu können, muss der Sender wissen, welche MAC-Adresse die Station mit der IP-Zieladresse hat.

Um diese Zuordung aufzulösen, wird das Adress Resolution Protocol (**ARP**)verwendet:

Die Sendestation macht das gleiche wie eine Person, die in einer Gruppe von 150 Leuten jemand bestimmtes finden will, von dem nur der Name (IP-Adr) bekannt ist: sie ruft laut dessen Namen so dass *alle* es hören können. Der Angesprochene wird sich dann melden.

Auf das Ethernet übertragen, entspricht das laute Rufen einer Broadcast-Sendung: ein Datagramm wird an **alle** Mitglieder des Netzes gesendet. Damit das Datagramm an alle Stationen gesendet wird,wird als physikalische Zieladresse die Broadcastadresse FF:FF:FF:FF:FF:FF verwendet.

Das Datagramm enthält eine Abfrage nach der Ziel-IP-Adresse. Der Rechner mit der zutreffenden IP-Adresse sendet ein Antwort-Datagramm aus dem der Empfänger wiederum die MAC-Adresse extrahieren kann.

Zusammengefasst und mit Beispieladressen sieht der Vorgang so aus (Tabelle 3):

ARP-Request: wer hat Adresse 172.16.231.64?	
Sender MAC-Adresse:	00:C0:3D:00:2A:D9
Sender IP-Adresse:	172.16.231.12
MAC-Adresse des Ziels:	FF:FF:FF:FF:FF:FF
IP-Adresse des Ziels:	172.16.231.64

ARP-Response	
Sender MAC-Adresse:	00:C0:3D:00:4A:05
Sender IP-Adresse:	172.161.231.64
MAC-Adresse des Ziels:	00:C0:3D:00:2A:D9
IP-Adresse des Ziels:	172.16.231.12

Tabelle 3: Sender- und Zieladressen bei ARP

Anschliessend kann das ursprüngliche Datagramm zugestellt werden. Damit die Adressauflösung nicht bei jedem Sendevorgang wiederholt werden muss, werden die IP-MAC-Adresspaare in einem sog. ARP-Cache gespeichert.

Die Speicherung ist aber nicht dauerhaft, sondern wird von Zeit zu Zeit gelöscht. Würde man die Zurdnungen dauerhaft speichern, hätte man spätestes dann ein Problem, wenn bei einem Rechner die Netzwerkkarte ausgetauscht werden muss.

6.2 Reverse ARP beim Booten vom Netz

Möchte man das Betriebssystem eines Rechners nach dem Einschalten über das Netzwerk laden (Remote-Boot), kommt das Reverse Address Resolution Protocol (RARP) zum Einsatz. Im Verlauf des Startvorgangs bekommt der Rechner zwar seine IP-Adresse von einem Server zugewiesen, direkt nach dem Einschalten "kennt" der Rechner zunächst aber nur die physikalische (MAC) Adresse seiner Netzwerkschnittstelle. Er sendet znächst einen RARP-Request "hier ist meine MAC-Adresse, wer hat eine IP-Adresse für mich?" und erhält daraufhin eine IP-Adresse zugewiesen.

Hier der Vorgang wiederum mit Beispieladressen (Tabelle 4) Dabei ist vorausgesetzt, dass auf dem Rechner mit der IP-Adresse 172.16.231.64 und der MAC-Adresse 00:C0:3D:00:4A:05 ein **RARP**-Dienst läuft, der die Anfrage bearbeitet:

Die beim Request verwendete IP-Adresse 0.0.0.0 hat einfach die Bedeutung: "eine beliebige IP-Adresse". Diese spezielle Adresse kommt häufig in Routingtabellen vor und wird dort als *Default Route* bezeichnet.

7 Routing in IP-Netzen

Um zwei unterschiedliche Subnetze verbinden zu können, benötigt man Rechner oder spezielle Geräte, die von jedem dieser Subnetze ereichbar sein müssen. D.h. diese Spezialrechner haben eine IP-Adresse aus jedem Netz, dessen Datagramme sie weiterleiten müssen.

RARP-Request: was ist meine IP-Adresse?	
Sender MAC-Adresse:	00:C0:3D:00:2A:D9
Sender IP-Adresse:	0.0.0.0
MAC-Adresse des Ziels:	FF:FF:FF:FF:FF:FF
IP-Adresse des Ziels:	255.255.255.255

RARP-Response	
Sender MAC-Adresse:	00:C0:3D:00:4A:05
Sender IP-Adresse:	172.161.231.64
MAC-Adresse des Ziels:	00:C0:3D:00:2A:D9
IP-Adresse des Ziels:	172.16.231.12

Tabelle 4: Sender- und Zieladressen bei RARP

Die genaue OSI-Bezeichnung derartiger Rechner ist *Router* (das wird französisch ausgesprochen!). In Unix-Umgebungen nennt man die Router oft auch *Gateways*, obwohl Gateway strenggenommen zwei Netze unterschiedlicher Protokolle verbinden.

Wer einen Router benötigt, muss dazu nicht ein spezielles Gerät kaufen. Die Kernel von Linux und NT sind beide dafür ausgelegt, auch Routingaufgaben auszuführen.

Damit der Kernel weiss, wohin welches Paket geroutet werden soll, muss man ihm die Routen in die sog. *Routingtabelle* eintragen.

Eine Route ist die Kombination aus Zielnetz und Gateway. Das Zielnetz wird dabei wiederum durch die Netzadresse und die zugehörige Netzmaske oder den Präfix definiert.

Hier ein Beispiel für drei IP-Netze, die über zwei Gateways verbunden sind (Abb. 8).

7.1 Wie findet ein Datagramm sein Ziel?

Ein IP-Datagramm soll versandt werden. Der Host-Rechner schickt es zunächst an das Gateway seines Netzes. Der Gateway-Rechner (Router) bearbeitet das Paket anschliessend nach folgendem Ablaufschema:

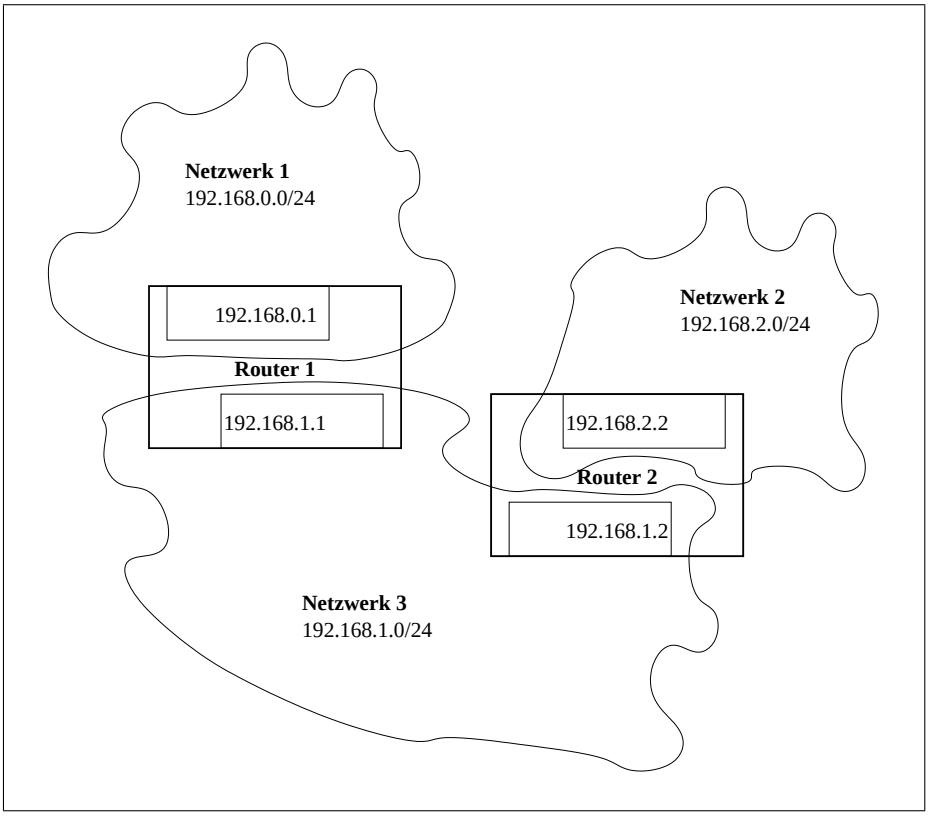

Abbildung 8: Drei IP-Netze mit Routern

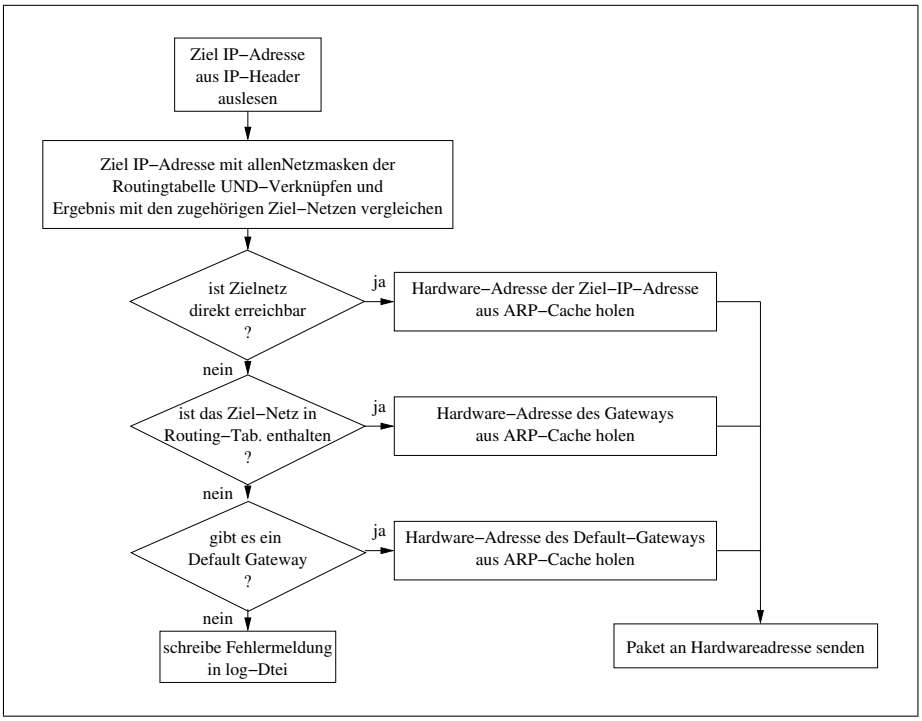

Abbildung 9: Entscheidungsablauf innerhalb eines Routers

8 Die Protokolle UDP und TCP

Die beiden Protokolle User Datagramm Protocol (UDP) und Transmit Control Protocol (TCP) arbeiten eng mit der darunterliegenden Schicht 3 (IP) zusammen. Die Zusammenarbeit ist so eng, dass man TCP und IP oft als eine Einheit, eben TCP/IP bezeichnet.

Wenn man nochmals zur Abb. 6 zurückblättert, findet man dort das 8 Bit lange Feld **Protocol**. Dieses Feld enthält eine Schlüsselnummer für das vom IP-Datenteil transportierte Protokoll. Hier die drei wichtigsten Schlüsselnummern:

Schlüssel-Nr.	Protokoll
1	ICMP
6	TCP
17	UDP

8.1 Portadressen

Beide Protokolle (TCP und UDP) arbeiten mit *Portnummern* oder *Portadressen*. Eine Portadresse kann man sich wie eine *Durchwahlnummer* einer Telefonanlage vorstellen. Dabei entspricht die IP-Adresse der Nummer des Hauptanschlusses.

Diese Analogie geht sogar noch weiter: der Sachbearbeiter, der über die Durchwahl zu erreichen ist, entspricht dann einem der Dienste der Schicht 7 wie z.B. ssh, ftp, http, pop3, imap, ...

Die Portadressen haben 16 bit, es gibt somit 65536 Ports.

Damit man die Standarddienste immer unter den selben Portadressen erreichen kann, sind die ersten 1024 Portadressen fest zugeordnet. Wer die Zuordnung genauer kennen lernern möchte, muss sich nur die Datei /etc/services auflisten lassen. Hier ein kleiner Auszug:

```
#
# Network services, Internet style
#
# Note that it is presently the policy of IANA to assign a single well-known
# port number for both TCP and UDP; hence, most entries here have two entries
# even if the protocol doesn't support UDP operations.
#
# The latest IANA port assignments can be gotten from
#
#       http://www.iana.org/assignments/port-numbers
#
# The Well Known Ports are those from 0 through 1023.
# The Registered Ports are those from 1024 through 49151
# The Dynamic and/or Private Ports are those from 49152 through 65535
#
# $FreeBSD: src/etc/services,v 1.89 2002/12/17 23:59:10 eric Exp $
#       From: @(#)services      5.8 (Berkeley) 5/9/91
#
# WELL KNOWN PORT NUMBERS
#

.......          ......     ......................

chargen          19/udp      # Character Generator
chargen          19/tcp      # Character Generator
ftp-data         20/udp      # File Transfer [Default Data]
ftp-data         20/tcp      # File Transfer [Default Data]
ftp              21/udp      # File Transfer [Control]
ftp              21/tcp      # File Transfer [Control]
#                            Jon Postel <postel@isi.edu>
ssh              22/udp      # SSH Remote Login Protocol
ssh              22/tcp      # SSH Remote Login Protocol
#                            Tatu Ylonen <ylo@cs.hut.fi>
telnet           23/udp      # Telnet
telnet           23/tcp      # Telnet
#                            Jon Postel <postel@isi.edu>
                 24/udp      # any private mail system
                 24/tcp      # any private mail system
#                            Rick Adams <rick@UUNET.UU.NET>
smtp             25/udp      # Simple Mail Transfer
smtp             25/tcp      # Simple Mail Transfer
#                            Jon Postel <postel@isi.edu>

.......          ......     ......................
```

Wenn man den einleitenden Kommentar der Datei /etc/services liest, erfährt man, dass es noch eine weitere Unterteilung der Ports gibt: von 1024 bis 49151 liegen die "registered ports" und ab 49152 liegen die dynamischen Ports. Diese dynamischen Ports werden vom Betriebssystem verwendet, um bei einer TCP-Verbindungsanfrage den Rückkanal aufzubauen.

8.2 Die Header von TCP- und UDP-Paketen

Die Abbildungen 10 und 11 zeigen die Header von UDP-Paketen bzw. TCP-Paketen.

8.3 TCP - Verbindungen

Die Hauptaufgaben des TCP-Protokolls sind:

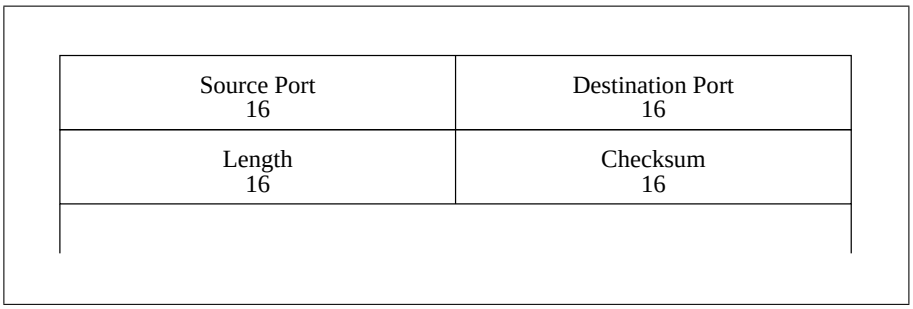

Abbildung 10: UDP-Header

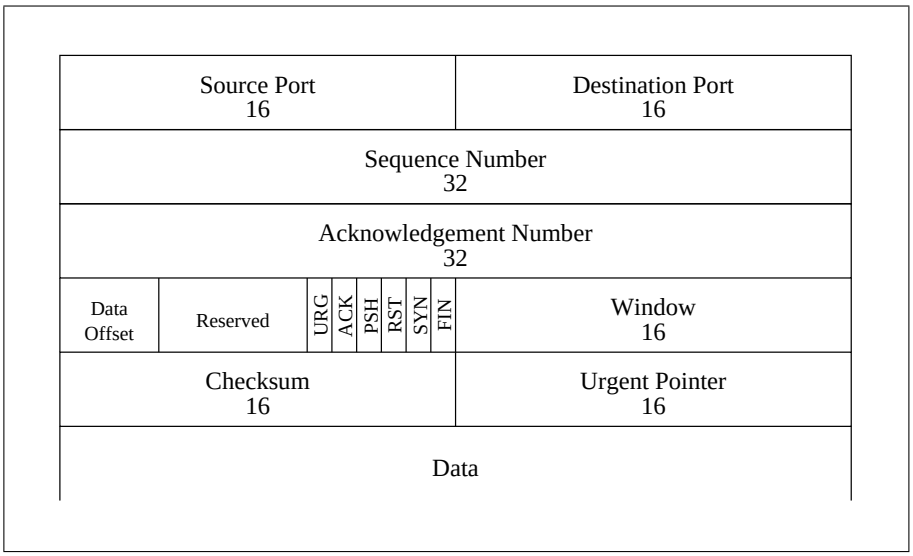

Abbildung 11: TCP-Header

- Verbergen der darunterliegenden *paketorientierten* Schichten (IP und Ethernet)
- TCP stellt anstelle einzelner Datenpakete (Datagramme) einen *Byte-Strom* zur Verfügung
- Dazu stellt TCP zwischen 2 Rechnern eine logische *Verbindung* her, die man sich wie eine fest verlegte Datenleitung zwischen zwei Endpunkten vorstellen kann. Die Endpunkte werden dabei auf jeder Seite von der IP-Adresse des Hosts **und** einer Portnummer gebildet. Eine solche Paarung (IP-Adr. und Port) wird auch als *Socket* bezeichnet.
- Eine TCP-Verbindung wird durch ihre beiden Endpunkte bestimmt. Das heisst, dass von einem bestimmten Port eines Servers gleichzeitig *mehrere TCP-Verbindungen* zu verschiedenen Client-Ports geöffnet sein können.

8.4 Der TCP-Bytestrom

Das Lesen- und Schreiben der Bytes einer TCP-Verbindung hat sehr grosse Ähnlichkeit mit dem Lesen- und Schreiben einer Datei. Dies wurde von den TCP Entwicklern [5] so gewollt, da das den Zugriff auf TCP-Verbindungen sehr vereinfacht: es können von Betriebssystemseite her die selben Funktionen wie für Dateizugriffe verwendet werden.

TCP muss also einen numerierten Bytestrom zur Verfügung stellen. Damit man nun nicht für jedes einzelne Byte ein ganzes Paket mit 20 Byte langem Header senden muss, werden die Bytes zu Blöcken, genannt Segmente zusammengefasst.

Der TCP-Bytestrom wird in Segmente eingeteilt.

Dabei gilt:

Die maximale Segmentgrösse (Maximum Sequence Size, MSS) ist 1500 Bytes - 20Bytes IP-Header - 20 Bytes TCP-Header = 1460 Bytes.

Nun wird jedes Segment nummeriert.

Die Segmentnummer oder **Sequence Number** ist die Nummer, die das erste Byte im Segment hat.

Wir erinnern uns: alle Bytes des Byte-Stroms werden durchnummeriert. Zur Verdeutlichung dieses Zusammenhangs siehe Abb. 12

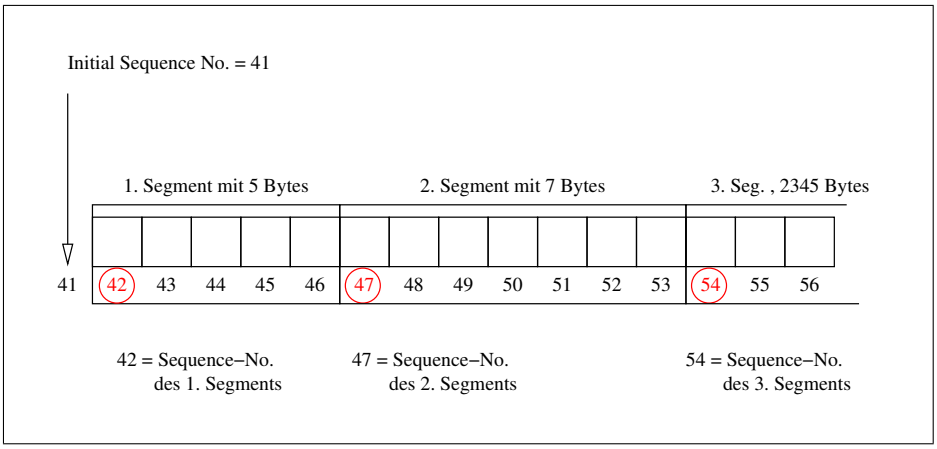

Abbildung 12: Sequence Number der TCP-Segmente

Bei der Nummerierung der Segmente wird **nicht** einfach bei 0 begonnen. Das würde es Angreifern sehr erleichtern, falsche Pakete in die TCP-Verbindung einzuschleusen. Um dies zu erschweren, beginnt die Nummerierung mit einer zufällig erzeugten 32-bit-Zahl, der sog. **Initial Sequence Number**. Im Beispiel sei dies die Zahl 41.

Im TCP-Header sind nun aber zwei je 32bit lange Felder enthalten:

Sequence-Number: dieses Feld wurde oben erklärt

Acknowledgement-Number: mit dieser Nummer bestätigt die Empfangsstation die

[5] Robert E. Kahn und Vinton G. Cerf, 1973-1981 (!)

korrekt empfangenen Bytes. Dabei wird als Acknowledgement-Number die Nummer des Bytes, das als **nächstes erwartet** wird verwendet. Die Gültigkeit der Acknowledgement-Number wird durch Setzen des **ACK**-Bits gekennzeichnet.

Für die Bestätigung mit der Acknowledgement-Number kann der Empfänger ein eigenes, leeres Paket (ohne Daten) erzeugen, er kann aber auch ein Datenpaket zur Gegenstelle verwenden und dort die Acknowledgement-Number "Huckepack" (engl. *piggyback*) mitgeben.

Ein weiteres, wichtiges Headerfeld ist:

Window: Dieses Feld gibt an, wieviele Bytes gesendet werden dürfen, bevor eine Acknowledgement-Number als Bestätigung eintreffen **muss**. Mit dieser Bestätigung sind dann auch alle vorhergehenden Bytes bestätigt. Diese Fenstergrösse ist variabel.

8.5 Verbindungsauf- und -abbau

Damit der Austausch der Sequence- und Ack-Numbers beim Start einer Verbindung synchronisiert werden kann, gibt es das **SYN**-Bit. Beendet wird die TCP-Verbindung durch Setzen des **FIN**-Bits.

Wie das im Detail abläuft lässt sich am besten durch Studieren einer aufgezeichneten Verbindung (vgl. 9.2), die in Abb. 13 nochmals schematisch dargestellt ist, verstehen.

Die Aufzeichnung wurde mit dem Kommando `tcpdump` erzeugt, allerdings wurde dessen Ausgabe etwas lesbarer formatiert und vor allem stark gekürzt.

Eine Anmerkung noch zum **PSH**-Bit (Push): Damit wird erreicht, dass die am Empfänger eintreffenden Bytes nicht gepuffert, sondern sofort zur Anwendung geschickt werden. Das ist bei diesem Beispiel einer SSH-Sitzung zwingend. Ohne PSH-Bit würden die Remote-Shell sehr träge werden: alle Tastatureingaben würden gepuffert und erst wenn der Puffer voll ist, würden die Zeichen angezeigt werden.

9 Aufzeichnen der Pakete einer Netzwerkschnittstelle

Bei der Fehlersuche oder um ein besseres Verständnis der Netzwerkprotokolle zu gewinnen [6] , kann man den Datenstrom an einer Netzwerkschnittstelle im sog. *promiscous mode* mithören und ggfs. aufzeichnen und analysieren. Der *promiscous mode* wird in Abschnitt 9.1 erklärt.

Hat man die nötigen Rechte, kann man mit den beliebten Programmen *ethereal* oder *tcpdump* den Netzwerkverkehr abhören und aufzeichnen. Die empfangenen Pakete können in einer Datei im pcap-Format gespeichert werden und stehen dann auch später zur Anaylse zur Verfügung.

pcap steht für *package capture library*. Eine Windows-Version gibt es auch: **WinPcap**.

Hier ein Beispiel mit tcpdump. Ausgegeben werden die TCP-Pakete der zuvor aufgezeichneten Datei `etherDump.pcap`:

```
tcpdump -nt -r etherDump.pcap  ip host not 224.0.0.251
```

[6] natürlich gibt es auch noch weniger ehrbare Motive, wie z.B. das Ausspähen von Passwörtern von Benutzern, die noch keine sicheren Protokolle verwenden. `telnet` und `ftp` z.B. sind heutzutage absolut verboten!

Die Angabe **ip host not 224.0.0.251** filtert unerwünschte Pakete aus, die Schalter **-nt** unterdrücken die Ausgabe des Aufzeichnungszeitpunkts (t) und verwenden Portnummern (n) anstelle der Dienstbezeichnungen (in diesem Fall **ssh**).

9.1 Freizügige Netzwerkschnittstellen

promiscous heist soviel wie "freizügig" (wahllos Unzucht treibend).

Im *promisous mode* empfängt eine Netzwerkschnittstelle **alle** Ethernet-Pakete, also auch diejenigen, deren Ziel-MAC-Adresse gar nicht der eigenen MAC-Adresse entsprechen. Alle empfangenen Pakete werden an die höheren Schichten übergeben.

9.2 Aufgezeichnetes TCP

```
IP 192.168.1.102.52783 > 192.168.1.101.22:
   S 704093757:704093757(0) win 65535

IP 192.168.1.101.22 > 192.168.1.102.52783:
   S 1759015107:1759015107(0) ack 704093758 win 65535

IP 192.168.1.102.52783 > 192.168.1.101.22:
    . ack 1759015108 win 65535


IP 192.168.1.101.22 > 192.168.1.102.52783:
   P 1759015108:1759015147(39) ack 704093758 win 65535

IP 192.168.1.102.52783 > 192.168.1.101.22:
    P 704093758:704093796(38) ack 1759015147 win 65535

IP 192.168.1.101.22 > 192.168.1.102.52783:
    . ack 704093796 win 65535


.....................................................
.....................................................
.....................................................


IP 192.168.1.102.52783 > 192.168.1.101.22:
    P 704096204:704096236(32) ack 1759024035 win 65535


IP 192.168.1.102.52783 > 192.168.1.101.22:
    F 704096236:704096236(0) ack 1759024035 win 65535

IP 192.168.1.101.22 > 192.168.1.102.52783:
    . ack 704096237 win 65535

IP 192.168.1.101.22 > 192.168.1.102.52783:
    F 1759024035:1759024035(0) ack 704096237 win 65535

IP 192.168.1.102.52783 > 192.168.1.101.22:
    . ack 1759024036 win 65534
```

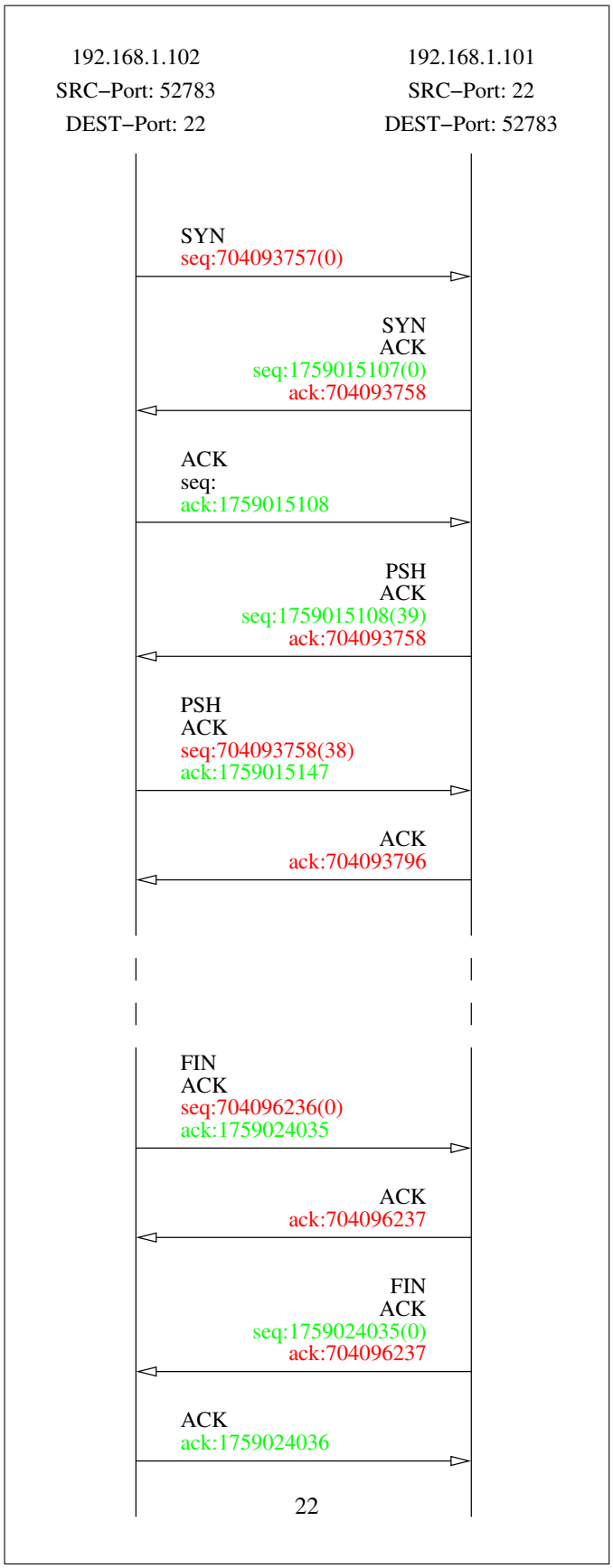

Abbildung 13: Schema einer kompletten SSH-Sitzung über TCP