Grundlagen von Internet-Protokollen

Martin Gaitzsch
Torsten Sorger
Seminar "18.415 Sicherheit in vernetzten Systemen"
Dr. Hans-Joachim Mück
Fachbereich Informatik
Universität Hamburg

2. Mai 2003

Inhaltsverzeichnis

1 Einleitung

Die Kommunikation im Internet muss nach bestimmten festgelegten Regeln ablaufen, wenn man eine zuverlässige Kommunikation wünscht. Im Internet wird dazu das paketorientierte Protokoll IP benutzt. Im Gegensatz zu einer leitungsvermittelten Kommunikation werden die Daten in einzelnen Paketen versandt, die keinen vorher festgelegten Weg durch das Kommunikationsnetz nehmen. Jedes Paket/Datagramm enthält Informationen zum Absender und Empfänger sowie Steuerinformationen und die Nutzdaten. Das Netz ist für die korrekte Weiterleitung der Pakete zuständig. Jeder Vermittlungsrechner muss dazu entscheiden, welcher Weg ein Paket am schnellsten oder sichersten näher an sein Ziel bringt. Die Wegewahl wird Routing genannt und es existieren ausgefeilte Algorithmen dafür. Falls ein Paket fehlerhaft oder gar nicht seinen Empfänger erreicht, gibt es Routinen zur Fehlerbehebung und Flusskontrolle.
Schwachpunkte von TCP/IP sind unter anderem, dass sämtliche Kommunikation im Klartext geführt wird und z.B. Absendeadressen im Prinzip beliebig fälschbar sind. Im Moment ist überwiegend die Version 4 des IP-Protokolls im Einsatz. Die seit einigen Jahren standardisierte Version 6 beseitigt viele Schwächen der Version 4 wie beispielsweise fehlende Verschlüsselung. IPv6 ist heute erst selten im Einsatz und hat noch keine große Bedeutung.
In dieser Arbeit geben wir im wesentlichen einen Überblick über verschiedenen Protokolle der TCP/IPv4 Protokollfamilie. Alle Grafiken beziehen sich ausschließlich auf Version 4 des Internet Protokolls.

2 ISO/OSI Schichten und TCP/IP-Äquivalente

Im Jahre 1984 wurde das "Reference Model for Open Systems Interconnection" oder kurz OSI-Modell veröffentlicht. Auf einem 7-Schichten-Kommunikationsmodell wurden einzelne Protokoll Layer und ihre spezifischen Aufgaben und Funktionen definiert. Dabei wurden jeder der sieben Funktionsschichten genau formulierte Aufgaben zugewiesen, die für eine erfolgreiche Kommunikation zu erfüllen sind. Die Festschreibung konkreter Netzwerksoftware erfolgt allerdings nicht. Die zu diesem Zeitpunkt schon durch einige Softwareprodukte umgesetzte TCP/IP-Architektur hatte sich hingegen bereits etabliert.
Der Grundgedanke des Schichtenmodells entwickelte sich aus der Überlegung, dass die zur Kommunikation erforderlichen Prozesse nur sehr schwer in einem einheitlichen Protokoll untergebracht werden können. Es müssen demnach mehrere funktional unterschiedliche aber abhängige Protokolle gebildet werden, die im Zusammenspiel untereinander schnelle und vor allem sichere Datenverbindungen ermöglichen sollen. Dabei erhält eine Schicht n

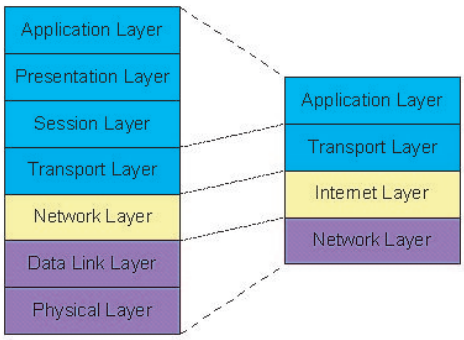

Abbildung 1: OSI Schichtenmodell und TCP/IP Äquivalent

kontinuierlich Steuerinformationen von der ihr unterliegenden Schicht $n-1$, die sie für die Bewältigung der Aufgaben verarbeiten muss. Sie wird dann allerdings auch ihrerseits Informationen an die darüberliegende Schicht $n+1$ beziehungsweise die darunterliegende Schicht $n-1$ weitergeben, um einen durchgängigen Informationsfluss über alle Kommunikationsschichten hinweg zu gewähren. Bei einer Rechner-zu-Rechner Kommunikation werden Informationen aus der Anwendungsschicht des Rechners A durch die einzelnen Schichten des Referenzmodells bis zur physikalischen Schicht 1 durchgereicht. Dabei fügt jede Schicht ihre Steuerinformationen vor der Weitergabe hinzu. Nach der eigentlichen Übertragung auf dem physikalischen Medium erfolgt die Übernahme bei Rechner B an der untersten Schicht. Der Informationsfluss verläuft nun vertikal in umgekehrter Reihenfolge, bis die Daten in der Anwendungsschicht des Rechners B angekommen sind. Die einzelnen Schichten entfernen vor Weitergabe an die nächsthöhere Schicht ihre Steuerinformationen, so dass letztlich die Zieldaten in Rechner B den Ursprungsdaten in Rechner A genau entsprechen.
Die linke Seite von Abbildung 1 zeigt eine schematische Übersicht über die einzelnen Schichten, die im folgenden erläutert werden.

2.1 Layer 1: Physikalische Schicht

Die Bitübertragungsschicht ist als unterste Schicht für die Herstellung einer physikalischen Verbindung zwischen zwei Kommunikationsendpunkten verantwortlich. Die zu Verfügung stehenden Hilfsmittel wie Kabel, Controller oder Modulationsverfahren müssen von ihr kontrolliert werden. Treten hier Probleme der Hardware auf, so werden diese Störungen an die nächsthöhere, die Datensicherungsschicht, weitergegeben. Diese Weitergabe erfolgt jedoch ungesichert. In der Regel wird der physikalische Layer unter anderem durch die Treibersoftware für die Netzwerkhardware repräsentiert.

2.2 Layer 2: Datensicherung Schicht

In dieser Schicht werden vor allem Sicherungsmechanismen implementiert, die bei fehlerhaften Übertragung für eine entsprechende Wiederherstellung sorgen. Es werden physikalische Verbindungen (Links) aufgebaut, über die zu Blöcken (Frames) zusammengefasste Bitströme transportiert werden. Oft wird diese Schicht nochmals in den unteren MAC[1]-Layer und den oberen LLC[2]-Layer unterteilt. Die Adressierbarkeit einer Netzkomponente wird bereits im MAC-Layer in Form seiner MAC-Adresse sichergestellt.

2.3 Layer 3: Netzwerk- oder Vermittlungsschicht

Hier erfolgt die optimale Wegewahl, das Routing. Wenn es für die Erreichbarkeit innerhalb eines Netzwerkes mehrere Alternativen gibt, so müssen Protokolle der Netzwerkschicht dafür sorgen, dass für die letztlich durchzuführende Datenübertragung eine optimale Route ermittelt wird. Bei Ausfällen ist eine Umleitung zu generieren. Der Auf- und Abbau von Verbindungen wird initiiert. Die Kommunikation erfolgt an dieser Stelle bereits unabhängig vom eingesetzten Übertragungsmedium. (mögliches Protokoll: IP)

2.4 Layer 4: Transportschicht

Diese Schicht gewährleistet die zuverlässige, transparente Datenübertragung zwischen den Endknoten. Es erfolgt eine Ende-zu-Ende-Kontrolle der Daten. Layer 4 übernimmt auch Schnittstellenfunktionen zwischen den übergeordneten anwendungsorientierten Schichten und den unteren netzwerkorientierten Schichten. Mechanismen wie Flusskontrolle und Segmentierung von Frames[3] finden in der vierten Schicht statt. Eine Orientierung im Netzwerk ist für den Anwender nun nicht mehr erforderlich, da die verwendeten Protokolle logisch wie physikalisch transparent arbeiten.

2.5 Layer 5: Sitzungsschicht

Die Sitzungsschicht koordiniert und synchronisiert die Kommunikation zwischen Anwendungsprozessen. Bei Verbindungsunterbrechung (Session Interrupts) können die stattgefundenen Dialoge durch Synchronisierung meist wiederhergestellt werden. So genannte Session Recoverys ermöglichen unter Umständen ein Aufsetzen des Dialogs an der Stelle, an der die Unterbrechung einsetzte. Bisher übertragene Daten gehen demnach nicht verloren.

[1] Media Access Control
[2] Logical Link Control
[3] Datenblöcke

2.6 Layer 6: Datendarstellungsschicht

Hier wird anwendungsspezifische Formatierung durchgeführt, damit unterschiedliche Systeme die übertragenen Daten einheitlich interpretieren können. Es findet z.B. die Konvertierung zwischen unterschiedlichen Zeichensätzen wie ASCII oder EBCDIC statt. Dazu wird meist eine gemeinsame Syntax zur Datenrepräsentation ausgewählt (Transfersyntax) und die Daten werden vor der Übertragung in diese konvertiert und nach der Übertragung in die des Zielsystems.

2.7 Layer 7: Anwendungsschicht

Die Anwendungsschicht stellt die eigentliche Schnittstelle zwischen den in zwei Systemen implementierten Anwendungen dar. Hier erfolgt der Einsatz unzähliger Softwareprodukte durch den Benutzer. Er kann die Anwendungsprogramme wechseln, ohne dabei die Hardware berücksichtigen zu müssen. Beispiele für Dienste der Anwendungsschicht sind: E-Mail, Remote-Login oder Dateitransfer.

3 TCP/IP Architekturmodell

Das TCP/IP-Referenzmodell ist nach seinen wichtigsten Protokollen benannt – TCP und IP. Die Netzarchitektur beruht auf den Vorschlägen, die bei der Fortentwicklung des ARPANETs gemacht wurden. Wie oben schon erwähnt wurde, ist das TCP/IP-Modell zeitlich vor dem OSI-Referenzmodell entstanden, deshalb sind auch die Erfahrungen des TCP/IP-Modells mit in die OSI-Standardisierung eingeflossen. Auch das TCP/IP-Architekturmodell besteht aus unabhängigen Schichten. Im Gegensatz zum OSI-Modell existieren in diesem Modell nur vier Schichten. Diese sind im rechten Teil von Abbildung 1 grafisch dargestellt. Das Prinzip der Kapselung der einzelnen Schichten ist für TCP/IP in Abbildung 2 deutlich zu sehen. Abbildung 3 zeigt konkrete Protokolle und Anwendungen.

3.1 Netzwerkschicht (Network Layer)

Die Netzwerkschicht des TCP/IP-Architekturmodells fasst die physikalische Schicht und die Datensicherungsschicht des OSI-Modells zusammen. Hier können verschiedenste Technologien wie Ethernet, X.25, Token Ring oder ATM zum Einsatz kommen.

3.2 Internetschicht (Internet Layer)

Die Internetschicht stellt der Transportschicht einen unzuverlässigen, verbindungslosen Dienst zur Verfügung. Das eingesetzte Protokoll heisst Internet-Protokoll (IP). Es arbeitet mit anderen Protokollen der Internetschicht wie

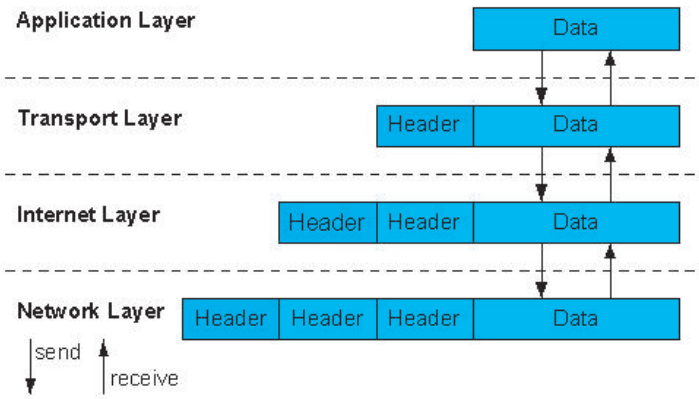

Abbildung 2: Kapselung der Schichten in TCP/IP

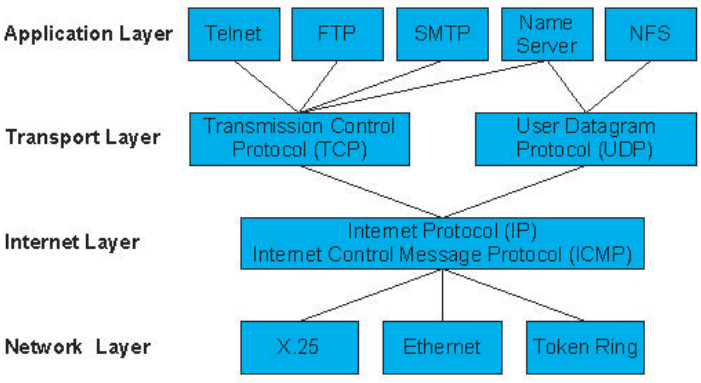

Abbildung 3: Übersicht über die Schichten in TCP/IP

z.B. ICMP zusammen. Die Kernaufgabe der Internetschicht ist das Routing. Routing ist notwendig, wenn ein Paket den Empfänger nicht direkt erreichen kann, sondern dazu über Vermittlungsrechner geschickt werden muss. Wie in der Einleitung erläutert, sollten alle Vermittlungsrechner jedes Paket auf möglichst optimalem Weg weitersenden.

3.3 Transportschicht (Transport Layer)

Die Transportschicht nutzt die Internetschicht, um Dienste zu implementieren. UDP stellt der Anwendungsschicht einen unzuverlässigen, verbindungslosen Dienst zur Verfügung. TCP sorgt für einen zuverlässigen, verbindungsorientierten Datentransport. Beide Protokolle ermöglichen im Gegensatz zu den unterliegenden Schichten Ende-zu-Ende-Kommunikation.

3.4 Anwendungsschicht (Application Layer)

Das TCP/IP-Modell hat keine Sitzungs- und Darstellungsschicht, da die meisten Anwendungen sie nicht benötigen. Anwendungen der Anwendungsschicht sind beispielsweise HTTP (Hyper Text Transfer Protocol), FTP (File Transfer Protocol), SSH (Secure Shell), SMTP (Simple Mail Transfer Protocol), NNTP (Network News Transfer Protocol) oder DNS (Domain Name System).

4 ARP/RARP ((Reverse) Address Resolution Protocol)

In einem Netzwerk werden alle Rechner, die miteinander kommunizieren wollen, mit ihrer physikalischen Adresse angesprochen. Diese ist im allgemeinen nicht veränderbar und weltweit eindeutig. Für die Kontaktaufnahme wird diese Adresse jedoch nicht genutzt, sondern eine logische Adresse verwendet, die für den Aufbau leistungsfähiger Adressstrukturen wesentlich besser geeignet ist. Typische Beispiele für physikalische Adressen sind die 48-Bit MAC-Adressen im Ethernet und für logische Adressen die 32-Bit IP-Adressen. Allerdings ist es für den Netzwerkkartentreiber nicht möglich, einen Rechner über seine logische Adresse anzusprechen. Deshalb müssen logische und pysikalische Adressen ineinander umgewandelt werden. Diese Funktionalität bieten das ARP bzw. RARP-Protokoll.

4.1 ARP

Das Address Resolution Protocol wandelt logische Adressen in physikalische Adressen um. Dazu wird eine ARP-Adresstabelle (auch ARP-Cache genannt) benutzt, in der die Zuordnung gespeichert ist. Findet ein Rechner

eine logische Adresse nicht in seiner Tabelle, dann wird ein Broadcast generiert, der als ARP-Request von allen im Netz befindlichen Rechnern empfangen wird. Erkennt ein Rechner in dem ARP-Request seine eigene logische Adresse, so schickt er dem anfragenden Rechner seine pysikalische Adresse in einem ARP-Reply zurück. Dieser trägt daraufhin die Informationen in seine Tabelle ein. Zur Vermeidung unnötiger Broadcasts trägt auch der antwortende Rechner die Adressinformationen in seinen Cache ein. Jeder Cache-Eintrag hat nur eine begrenzte Lebenszeit. Damit wird verhindert, dass beim Austausch von logischen Adressen dauerhafte Inkonsistenzen entstehen. ARP-Datagramme werden nicht über Routergrenzen hinweggeführt. ARP-bedingte Broadcasts gelangen somit nicht in alle theoretisch erreichbaren Netzwerke, sondern bleiben im lokalen Router-Netzwerk. Wenn zwei Netzwerke auf Schicht 2 über Brücken und Switches verbunden werden, können sich ARP-Broadcast weit ausbreiten. Hier besteht die Gefahr von "Denial of Service" Angriffen.

4.2 RARP

Das Reverse Address Resolution Protocol kommt z.B. zum Einsatz, wenn "Diskless Clients" eingesetzt werden, die beim Booten ihre logische Adresse nicht kennen. Ein Host sendet einen RARP-Request mit seiner pysikalischen Adresse aus und ein im Netzwerk befindlicher ARP-Server sendet ein ARP-Reply mit der entsprechenden logischen Adresse zurück.

5 IP (Internet Protocol)

Gemeinsam mit dem in Layer 4 angesiedelten Transmission Control Protocol (TCP) bildet das Internet Protocol als verbindungsloser Datagramm-Dienst das zentrale Protokollpaar innerhalb der TCP/IP-Architektur. Auf dieser Protokollebene werden keine Aufgaben zur Datensicherung übernommen, dies geschieht auf der übergeordneten Protokollschicht. IP übernimmt auch keine Garantie für den Reihenfolgeerhalt der Pakete. Das IP-Protokoll dient der Abstraktion der Besonderheiten der unterliegenden Netzwerkschicht.
Die Leistungsfähigkeit eines Datagrammverkehrs ergibt sich aus der flexiblen Adressierbarkeit (es können eine, mehrere oder auch alle Stationen im Netz angesprochen werden), der hohen Übertragungsgeschwindigkeit (Datagramme brauchen nicht bestätigt zu werden, Algorithmen zur Fehlerkorrektur fehlen) und einem variablen Routing (die Wegewahl ist nicht festgelegt, sondern kann variieren; Leitungsstörungen können durch Alternativrouten umgangen werden).
Abbildung 4 beschreibt den Aufbau eines IP-Pakets. Sie ist so zu verstehen, dass jede Zeile 32 Bit entspricht und die Zeilen von oben nach unten hintereinandergeschrieben ein Paket ergeben. Der Header ist mindestens 20

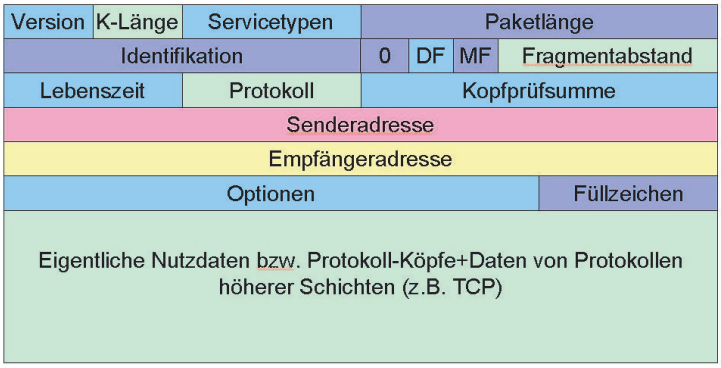

Abbildung 4: Aufbau eines IPv4-Segments

Bytes und maximal 60 Bytes lang (wenn das Optionen-Feld mit 40 Byte maximal benutzt wird). Insgesamt ist die maximale Paketgröße (Header incl. Optionen + Daten) 65.535 Bytes.

- **Version (4 Bit):** Das erste Feld enthält Angaben zur benutzten Versionsnummer des IP-Protokolls. Im Moment ist Version 4 aktuell; die Ablösung durch Version 6 steht u.a. aufgrund des in IPv4 zu kleinen Adressraums bevor.
- **Kopflänge / IHL - Internet Header Length (4 Bit):** Länge des Header einschliesslich der Optionen in 32 Bit-Worten. Der dezimale Wert 5 bedeutet also, dass nur der minimale 20-Byte-Header ohne Optionen existiert ($5 * 32$ Bit $= 20$ Byte). Der dezimale Maximalwert 15 beschränkt somit die maximale Länge des Optionen-Feldes auf ($15 *$ 32Bit) $-$ 20Byte $=$ 40Byte. Dass die kleinstmögliche Headerlänge fünf 32-Bit Worte sind, ist aus Abbildung 4 ersichtlich.
- **Servicetypen / Type of Service (8 Bit):** Dieses Feld enthält eine Beschreibung der Qualität des angeforderten Service. In der Praxis bleibt dieses Feld von vielen Routern unbeachtet. Die ersten 3 Bits geben den Prioritätslevel an:

000	normal
001	priority
010	immediate
011	flash
100	flash override
101	critical
110	internet control
111	network control

Beträgt der Wert eines der folgenden 4 Bits "1", so sind die Eigenschaft dieses Bit aktiviert:

D-Bit	Delay (fordert Verbindung mit kurzer Verzögerung an)
T-Bit	Throughput (fordert hohen Datendurchsatz an)
R-Bit	Reliability (fordert hohe Sicherheit; Pakete sollen nicht verworfen werden)
C-Bit	Cost (fordert Route zu niedrigen Kosten)

Das C-Bit bleibt für gewöhnlich ungenutzt.

- **Paketlänge / Total Length (16 Bit):** Gesamtlänge des Datagramms. Die Differenz zwischen dem Wert dieses Feldes und der Kopflänge ergibt die Länge der transportierten Nutzdaten. Infolge der Feldlänge von 16 Bits kann hier ein Wert von maximal 65.535 (dezimal) eingetragen werden. Diese Datagramm-Länge wird von den meisten Netzen nicht unterstützt; somit muss fragmentiert[4] werden. Die generierten Einzelfragmente des Original-Datagramms führen in diesem Feld allerdings nicht mehr den ursprünglichen Wert, sondern beschreiben die Gesamtlänge des vorliegenden Einzelfragments. Näheres ist unter Kapitel 5.3 erklärt.

- **Identifikation / Identification (16 Bit):** Integer-Wert, der zur Nummerierung fragmentierter Datagramme verwendet wird.

- **Anzeigen / Flags (3 Bit):** Steuerung der Fragmentierung, sofern sie angewendet wird. Das erste Bit (high-order-bit) wird nicht benutzt und enthält den Wert "0". Das mittlere Bit wird auch DF-Bit oder *Do-not-fragment-Bit* genannt. Hier bedeutet der Wert "1" dass fragmentieren nicht erlaubt ist. Das dritte Bit ist das MF-Bit (*More-Fragments-Bit*). Es gibt an, ob noch ein weiteres Fragment folgt (Wert "1") oder das letzte Fragment des Datagramms vorliegt (Wert "0").

- **Fragmentabstand / Fragment Offset (13 Bit):** Zur korrekten Zusammenführung fragmentierter Datagramme ist ein Hinweis erforderlich, der Angaben zur Position der Daten des jeweiligen Datagrammfragments innerhalb des Original-Datagramms (vor der Fragmentierung) macht. Daher erfolgt hier die Angabe der Anzahl aller vorherigen Datagramm-Fragmente, die jeweils 8 Bytes Daten umfassen. Da sich mit 13 Bit maximal der Wert 8.192 darstellen lässt, ergibt sich daraus die max. Datenlänge von 65.536 Bytes. Allerdings können zusammengehörige, fragmentierte IP-Pakete keine 65.536 Byte Nutzdaten beinhalten, da das entsprechende nicht fragmentierte Paket nur 65.536 Byte abzüglich min. 20 Bytes Headerlänge groß sein kann.

[4] Aufteilen eines Pakets in mehrere kleinere

- **Lebenszeit / TTL - Time to Live (8 Bit):** Dieses Feld wird vom sendenden Rechner mit einer Zahl größer Null initialisiert. Es sollte ursprünglich die Paketlebenszeit in Sekunden beinhalten. Aufgrund der mittlerweile sehr leistungsfähigen Router, in denen die Pakete nicht annähernd eine Sekunde verweilen, verringert aber jeder Router dieses Feld immer um eins, unabhängig davon, wie lange das Paket aufgehalten wird. Ist dieses Feld bei einem Paket gleich Null, dann wird das Paket verworfen. So wird verhindert, dass Pakete z.B. bei Routingfehlern endlos in einem Netz kreisen und Ressourcen verschwendet werden. Natürlich darf der Wert dieses Feldes bei der Initialisierung nicht zu klein gewählt werden, denn sonst kann es passieren, dass die Pakete ihren Empfänger gar nicht erreichen können. Der Rechner, der ein Paket verwirft, kann dem Absender über das ICMP-Protokoll eine Fehlermeldung zukommen lassen (siehe unten).

- **Protokoll / Protocol (8 Bit):** Hier wird ein Wert zur Identifizierung des übergeordneten Transportprotokolls angegeben, an das die Daten des IP-Pakets weitergegeben werden sollen:

Protokollnummer	Protokollname
1	ICMP
6	TCP
17	UDP
89	OSPF

- **Kopfprüfsumme / Header Checksum (16 Bit):** Zur Sicherung der Informationen im Paketkopf existiert eine Prüfsumme. Sie bezieht die transportierten Nutzdaten nicht mit ein. Zur Berechnung wird das Einerkomplement der Summe aller Einerkomplemente aller 16 Bit-Blöcke des Headers gebildet. Dabei wird das Feld der Prüfsumme als Null angenommen. Stimmt die Prüfsumme nicht, wird das Paket verworfen. Siehe auch 10.1 hierzu.

- **Senderadresse / Source IP-Address (32 Bit):** Angabe der IP-Adresse des sendenden Rechners (siehe 5.1).

- **Empfängeradresse / Destination IP-Address (32 Bit):** Angabe der IP-Adresse des empfangenden Rechners.

- **Optionen / Options (0 bis 40 Byte):** Die Länge des Optionsfeldes ist variabel. Es dient der flexiblen Erweiterbarkeit. Eine mögliche Nutzung ist z.B. die Speicherung von Adressen der durchlaufenen Router (ggf. mit Zeitstempeln). Dies kann der Fehlerbehebung dienen und wird in nahezu allen TCP/IP-Implementierungen durch den "Traceroute"-Befehl implementiert.

- **Füllzeichen / Padding:** Das Padding-Feld schließt den IP-Header ab und ergänzt ihn auf den nächsten 32-Bit Block.
- **Nutzdaten:** Hier stehen die eigentlich zu transportierenden Daten bzw. die Protokollköpfe und Nutzdaten der nächsthöheren Schicht (z.B. TCP).

5.1 IP-Adressen

Wie aus der Beschreibung des IP-Headers hervorgeht, sind IPv4-Adressen 32 Bit lang. Zur besseren Lesbarkeit werden IP-Adressen als Gruppe von 4 Dezimalzahlen geschrieben, die durch Punkte voneinander getrennt werden. So entspricht der binären Adresse 10000110011001000000100101001101 zum Beispiel die Dezimalzahlengruppe 134.100.9.77. IP-Adressen sind mit einigen Ausnahmen global eindeutig, um zweifelsfreie Adressierbarkeit zu gewährleisten. Jede IP-Adresse besteht aus einem Netz- und einem Hostanteil. Früher gab es Netzklassen mit festgelegter Länge von Netz- und Hostnummer:

Klasse A:	"0" + 7 Bit Netz-ID + 24 Bit Host-ID
Klasse B:	"10" + 14 Bit Netz-ID + 16 Bit Host-ID
Klasse C:	"110" + 21 Bit Netz-ID + 8 Bit Host-ID
Klasse D:	"1110" + 28 Bit Multicast-Adresse
Klasse E:	"1111" + 28 Bit für experimentelle Nutzung

In der Klasse A gibt es 126 Netze mit je 16.646.144 Hosts, in der Klasse B 16.256 Netze mit je 65.024 Hosts und in der Klasse C 2.080.768 Netze mit je 254 Hosts. Zusätzlich gibt es besondere Adressen, wie z.B. private. Diese werden im Internet nicht weitergeleitet und tragen zur Linderung des Problems des zu kleinen Adressraums bei, indem mehrere Rechner mit privaten Adressen über eine öffentliche Adresse kommunizieren können.
Die Aufteilung in feste Netzklassen bringt Nachteile mit sich. Wenn eine Firma beispielsweise 300 Rechner an das Internet anschließen möchte, brauchte sie dafür gleich eine Klasse-B Netz mit möglichen 65.024 Hosts. Es wurde also ein großer Teil des Adressraum verschwendet. Diesem Problem begegnete man mit Einführung der Subnetzadressen (subnetwork mask). Die 32-Bit lange Subnetzadresse ist so zu interpretieren, dass jede Stelle, an der eine binäre Null steht, die Host-ID der IP-Adresse identifiziert, jede Stelle an der eine binäre 1 steht die Netz-ID. Beispiel:

Netz-ID	Host-ID
11111111 11111111 111	00000 00000000 Subnetzadresse
10111110 10001000 000	00000 00000001 IP-Adresse (190.136.0.1)

Hier ist der Netzteil der Subnetzadresse 19-Bit lang und der Hostteil 13-Bit. Folglich können in diesem Subnetz 8192 Rechner adressiert werden.

Die Subnetzadresse wird nicht in den IP-Paketen mitgeführt, sondern muss nur den Routern und Hosts bekannt sein. Dieses Prinzip der Adressierung von Adressen ohne die klassischen A-/B-/C-Netzbezeichnungen nennt man auch CIDR, was für *Classless Inter Domain Routing* steht. Dieser Begriff beschreibt zudem noch eine einfachere Routingstruktur, die der des Telefonnetzes ähnelt. Die genaue Spezifizierung ist in [6] nachzulesen.

5.2 Ports

Über die IP-Adresse kann nur ein bestimmter Rechner (oder Netz) angesprochen werden. Eine bestimmter Dienst auf einem Rechner wird über die Portnummer angesprochen. Für wichtige Standarddienste sind bestimmte Portnummern reserviert.
Würde eine Verbindung Client-IP → Server-IP:Server-Port stattfinden, bestände das Problem, daß der Server bei mehreren gleichzeitig geöffneten Verbindungen von einem Client auf den gleichen Dienst die Verbindungen nicht auseinanderhalten könnte. Um dieses Problem zu umgehen, öffnet der Client auch einen derzeit unbenutzten Port und teilt dem Server mit, daß er die Verbindung auf diesem Port etablieren möchte. Eine Verbindung zwischen Client und Server ist immer durch Client-IP:Client-Port → Server-IP:Server-Port definiert, dem sogenannten Socket. Hierdurch kann der Server die einzelnen Pakete den jeweiligen Verbindungen zuordnen, da die Portnummern im TCP- und UDP-Paket enthalten sind.
An dieser Stelle sei der Begriff der *well known ports* genannt, der die übliche Zuordnung von Diensten zu Portnummern beschreibt. Ein Beispiel hierfür wäre der Port 22, der standardmässig für SSH benutzt wird.

5.3 IP-Fragmentierung

Die Fragmentierung eines IP-Pakets wird dann notwendig, wenn es in ein Netz geleitet werden soll, das nur eine kleinere Paketgröße erlaubt, als das zu transportierende Paket hat. Hier einige Beispiele der maximalen Transfergrößen (MTU[5]):

Ethernet	1500 Bytes
X.25	576 Bytes
Token Ring (16MBit/s)	17914 Bytes

Bei Bedarf nimmt ein Router die Fragmentierung eigenständig vor. Sie wird von den Feldern DF, MF, Identifikation und Fragmentabstand des IP-Headers gesteuert. Muss ein Paket zum weiteren Transport fragmentiert werden, wird das DF-Bit geprüft. Ist es "1", darf das Paket nicht fragmentiert werden und wird verworfen. Das MF-Bit ist bei allen Teilpaketen bis

[5]Maximum Transfer Unit

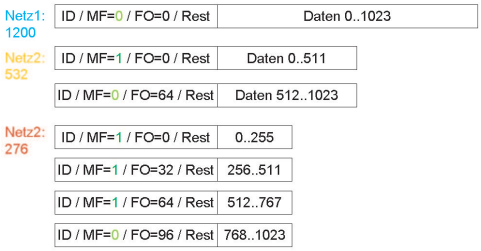

Abbildung 5: Fragmentierung in Netzen mit unterschiedlich grosser MTU

auf dem letzten gesetzt. So erkennt der Zielrechner die Länge des ursprünglichen Pakets. Das Feld Identifikation ist bei allen Teilpaketen gleich und hilft dem Zielrechner zu erkennen, welche Teilpakete zusammengehören. Die Reihenfolge der Pakete wird durch das Feld Fragmentabstand bestimmt. Es enthält die Position des Fragments innerhalb des urprünglichen Pakets.
Sobald der Zielrechner den einen Teil eines fragmentierten Pakets empfängt, setzt er einen Timer. Wenn nach dessen Ablauf nicht alle Fragmente empfangen wurden, wird das gesamte Paket verworfen. Fragmentierung wird bei Bedarf wiederholt angewendet.Abbildung 5 zeigt hierzu ein Beispiel.

Es soll ein Paket von 1044 Bytes Länge (1024 Bytes Nutzdaten + 20 Bytes Header) über Netze mit den maximalen Paketgrößen von 1200 Bytes, 532 Bytes und 276 Bytes übertragen werden. Das DF-Bit ist nicht gesetzt. Die erste Zeile stellt das ursprüngliche Paket dar. In dem ersten Netz mit MTU 1200 kann es problemlos transportiert werden, da es nur 1044 Bytes lang ist. Wird es in das zweite Netz geleitet, muss es wegen der MTU von 532 fragmentiert werden. Zeile 2 und 3 zeigen die beiden neuen Pakete. Zu beachten ist, dass im ersten Paket das MF-Bit gesetzt ist und im zweiten nicht, da ja nur dem ersten Paket ein weiteres folgt. Die Daten des ersten Pakets beginnen relativ zum ursprünglichen Paket bei Null (also FO=0) und sind 512 Bytes lang (Bytes 0 bis 511 des ursprünglichen Pakets). Das zweite Paket enthält ebenfalls 512 Bytes Daten. Dabei handelt es sich um die Datenbytes 512 bis 1023 des ursprünglichen Pakets. Da die Daten relativ zum ursprünglichen Paket bei Byte 512 beginnen, enthält das FO-Feld den Wert 512 geteilt durch 8, also 64. Nach dem gleichen Schema werden die Pakete weiter aufgeteilt, wenn sie durch das dritte Netz transportiert werden müssen.

6 ICMP (Internet Control Message Protocol)

ICMP ist ein IP-basierendes Protokoll, das für die Übermittlung von Steuerinformationen und Fehlermeldungen verantwortlich ist, beispielsweise wenn ein Ziel-Host nicht erreichbar ist. Es benutzt IP, als wäre es selbst ein höheres

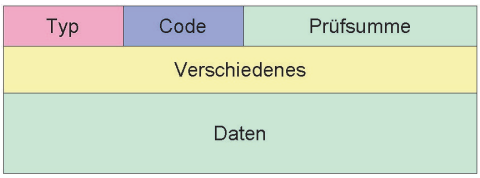

Abbildung 6: Aufbau eines ICMP-Segments

Protokoll, obwohl ICMP integraler Bestandteil von IP ist. Dies widerspricht der Trennung der einzelnen Schichten nach dem OSI-Modell. Um beim Melden von Fehlern große Netzlast und Endlosschleifen zu vermeiden, meldet ICMP keine Fehler im eigenen Protokoll. ICMP dient auch zu Testzwecken. Den Aufbau eines ICMP-Segments zeigt Abbildung 6.

- **Typ / Type (8 Bit):** Enthält den Typ der Nachricht. Folgende ICMP-Nachrichten werden unterschieden:

  Fehlermeldungen:
  3 Destination unreachable (Zielstation nicht erreichbar)
  4 Source quench (Buffer-Ressourcen verbraucht)
  5 Redirect (Pfadumleitung)
  11 Time Exceeded (Timer abgelaufen)
  12 Parameter Problem (Parameterproblem)

  Informationsmeldungen:
  0 Echo reply
  8 Echo request
  13 Time stamp
  14 Time stamp reply
  15 Information request
  16 Information reply
  17 Address mask request
  18 Address mask reply

- **Code (8-Bit):** In Abhängigkeit vom Message-Typ erfolgt durch den ICMP-Code eine weitere Differenzierung der ICMP-Nachricht. Beispiel für Typ 3 (destination unreachable):

  0 Netz nicht erreichbar
  1 Host nicht erreichbar
  2 Protokoll nicht erreichbar
  3 Port nicht erreichbar
  4 Fragmentierung erforderlich, aber DF-Bit gesetzt
  5 Source Route nicht erreichbar

Abbildung 7: Aufbau eines UDP-Segments

- **Prüfsumme / Checksum (8 Bit):** Enthält eine Prüfsumme über das ICMP-Paket beginnend mit dem Typ-Feld.
- **Verschiedenes (32 Bit):** Enthält spezifische Daten für die einzelnen Nachrichtentypen. Bei manchen Typen bleibt dieses Feld leer und besteht aus Nullen.
- **Daten / Data (Länge variabel):** Enthält bei den meisten Typen den IP-Header des fehlererzeugenden Pakets zusammen mit den ersten 64 Bit an Daten.

7 UDP (User Datagramm Protocol)

UDP ist neben TCP das zweite wichtige Protokoll auf der Transportebene. Es arbeitet verbindungslos und bietet keine gesicherte Übertragung, keine Flusskontrolle und keinen Garantie auf Reihenfolgeerhalt. Dafür ist es einfach aufgebaut und kommt ohne großen Overhead aus, was es attraktiv für Echtzeitanwendungen macht. Der gegenüber TCP viel einfachere Aufbau ist schon in Abbildung 7 ersichtlich.

- **Sender-Port / Source-Port (16 Bit):** Portnummer des sendenden Prozesses
- **Empfänger-Port / Destination-Port (16 Bit):** Portnummer des empfangenden Prozesses
- **Länge / Length (16 Bit):** Länge des gesamten UDP-Datagramms incl. Header
- **Prüfsumme / Checksum (16 Bit):** Prüfsumme der Daten, des UDP-Headers und des Pseudo-Headers (s.o.)

8 TCP (Transmission Control Protocol)

TCP ist das wichtigste Protokoll der Transportschicht. Es bietet der Anwendungsschicht einen zuverlässigen, verbindungsorientierten Transportdienst. TCP garantiert die richtige Reihenfolge der Daten und ermöglicht mit Hilfe

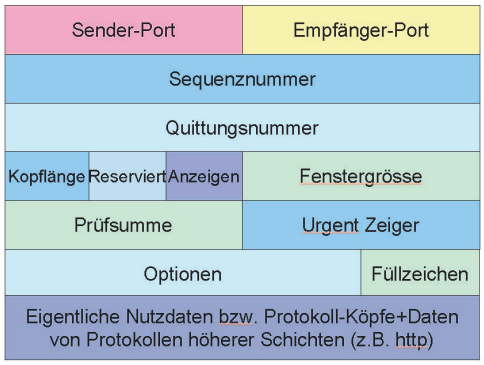

Abbildung 8: Aufbau eines TCP-Segments

eines Fensteralgorithmus Ende-zu-Ende-Flusskontrolle. Es benutzt verschiedene Algorithmen zur Fehlererkennung und -behandlung. Die grundlegende Funktionsweise lässt sich folgendermassen beschreiben:
Eine Anwendung veranlasst TCP zum Aufbau einer Verbindung und übergibt dann datenstromorientiert (in Bytes) die Daten. Der möglicherweise sehr lange Datenstrom wird von TCP geteilt/segmentiert und jedes Segment mit einem eigenen (TCP-)Header versehen. Nun erfolgt die Übergabe an das Internet-Protokoll in Layer 4. Nach Transport der Daten über das Netz erfolgt nach Entfernung des IP-Headers die Übergabe zurück an TCP des Empfängers. Nun muss der ankommende Datenfluss auf Fehler geprüft werden, die ankommenden Segmente sortiert werden, die Header entfernt werden und die Daten der Zielanwendung zugeführt werden.
Abbildung 8 zeigt den Aufbau eines TCP-Segments.

- **Sender-Port / Source-Port (16 Bit):** Dieses Feld beinhaltet den Quellport der Anwendung, die TCP zur Datenübertragung nutzt.
- **Empfänger-Port / Destination-Port (16 Bit):** Dieses Feld beinhaltet den Zielport der Anwendung, die TCP zur Datenübertragung nutzt.
- **Sequenznummer / Sequenz Number (32 Bit):** Die Sequenznummer führt eine Nummerierung des jeweils ersten Daten-Bytes innerhalb des Gesamt-Stroms durch. Näheres zur Funktion dieses und weiterer Felder folgt unter 9.2.
- **Bestätigungsnummer / Acknowledgement Number (32 Bit):** Alle empfangenen Daten werden dem Sender durch die Acknowledgement Number bestätigt. Sobald das ACK-Bit gesetzt ist, enthält dieses Feld den Wert der Sequenznummer, die der Empfänger nun vom Sender erwartet.

- **Kopflänge / Data Offset (4 Bit):** Dieses Feld gibt die Länge des TCP-Headers in 32-Bit-Blöcken an. Erforderlich ist diese Angabe, da das Optionsfeld variable Länge hat.
- **Reserviert / Reserved (6 Bit):** Dieses Feld wird zur Zeit nicht benutzt und enthält Nullen.
- **Anzeigen / Flags (6 Bit):**
  - *Urgent-Pointer-Flag:* Durch dieses Bit werden Vorrang-Daten gekennzeichnet (z.B. für Unterbrechung von Prozessen)
  - *Acknowledgement-Flag:* Ist dieses Bit gesetzt, wird mit diesem Segment u.a. der Empfang von Daten bestätigt.
  - *Push-Flag:* Die Daten sollen sofort an die Anwendung hochgereicht und nicht gepuffert werden. Dies ist zum Beispiel für Telnetdialoge wichtig.
  - *Reset-Flag:* Teilt dem Empfänger mit, dass die Verbindung aufgrund eines nicht näher bestimmten Fehlers beendet werden soll.
  - *Synchronization-Flag:* Wird zum Verbindungsaufbau benutzt.
  - *Final-Flag:* Kommt beim Verbindungsabbau zum Einsatz.
- **Fenstergröße / Window Size (16 Bit):** Dient zur Flusskontrolle und gibt die Anzahl der Bytes an, die der Sender unbestätigt senden darf. Im Kapitel 9.3 wird genauer auf dieses Feature eingegangen.
- **Prüfsumme / Checksum (16 Bit):** Enthält die Prüfsumme aus TCP-Header, Daten und dem Pseudo-Header. Der Pseudo-Header besteht aus den IP-Adressen, der Protokollnummer und der Angabe über die Länge des TCP-Segments.
- **Urgent-Zeiger / Urgent-Pointer (16 Bit):** Der Urgent-Pointer weist auf das Ende der Vorrang-Daten innerhalb eines TCP-Segments. Er stellt einen Offset-Wert dar, der in Addition mit der Sequenznummer die beschriebene Datenposition identifiziert.
- **Optionen / Options (0 bis 352 Bit):** Das Optionsfeld besteht aus einem Optionstyp, einer Optionslänge und ggf. aus den Optionsdaten.
- **Füllzeichen / Padding:** Das Padding-Feld schließt den TCP-Header ab und ergänzt ihn auf den nächsten 32-Bit Block.
- **Nutzdaten:** Nach dem Header folgen die Nutzdaten, also die Daten des transportierten Anwendungsprotokolls. Beispiele hierfür sind HTTP oder FTP.

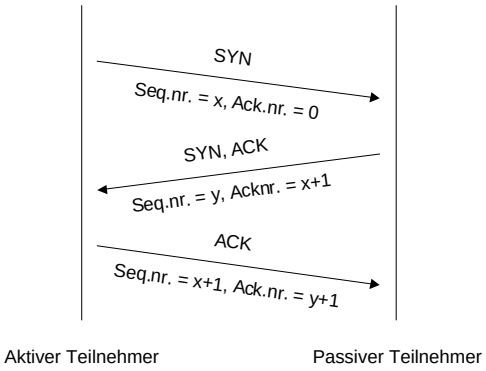

Abbildung 9: TCP-Verbindungsaufbau

9 TCP Verbindungsablauf

Verbindungen, die TCP benutzen, lassen sich in drei Teile untergliedern: Den Aufbau der Verbindung, den Datenaustausch und den Abbau der Verbindung. Diese drei Schritte sollen nun genauer erläutert werden.

9.1 Verbindungsaufbau

Für den Aufbau einer TCP Verbindung ist die Bestätigung des Verbindungswunsches durch beide Seiten nötig. Diese Prozedur wird auch des öfteren als *three-way-handshake* bezeichnet, was sich aus der Abfolge von TCP Paketen nachvollziehen lässt.

9.2 Sequenznummern

Sequenznummern garantieren, dass auch, wenn Pakete bei der Übermittlung in der Reihenfolge vertauscht worden sind, diese in der richtigen Reihenfolge beim Empfänger verarbeitet werden können. Zu Beginn der Verbindung wählt jede Seite eine zufällige Zahl als initiale Sequenznummer, diese wird dann bei jedem Paket um die Anzahl der Bytes an Nutzdaten in diesem Paket erhöht; mindestens jedoch um eins.
Wenn der *three-way-handshake* erfolgreich abgeschlossen ist, können nun Daten gesendet werden.

9.3 Datenaustausch

Für jedes gesendete Datenpaket wird ein Timer gestartet und auf die Bestätigung dieses Pakets durch ein ACK-Paket gewartet. Wenn dieser Timer abgelaufen ist und vorher keine Empfangsbestätigung eingetroffen ist, wird das Paket erneut gesendet. Die Bestätigung enthält unter anderem die als nächste erwartete Sequenznummer.

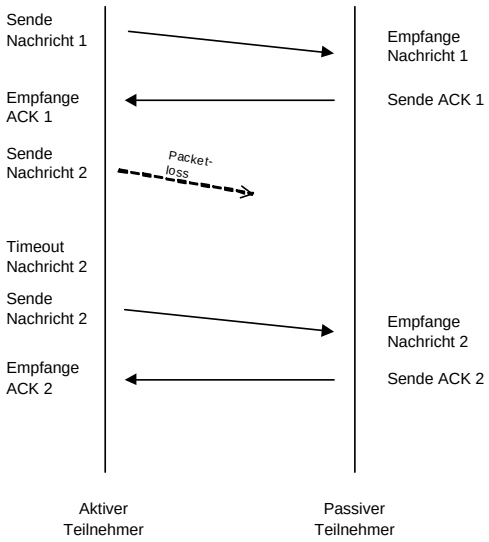

Abbildung 10: Austausch von Daten mit TCP

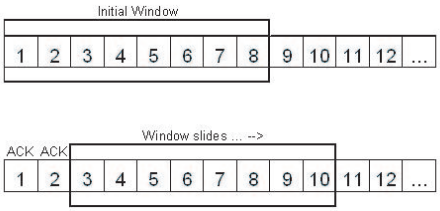

Abbildung 11: Sliding Window Verfahren

9.3.1 sliding window

Wenn nur ein Paket zur Zeit gesendet werden kann, hat dies eine sehr geringe Effizienz bei Leitungen mit hoher Latenz (wie zum Beispiel bei analogen Modemleitungen) zur Folge. Um dem entgegenzuwirken, entwickelte man das *sliding window* Verfahren. Bei dieser Art von Datenübertragung wird ein Satz von Paketen losgeschickt, die nach und nach bestätigt werden. In der Abbildung hat das Fenster die Grösse 8. Wenn die Bestätigungen für das erste Paket eingetroffen ist, wird das Fenster weitergeschoben.

9.3.2 variable window

Das in 9.3.1 vorgestellte Verfahren kann noch verfeinert werden, um eine Flusskontrolle zu realisieren. Zu diesem Zweck wird die Grösse des *sliding window* dynamisch angepasst, so dass der Durchsatz der Verbindung ge-

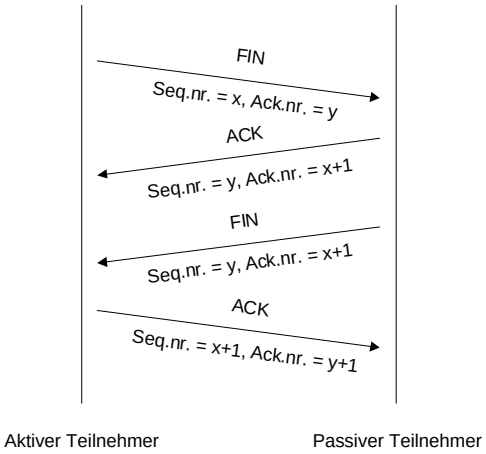

Abbildung 12: TCP-Verbindungsabbau

drosselt werden kann, wenn es zum Beispiel auf einer Seite zu Verarbeitungsschwierigkeiten kommt.

9.4 Verbindungsabbau

Zum Abbau der Verbindung wird von jeder Seite ein Paket mit einem gesetzten FIN Flag gesendet, das dann von dem Gegenüber mit einem ACK Paket bestätigt werden muss. Selbst wenn eine Station die Verbindung schliessen will, kann die andere Station noch Daten senden, bis auch sie fertig ist.
Eine abruptere Art des Verbindungsabbaus ist das Abbrechen mittels einem Paket, in dem das RST Flag gesetzt ist. Wird ein solches Paket empfangen, gilt die Verbindung als geschlossen.

9.5 Zustandsautomat

Alle möglichen Zustände der Stationen und die gesendeten oder empfangenen Pakete lassen sich in einem Zustandsautomaten wie in Abbildung 13 darstellen.

10 Fehlerbehandlung

In diesem Kapitel werden verschiedene Fehler und deren Behandlung in der IP Protokollfamilie behandelt.

10.1 Checksummenfehler

Die Checksumme in einem IP Paket ist lediglich zur Fehlererkennung geeignet und nicht zur Fehlerkorrektur, da der Algorithmus sehr schnell be-

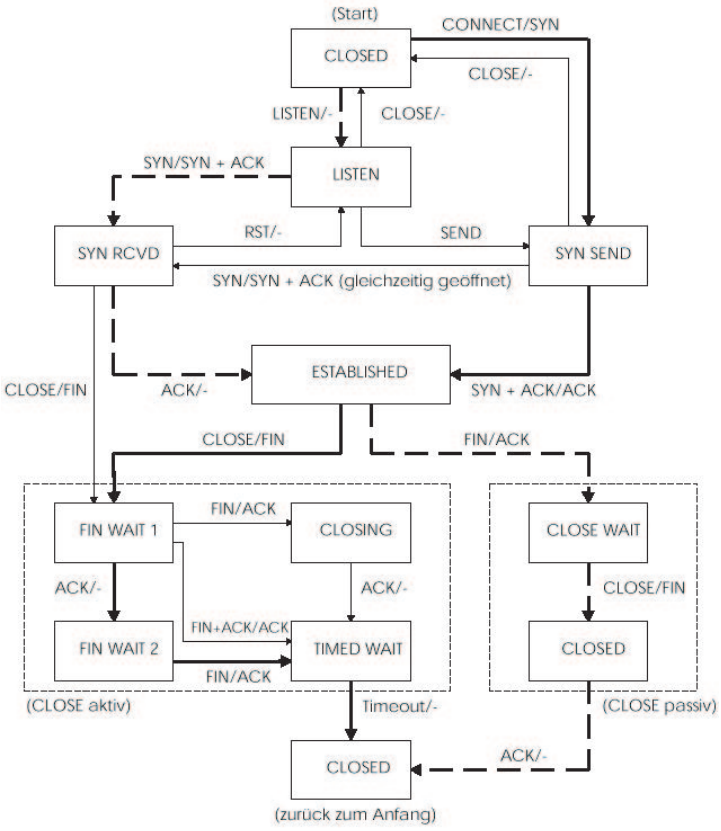

Abbildung 13: Zustandsautomat für TCP

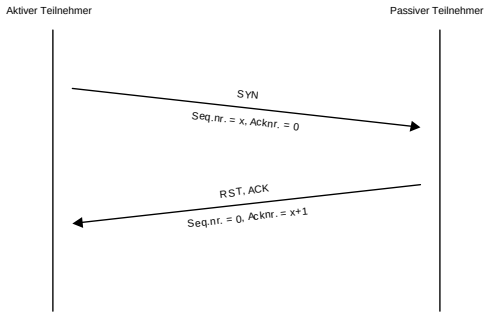

Abbildung 14: Ablehnung einer TCP-Verbindung

rechenbar sein soll und ein erneutes Versenden des Paketes wahrscheinlich weniger Zeit bei einer Verbindung beanspruchen wird, als die zusätzliche Rechenzeit für eine fehlerkorrigierende Lösung. Die Checksumme wird aus dem 16 Bit Einerkomplement der Summe aller Einerkomplemente der 16 Bit Worte aus dem Header und den Nutzdaten eines Pakets gebildet. Wenn ein Paket nicht ein Vielfaches von 16 Bit lang ist, werden die nötigen Bits mit Nullen aufgefüllt. Wenn sie nicht mit der beim Empfänger neu berechneten Checksumme übereinstimmt, wird das Paket verworfen und nach Ablauf des in Kapitel 9.3 genannten Timers neu gesendet.

10.2 Ablehnung von Kommunikation

Anhand von zwei Beispielen soll die Ablehnung von Kommunikation gezeigt werden:

10.2.1 TCP

Der Aufbauwunsch einer Station mit einem SYN-Paket wird von der Gegenstelle mit einem RST,ACK-Paket beantwortet und somit abgebrochen.

10.2.2 UDP

Da UDP ein verbindungsloses Protokoll ist, können gleich zu Beginn Daten gesendet werden. Das erste eintreffende Paket wird vom ablehnenden Empfänger mit einem ICMP Paket beantwortet (Destination Unreachable (Type 3) - Port Unreachable(Code 3)).

10.3 Trennung von Verbindungen

Wenn eine Verbindung unerwartet getrennt wird, zum Beispiel durch das Entfernen eines Netzwerkkabels oder das Abstürzen eines beteiligten Sys-

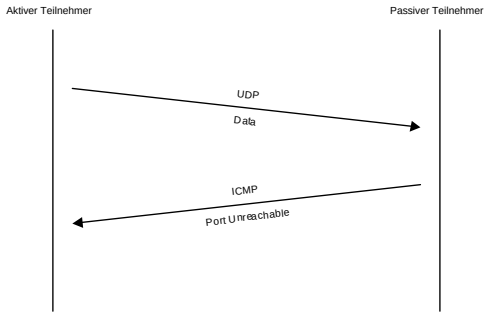

Abbildung 15: Ablehnung einer UDP-Verbindung

tems, wird die Verbindung nach Ablauf des in 9.3 genannten Timeouts geschlossen.

11 IP Routing Protokolle

Wie in der Einleitung gesagt wird Routing nötig, damit die Pakete ihren Weg durch das Internet zum Zielsystem finden können. Diese Entscheidungen werden auf der dritten Ebene des OSI Modells getroffen. Routing-Protokolle dienen dem Austausch von Routing-Informationen zwischen einzelnen Routern. Sie sind nötig, da nicht jeder Router alle Informationen über die Wege zu anderen Systemen haben kann. Routing-Protokolle lassen sich nach zwei Kriterien unterscheiden:

11.1 Intern oder Extern

Wenn eine Route ausgewählt werden soll, muss zunächst bestimmt werden, ob das Ziel innerhalb oder ausserhalb des eigenen autonomen Systems liegt. Ein autonomes System ist ein gemeinsam administriertes Netz mit mehr als einem Zugang zum Internet. Anhand dieser Information wird dann eine Klassifikation in IRP[6] und ERP[7] vorgenommen.

11.2 Kosten oder Entfernung

Bei der Festlegung, welche Route gewählt werden soll, kann entweder die kürzeste Entfernung, also möglichst wenig Hops[8], nach dem *Distance Vector Algorithmus*[9] oder der preiswerteste Weg nach dem *Link State Algorithmus*[10] berechnet werden.

[6] Interior Routing Protocols
[7] Exterior Routing Protocols
[8] Zwischenstationen zwischen dem Anfangs- und dem Zielsystem
[9] mit dem Bellman-Ford-Algorithmus
[10] mit dem Dijkstra-Algorithmus

	IRP	ERP
Distance Vector Algorithmus	RIP	BGP4
Link State Algorithmus	OSPF	–

Es kann kein ERP mit einem Link State Algorithmus geben, da hierzu die Verbindungskosten fremder autonomer Systeme bekannt sein müssten.

11.3 Übersicht über gängige IP Routing Protokolle

Im folgenden Abschnitt werden die heutzutage gängigen IP Routing Protokolle kurz vorgestellt.

11.3.1 RIP

Das beim XEROX PARC entwickelte *Routing Information Protocol* wird gerne in kleineren Netzen benutzt, welche maximal 15 Hops besitzen. Ein Hop ist ein Knoten innerhalb des Netzes, über den das Paket geleitet wird. Alle 30 Sekunden werden die Routing-Tabellen mit der Anzahl der Hops zu anderen Netzen von sich selber aus an alle Nachbarn im LAN versendet. RIP gewichtet die Routen ausschliesslich anhand der benötigten Hops, so dass es vorkommen kann, dass eine langsame Route mit 3 Hops gegen eine schnellere Route mit 5 Hops verliert.

11.3.2 OSPF

Bei dem vom IETF entwickelten *Open Shortest Path First* Protokoll baut jeder Router für sich selber einen gerichteten bewerteten Graphen auf, an dem sich nach Dijkstra der kürzeste (also der preiswerteste) Pfad zum Ziel berechnen lässt. Dies kann nur innerhalb eines autonomen Netzes funktionieren, aus dem in 11.2 genannten Grund. Im Gegensatz zu RIP ist OSPF wesentlich leistungsfähiger aber auch aufwändiger, was den Einsatz erst bei grösseren Netzen rechtfertigt.

11.3.3 BGP4

Das *Border Gateway Protokol Version 4* bildet einen zyklenfreien Graphen aller autonomen Systeme im Internet, indem sich benachbarte Router untereinander austauschen und sich so gegenseitig immer auf dem neuesten Stand halten. Über diesen Graphen wird dann der optimale[11] Weg für ein Paket berechnet.
BGP4 hat sich in den vergangenen Jahren zu dem Standardprotokoll für externes Routing etabliert.

[11] Also der Weg, der die wenigsten Hops beinhaltet.

12 IPv6

Als Anfang der 80er Jahre IPv4 eingeführt wurde, konnte niemand vorhersehen, dass das Internet so stark wachsen würde, so dass nach circa 20 Jahren ernsthafte Adressenknappheit herrschen würde. Im Juli 1994 wurde der Nachfolger von IPv4 bekanntgegeben (IPv6[12]), der folgende Neuerungen bringt:

- grösserer Adressraum (128 Bit)
- einfacherer, erweiterbarer Header
- optionale IPsec Verschlüsselung
- bessere Routing-Effizienz durch Flowlabel

Die Umstellung auf die Version 6 des Internetprotokolls wird noch längere Zeit dauern, da zum einen die Provider und Backboneanbieter neue Router kaufen oder zumindest die alten upgraden müssen und zum anderen 95% der momentan benutzten Software und ein verbreitetes Betriebssystem dieses Protokoll noch nicht unterstützt.

13 Schwachstellen

Es lassen sich bei näherem Betrachten der Internet-Protokolle folgende Sicherheitsmängel feststellen, die größtenteils historisch bedingt sind. Das Internet wurde schliesslich erfunden, um einen zuverlässigen (also stabilen) Datenaustausch zu gewähren und nicht um sichere (also gegen Angriffe geschützte) Kommunikation bereitzustellen.

13.1 Übertragung im Klartext

Für gewöhnlich erfolgt die Kommunikation über Internet-Protokolle wie TCP oder UDP unverschlüsselt. Das bedeutet, dass jeder, der in der Lage ist, diese Pakete mitzulesen, den Inhalt entziffern kann. Diese Technik nennt sich *sniffing*. Ein Beispiel für ein solches Sicherheitsproblem ist die Fernwartung eines Servers über Telnet. Angebrachter ist es, diese Fernwartung über das verschlüsselte SSH, die Secure Shell, zu betreiben.

13.2 Manipulation der Adressen

IPv4 überprüft nicht die Absender- oder Zieladresse eines Paketes, so dass man leicht Nachrichten mit fremdem Absender senden kann, was man als

[12] Die Versionsnummer 5 des Internetprotokolls war schon einem Multimediaprotokoll namens "Stream Transport Protokoll" zugesichert worden.

spoofing bezeichnet.
Es ist auch möglich, Pakete so umzuschreiben, dass sich die Zieladresse ändert. Man kann also Verbindungen umlenken, ohne dass der angegriffene Rechner etwas davon bemerkt. Wenn man beide eben genannten Methoden benutzt um ein System anzugreifen, wird dies eine *man-in-the-middle* Attacke genannt. Man stellt sich symbolisch zwischen zwei Systeme und liest deren Kommunikation mit oder verändert sie.
Die so abgefangene Verbindung kann weiter verändert werden, indem die Verbindung übernommen wird, was *hijacking* genannt wird.

13.3 Fragmentierung von Paketen

Wie in 5.3 erwähnt, können IP-Pakete fragmentiert werden, wenn das Paket zu gross für die MTU ist. Manche Firewalls haben Probleme mit extrem fragmentierten oder überlappenden Fragmenten, so dass sie umgangen werden können. Einige Betriebssysteme zeigten in der Vergangenheit instabiles Verhalten bei solchen nicht vom Programmierer vorgesehenen Paketen. Gegen solche Schwachstellen hilft nur sorgsame Programmierung.

13.4 Hängender TCP Verbindungsaufbau

Der Ablauf des Verbindungsaufbaus ist in 9.1 schon beschrieben worden. Wenn man einem System nur das SYN-Paket schickt und das SYN-ACK Paket ignoriert, halten alte System diese halb-offene Verbindung aufrecht und blockieren damit Ressourcen. Bei hinreichend vielen SYN-Paketen des Angreifers kommt dies einem Denial of Service Angriff gleich. Diese Attacke nennt sich *synflooding*.

13.5 Routing Probleme

Routing-Protokolle haben neben den oben genannten Schwachstellen noch spezielle Probleme wie Loops und Konvergenzschwierigkeiten. Loops sind Schleifen in Netzen, in denen die Pakete hin und hergeroutet werden, bis das TTL Feld im Header auf 0 gesunken ist. Konvergenzschwierigkeiten bezeichnet das Problem, dass eine gewisse Zeit, nachdem eine Route ungültig wurde (z.B. durch einen ausgefallenen Router), die Route noch immer in einigen Routern als gültig markiert ist.

14 Schlusswort

Es wird sich zeigen, wie schnell sich IPv6 gegen IPv4 durchsetzen wird und ob sich in den nächsten Jahren ganz andere Schwachstellen in IPv6 zeigen werden, mit denen heute noch niemand rechnet. Ein Vorteil ist, dass man bei der kommenden Version des Internet-Protokolls den Sicherheitsaspekt in

die Planung miteinbezogen hat, was bei IPv4 nicht der Fall war. Dies bringt zumindest einen gewissen Grad an Sicherheit.

Literatur

[1] *RFC 768: User Datagram Protocol*
`http://www.ietf.org/rfc/rfc0768.txt`

[2] *RFC 791: Internet Protocol Version 4*
`http://www.ietf.org/rfc/rfc0791.txt`

[3] *RFC 792: Internet Control Message Protocol*
`http://www.ietf.org/rfc/rfc0792.txt`

[4] *RFC 793: Transmission Control Protocol*
`http://www.ietf.org/rfc/rfc0793.txt`

[5] *RFC 1058: Routing Information Protocol*
`http://www.ietf.org/rfc/rfc1058.txt`

[6] *RFC 1519: Classless Inter-Domain Routing*
`http://www.ietf.org/rfc/rfc1519.txt`

[7] *RFC 1538: OSPF Version 2*
`http://www.ietf.org/rfc/rfc1538.txt`

[8] *RFC 1771: A Border Gateway Protocol 4 (BGP-4)*
`http://www.ietf.org/rfc/rfc1771.txt`

[9] *RFC 2018: Transmission Control Protocol Selective ACK*
`http://www.ietf.org/rfc/rfc2018.txt`

[10] *RFC 2460: Internet Protocol Version 6*
`http://www.ietf.org/rfc/rfc2460.txt`

[11] Craig Hunt. *TCP/IP Network Administration 2nd Edition* O'Reilly 1997.

[12] Anatol Badach und Erwin Hoffmann. *Technik der IP-Netze* Hanser 2001.

[13] Dr. B. Hackler. *Skript zur Vorlesung "Network Engineering"* an der TU Dresden. `http://www.rn.inf.tu-dresden.de/scripts_lsrn/lehre/net/print/net-kap73.ps`