HANSER

Leseprobe

Anatol Badach, Erwin Hoffmann

Technik der IP-Netze

Internet-Kommunikation in Theorie und Einsatz

ISBN (Buch): 978-3-446-43976-4

ISBN (E-Book): 978-3-446-43986-3

Weitere Informationen oder Bestellungen unter

http://www.hanser-fachbuch.de/978-3-446-43976-4

sowie im Buchhandel.

Inhaltsverzeichnis

Vorwort

Begriff: IP-Netze

Das Internet ist inzwischen zum unabdingbaren Kommunikationsmedium geworden, über das jeder zu jeder Zeit Information über fast alles abrufen sowie Nachrichten senden und empfangen kann. Unsere heutige Gesellschaft kann man sich ohne Internet kaum noch vorstellen. Voraussetzung zur Kommunikation zwischen Rechnern sind bestimmte Regeln, die vor allem die Datenformate und die Prinzipien der Datenübermittlung festlegen. Diese Regeln werden als Kommunikationsprotokolle bezeichnet. TCP/IP (*Transmission Control Protocol / Internet Protocol*) stellt eine derartige Protokollfamilie dar, sie wird im weltweiten Internet, in privaten Intranets und in anderen Netzen verwendet. Netze, die auf dieser Protokollfamilie aufbauen, bezeichnet man als *IP-Netze*.

Komplexität und Weiterentwicklung

Ein IP-Netz – und insbesondere das Internet – besteht nicht nur aus mehreren Rechnern und IP/TCP dazwischen, sondern dahinter verbergen sich sehr komplexe Vorgänge. Das Internet stellt einen weltweiten Dienst zur Übermittlung nicht nur von Daten, sondern auch von audiovisuellen Informationen, also von Audio und Video, in Form von IP-Paketen dar. Vergleicht man diesen Dienst mit dem Briefdienst der Post, so entspricht ein IP-Paket einem Brief und die sog. IP-Adresse einer postalischen Adresse. Das massive Wachstum des Internet und die dabei entstehenden Probleme und neuen Anforderungen haben die Entwicklung sowohl eines neuen Internetprotokolls, des IPv6, als auch von Techniken MPLS und GMPLS für die Übermittlung der IP-Pakete über Hochgeschwindigkeitsnetze, insbesondere über optische Netze, vorangetrieben. Noch in der ersten Dekade dieses Jahrhunderts hat man von *Next Generation IP Networks* gesprochen und sie sind bereits heute Realität geworden.

Ziel des Buches

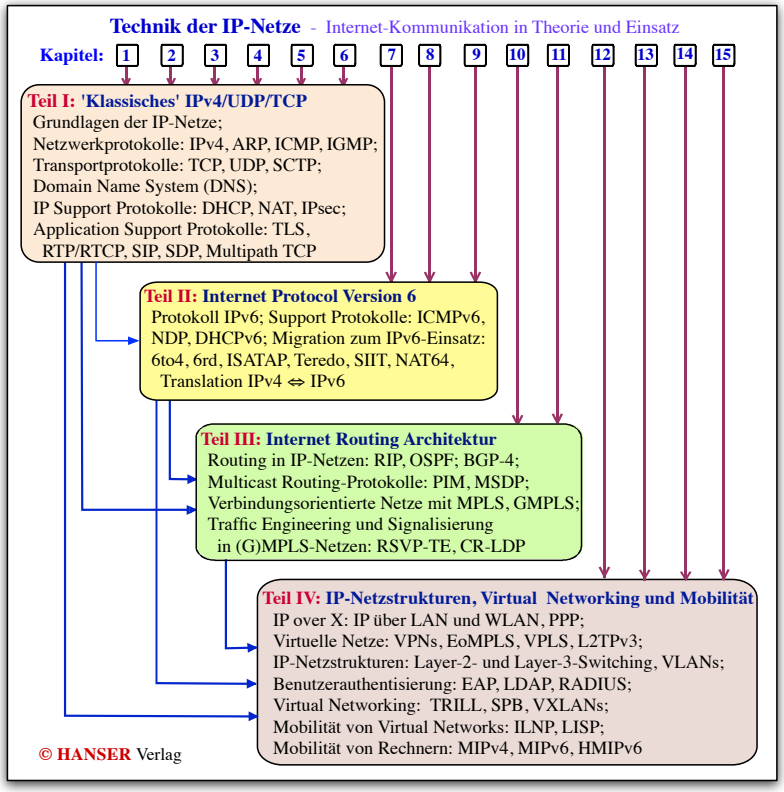
Dieses Buch gibt eine fundierte Darstellung zentraler Komponenten der TCP/IP-Protokollfamilie, wie z.B. IP, TCP, UDP, DNS und DHCP, sowie von Routing sowohl beim klassischen IP, IPv4 genannt, als auch beim neuen IPv6. Das Buch erläutert die Strategien für die Migration zum Einsatz von IPv6, präsentiert die Konzepte zum Aufbau der IP-Netze auf Basis verschiedener Netztechnologien, wie LANs, WLANs, SDH und WDM, und geht auch auf die IP-Weitverkehrsnetze mit (G)MPLS ein. Die Themen wie die Realisierung von VPNs, *Virtual Networking* in LANs durch die Bildung von VLANs und VXLANs, Konzepte und Einsatz von TRILL und *Shortest Path Bridging* werden ebenso präsentiert. Die Darstellung der Protokolle MIPv4, MIPv6 und HMIPv6 zur Unterstützung der Mobilität von Rechnern wie auch der Protokolle ILNP und LISP, mit denen man die Mobilität virtueller Netzwerke erreichen kann, rundet den Inhalt dieses Buches ab.

An wen richtet sich das Buch?

Das Buch ist so aufgebaut, dass sowohl die notwendigen technischen Grundlagen fundiert dargestellt als auch verschiedene Aspekte bei der Planung und Verwaltung der IP-Netze diskutiert werden. Damit eignet es sich nicht nur als Lehrbuch für Studenten und Neueinsteiger, sondern auch als Nachschlagewerk für alle Interessenten, die für die Planung, Realisierung, Verwaltung und Nutzung des Internet, von privaten Intranets und anderen IP-Netzen verantwortlich sind.

Aufbau des Buches

Zurzeit ist kein Buch verfügbar, in dem die Technik der IP-Netze so breit dargestellt wäre. Daher kann dieses Buch als ein Handbuch für alle Netzwerk-Verantwortlichen dienen. Durch die fundierte und praxisorientierte Darstellung der Inhalte eignet sich gut dieses



Buch auch für alle 'Internet-Fans' zum Selbststudium. Dieses Buch präsentiert in 15 Kapiteln, die auf vier Teile verteilt sind, alle wichtigen Aspekte der IP-Netze und kann nicht wie ein spannender Roman in einem Schlag durchgelesen werden. Das vorliegende Bild zeigt dessen logische Struktur und Abhängigkeiten zwischen Inhalten einzelner Teile, um den Lesern eine Orientierung zu geben, aus welchen Teilen man Kenntnisse benötig, um beim Lesen verschiedene, voneinander abhängige Themenbereiche besser zu verstehen.

Inhalte der Kapitel

Betrachtet man die einzelnen Kapitel dieses Buches etwas detaillierter, so lassen sie sich wie folgt kurz charakterisieren:

Kapitel 1

Kapitel 1 präsentiert die Entwicklung des Internet sowie die notwendigen Grundlagen der Rechnerkommunikation und der *Kommunikationsprotokolle* und geht u.a. daher auf die folgenden Probleme ein: Welche Funktionen liegen den Kommunikationsprotokollen zugrunde und wie kann die Kommunikation in IP-Netzen mittels eines Schichtenmodells anschaulich dargestellt werden? Wie können die verbindungslose und die verbindungsorientierte Kommunikation in IP-Netzen interpretiert werden und welche Bedeutung hat die Transportschicht? Welche Sicherheitsziele werden in IP-Netzen verfolgt und wie können diese technisch umgesetzt werden? Wie koordiniert die IETF die technologische Entwicklung des Internet und wie können wir diese verfolgen?

Kapitel 2 stellt sowohl IPv4 als auch dessen Hilfsprotokolle ARP und ICMP umfassend dar und erläutert u.a. folgende Fragestellungen: Wie sind IPv4-Pakete aufgebaut und welche Steuerungsangaben kann der Header eines IPv4-Pakets enthalten? Welche Arten von IPv4-Adressen gibt es und wie werden sie aufgebaut? Wie erfolgt die *Adressierung* in IP-Netzen und wie werden Subnetze gebildet? Welche Bedeutung haben die Protokolle ARP und ICMP und wie funktionieren sie? Wie realisiert man *Multicasting* in IP-Netzen mit dem Protokoll IGMP? Kapitel 2

Von großer Bedeutung in IP-Netzen ist die sog. *Transportschicht* mit den klassischen Protokollen TCP und UDP; hierzu kommen noch die neuen Protokolle SCTP und UDP-Lite. Weil das IP keine zuverlässige Übermittlung der Pakete garantiert, verwendet man hauptsächlich das TCP und in einigen Fällen das SCTP, um die zuverlässige Übermittlung der IP-Pakete zu gewährleisten. Kapitel 3 präsentiert die Aufgaben der Transportschicht und geht detailliert auf die folgenden Probleme ein: Welche Aufgaben haben die *Transportprotokolle* UDP, TCP und SCTP? Warum wurde UDP-Lite entwickelt und wann wird es eingesetzt? Wie werden die TCP-Pakete aufgebaut und welche Steuerungsangaben enthalten sie? Wie wird eine TCP-Verbindung auf- und abgebaut und wie verläuft die Flusskontrolle nach TCP? Was Neues bringt SCTP? Kapitel 3

Die Rechner in IP-Netzen werden zwar durch ihre IP-Adressen lokalisiert, aber es ist sinnvoll, statt einer IP-Adresse einen Rechner über seinen Namen anzusprechen – wie es auch unter Menschen üblich ist. Dies ist mit dem *Domain Name System* (DNS) möglich. Kapitel 4 liefert eine fundierte Darstellung von DNS, geht auf verschiedene Möglichkeiten des DNS-Einsatzes ein und erörtert u.a. die folgenden Probleme: Wie funktioniert DNS und welche Aufgaben kann DNS wahrnehmen? Wie erfolgt die Ermittlung der IP-Adresse aufgrund des Hostnamens und umgekehrt? Welche Informationen als sog. Resource Records enthält DNS und wie werden diese strukturiert? Welche Ziele werden mit ENUM, DynDNS und DNSSEC und CurveDNS verfolgt? Kapitel 4

Kapitel 5 stellt, als *IP-Support-Protokolle* bezeichnete, ergänzende Lösungen für das IPv4-Protokoll dar und erläutert hierbei u.a. die folgenden Aspekte: Wie können sich Rechner mittels DHCP automatisch eine gültige IPv4-Adresse zuweisen? Welche Lösungen für die Nutzung von privaten IPv4-Adressen mithilfe von NAT (*Network Address Translation*) gibt es und welche Probleme entstehen dabei – insbesondere bei der audiovisuellen Kommunikation? Wie kann *IPsec* zum verschlüsselten und authentisierten Austausch von IP-Paketen genutzt werden? Welche Probleme verursacht NAT bei IPsec und wie können diese bewältigt werden? Kapitel 5

Mehrere ergänzende Lösungen sind nicht nur für das IPv4 nötig, sondern in Form spezieller Protokolle auch für Applikationen, sodass wir von *Application Support Protokollen* sprechen. Kapitel 6 präsentiert diese, erläutert deren Aufgaben und geht u.a. auf folgende Fragestellungen ein: Wie kann der Datentransport zwischen Applikationen mittels TLS gesichert realisiert werden und welche Voraussetzungen sind hierfür nötig? Welche Protokolle zur Realisierung der Echtzeitkommunikation in IP-Netzen benötigt werden, welche Aufgaben haben sie und wie werden sie konzipiert? Wie funktioniert *Multipath TCP* und wie kann es eingesetzt werden, u.a. wie lassen sich parallele Datenpfade über mehrere Internetanbindungen für eine TCP-Verbindung nutzen? Kapitel 6

Um den steigenden Anforderungen an IP-Netze gerecht zu werden, wurde das IPv6 als '*IP der nächsten Generation*' entwickelt und die Ära von IPv6 hat bereits begonnen. IPv6 bringt neue Möglichkeiten und diese reichen von Sicherheitsfunktionen über mehr Flexibilität bis hin zur Unterstützung von neuartigen Anwendungen. Das IPv6 ermög- Kapitel 7

licht die automatische Konfiguration von Rechnern, sodass man sogar von *Plug&Play-Konfiguration* spricht. Kapitel 7 stellt das IPv6 ausführlich dar und geht u.a. auf die folgenden Probleme ein: Welche Ziele wurden bei der Entwicklung von IPv6 verfolgt? Welche neuen Funktionen bringt IPv6 mit sich? Welche Arten von IPv6-Adressen gibt es und wie können sie den Rechnern zugewiesen werden?

Kapitel 8 Ein wichtiges Ziel bei der Entwicklung von IPv6 war die Unterstützung der automatischen Konfiguration von Rechnern. Hierfür stehen die Protokolle ICMPv6, NDP und DHCPv6 zur Verfügung. Diese *IPv6 Support Protokolle* stellt Kapitel 8 dar und geht hierbei u.a. auf folgende Aspekte ein: Wie wurde ICMPv6 konzipiert und welche Aufgaben hat es? Welche Funktionen liefert NDP, um die automatische Konfiguration von Rechnern mit IPv6 zu unterstützen? Wie bekommt ein IPv6-Rechner seine Netzkonfiguration automatisch zugewiesen?.

Kapitel 9 Da die Umstellung von allen Rechnern, in denen das klassische IPv4 verwendet wird, auf das IPv6 nicht auf einen Schlag geschehen kann, benötigt man geeignete Systemlösungen für die Migration zum IPv6-Einsatz. Kapitel 9 präsentiert verschiedene Ansätze und Systemlösungen für die Koexistenz von IPv4 und IPv6 – vor allem die Konzepte *IPv6 over IPv4* und *IPv4 over IPv6*. Die Integration der IPv4- und der IPv6-Netze dank der Translation *IPv4* $\Leftrightarrow$ *IPv6* wird ebenso präsentiert. Hier werden u.a. folgende Probleme erörtert: Wie kann man sich die Koexistenz von IPv4 und IPv6 in einem Rechner vorstellen, wann ist diese Koexistenz möglich und welche Bedeutung hat sie? Wie kann die IPv6-Kommunikation über IPv4-Netze erfolgen? Wie lassen sich IPv6-Netzsegmente über IPv4-Netze verbinden? Wie können die Rechner aus IPv6-Netzen auf das IPv4-Internet zugreifen? Wie erfolgt die Translation IPv4 $\Leftrightarrow$ IPv6 und was ermöglicht sie?

Kapitel 10 Router fungieren in IP-Netzen als Knoten, ermitteln optimale Übermittlungswege, die sog. *Routen*, für die empfangenen IP-Pakete und leiten sie weiter. Kapitel 10 vermittelt eine kompakte Darstellung von Routing-Grundlagen und -Protokollen. Es werden hier die *Routing-Protokolle* RIP-1, RIP-2 und OSPF sowie BGP-4 erläutert. Dieses Kapitel zeigt auch, wie eine redundante Router-Auslegung mithilfe der Protokolle HSRP und VRRP erfolgen kann, und stellt die Protokolle PIM-SM und MSDP für das *Multicast-Routing* dar. Hierbei werden u.a. folgende Fragen beantwortet: Welche Aufgabe haben die Router und wie funktionieren sie? Welche Prinzipien liegen den Routing-Protokollen zugrunde? Wie verlaufen die Routing-Protokolle RIP und OSPF? Welche Erweiterungen dieser Protokolle sind für IPv6 notwendig? Wie funktioniert BGP-4, für welche Zwecke und wie kann es eingesetzt werden? Wie können die Router am Internetzugang redundant ausgelegt werden? Wie realisiert man Multicast-Routing in IP-Netzen?

Kapitel 11 Die IP-Netze im Weitverkehrsbereich basieren überwiegend auf dem MPLS-Konzept und auf der, als GMPLS (*Generalized MPLS*) bezeichneten, dessen Erweiterung. Die Techniken MPLS und GMPLS ermöglichen die Konvergenz von Ethernet u.a. mit SDH- und WDM-Netzen. Dank dieser Konvergenz können Ethernet heutzutage nicht nur als LAN eingerichtet werden, sondern *Ethernet-Services* können sogar weltweit verfügbar gemacht werden. Kapitel 11 stellt die Konzepte und Protokolle zum Aufbau der IP-Netze mit dem MPLS und dem GMPLS vor und geht u.a. auf die folgenden Probleme ein: Worin bestehen die Konzepte MPLS und GMPLS und welche Möglichkeiten entstehen durch deren Einsatz? Welche Services werden durch *Traffic Engineering* in IP-Netzen erbracht? Wie werden (G)MPLS-Netze aufgebaut und wie wird die IP-Kommunikation über sie realisiert? Wie erfolgt die IP-Kommunikation über optische Netze? Wie können Datenpfade über (G)MPLS-Netze dynamisch eingerichtet werden?

Kapitel 12

In vielen Unternehmen können gleichzeitig unterschiedliche Netztechnologien eingesetzt und entsprechend integriert werden. Sie lassen sich mithilfe von *Tunneling-Techniken* so einsetzen, dass virtuelle Standleitungen für den Transport von Daten über öffentliche IP-Netze aufgebaut werden können. Diese Idee hat zur Entstehung von VPNs geführt. Somit erläutert Kapitel 12 einerseits die Konzepte für den IP-Einsatz in Netzen mit klassischen LANs (Ethernet), Punkt-zu-Punkt-Verbindungen (z.B. physikalischen Standleitungen, Satellitenverbindungen) und WLANs. Andererseits präsentiert dieses Kapitel auch die Lösungen und Protokolle für den Aufbau von VPNs auf Basis sowohl klassischer IP-Netze mittels des IPsec als auch der IP-Netze mit den Techniken MPLS bzw. GMPLS, die auch als *Provider Provisioned VPNs* bezeichnet werden. Hierbei geht dieses Kapitel u.a. auf folgende Fragestellungen ein: Wie kann man sich ein logisches LAN-Modell vorstellen und wie kann die *Multiprotokollfähigkeit in LANs* erreicht werden? Welche Ideen liegen den WLANs nach IEEE 802.11 zugrunde und wie werden WLANs mit einem Ethernet gekoppelt? Welche Typen von virtuellen Netzen gibt es, wie werden sie aufgebaut und wie können sie genutzt werden? Wie lassen sich sichere, virtuelle IP-Netze aufbauen?

Kapitel 13

In den letzten Jahren haben sich einige Megatrends auf der 'Netzwerkwelt' herauskristallisiert. In privaten Netzwerken spricht man heute von *Layer-3-Switching* und von VLANs (*Virtual LANs*). Dabei stellt die Virtualisierung von Rechnern neue Anforderungen an IP-Netze. Die Unterstützung der Mobilität virtueller Rechner und virtueller Netzwerke sowie der Wunsch nach flexibler Möglichkeit, einen Rechner bzw. ein Netzwerk an das Internet parallel anbinden zu können, sind nur die wichtigsten von ihnen. Diese Probleme erläutert Kapitel 13 und präsentiert neue Konzepte und Protokolle hierfür, um diesen Anforderungen gerecht zu werden. Hervorgehoben sei hier *Shortest Path Bridging* (SPB), TRILL, VXLANs, ILNP und LISP. Dieses Kapitel geht u.a. auf folgende Fragen ein: Wie werden moderne IP-Netzwerke physikalisch und logisch strukturiert? Wie funktionieren Layer-2- und Layer-3-Switches, wo und wie werden sie eingesetzt? Wie können komplexe, auch virtuelle Rechner enthaltene VLANs gebildet werden und welche Bedeutung dabei hat VLAN Tagging? Worin bestehen die Ideen von TRILL, SPB, VXLAN, ILNP und LISP? Welche Möglichkeiten der Integration von IPv4 und IPv6 liefert LISP?

Kapitel 14

Ein großes Problem in Kommunikationsnetzen ist die Feststellung, ob der jeweilige Kommunikationspartner der 'richtige' ist und ob ein Benutzer auf bestimmte Netzressourcen zugreifen darf. Somit sind hierfür entsprechende Verfahren zur Benutzerauthentisierung notwendig. Kapitel 14 präsentiert, welche Möglichkeiten in Frage kommen, auf welchen Prinzipien sie basieren und geht dabei auf die vorrangigen Probleme ein: Worin besteht das Konzept des *Authentisierungsprotokolls EAP* und warum dieses ist in Netzwerken unabdingbar? Wie funktioniert das Protokoll RADIUS und welche Bedeutung hat es in IP-Netzen? Wozu und wie werden die Verzeichnisdienste eingerichtet und welche Funktionen realisiert dabei das Protokoll LDAP?

Kapitel 15

Um die Mobilität in IP-Netzen zu ermöglichen, wurden die Protokolle MIP (*Mobile IP*), MIPv6 (*Mobile IPv6*) und HMIPv6 (*Hierarchical MIPv6*) entwickelt. Kapitel 15 zeigt, wie diese Protokolle funktionieren und was gemacht werden muss, damit ein mobiler Rechner während bestehender Verbindungen ein Subnetz verlassen und in ein neues hinein bewegen kann, ohne die bestehenden Verbindungen abbrechen zu müssen. Auch die Integration von Hotspots mit dem Internet und die Möglichkeiten von Roaming zwischen Hotspots werden präsentiert. Darüber hinaus werden u.a. die folgenden Aspekte erörtert: Welche Ansätze und Protokolle zur Unterstützung der Mobilität in IP-Netzen gibt es? Wie kann *Roaming* zwischen Hotspots realisiert werden? Wie verläuft die Kommunikation beim Einsatz von MIP bzw. von MIPv6?

Vorwort zur dritten Auflage

Im Jahr 1997 haben Prof. Badach und ich entschieden, unser Buch 'High Speed Internetworking' in mehrere Bände aufzuspalten und ein Band dem Protokoll IP zu widmen und zugleich einen neuen Verlag zu suchen. Es war damals bereits klar, dass die Protokollfamilie 'TCP/IP' zu einem Erfolgsfall werden wird, der die Kommunikation des kommenden 21. Jahrhunderts bestimmen würde.

Auch war bereits damals absehbar, dass IPv4 mit seinem 32 Bit Adressraum nicht ausreichend sein wird, den wachsenden Kommunikationsbedarf zu bedienen. In der zweiten Auflage haben wir dann den Versuch gewagt, das IPv6-Protokoll an zentraler Stelle des Buches zu positionieren und zugleich statt einer (aussichtslosen) umfangreichen Beschreibung der TCP/IP-Applikationsprotokolle uns auf die Kernbestandteile der IP-Kommunikation zu konzentrieren.

Diesen Weg gehen wir nun mit der dritten Auflage von 'Technik der IP-Netze' weiter. Der geneigte Leser wird beim Vergleich beider Auflagen bemerken, dass Sicherheitsaspekte der IP-Nutzung nun wesentlich in den Vordergrund gerückt sind. Dies ist keine überraschende Entwicklung, stand die Neubearbeitung des Buches in der zweiten Hälfte des Jahres 2013 doch unter starkem Einfluss der NSA-Überwachungsaffäre.

Mit der dritten Auflage des Buches haben wir sowohl aktuelle Themen aufgenommen (IP-Security, Mobility in IP-Netzen, Protokolle für die audiovisuelle Echtzeitkommunikation, Benutzerauthentisierung, neue Entwicklungen wie MPTCP, LISP und ILNP), als auch den Stoff 'reifen' lassen: Nahezu 15 Jahre diente das Buch als Vorlage für unsere Vorlesungen sowohl an der Hochschule Fulda, als auch an der Fachhochschule Frankfurt/Main.

Die heutigen Leser von 'Technik der IP-Netze' sind 'Digital Natives'; also im täglichen Umgang mit dem 'Netz' (Internet) vertraut. Dies unterscheidet in Retrospektive auch die erste von dieser nun vorliegenden dritten Auflage (nach 14 Jahren) fundamental. Haben wir in der ersten Auflage noch eine CD mit den Internet RFCs beigelegt, können nun unsere Quellen quasi Online mittels Hyperlinks verfügbar gemacht werden.

Wir haben daher zur Realisierung der dritten Auflage einen Medienbruch vorgenommen:

> War das Buch bislang in 'Word' erstellt worden, wurde die dritte Auflage in *LaTeX* realisiert. Bei allem Respekt vor Donald Knuth's *TeX* wird der aufmerksame Leser jedoch hin und wieder an die Grenzen und Eigenheiten des Systems stoßen, die wir nicht immer zufriedenstellend 'umschiffen' konnten.

Technik der IP-Netze – Homepage

Die Homepage des Buches ist unter `http://www.fehcom.de/tipn` erreichbar, wo sich auch Korrekturen einfinden werden.

Ihre Kritik, Verbesserungsvorschläge und eventuell Ihre Korrekturen sind willkommen und wir nehmen sie gerne entgegen. Für Lehr- und Ausbildungszwecke stellen wir die Abbildungen gerne zur Verfügung.

Die Autoren

Prof. Dr.-Ing. Anatol Badach

über 30 Jahre war auf den Gebieten Informatik und Telekommunikation beruflich tätig; Promotion (1975), Habilitation (1983). Von Dezember 1985 bis August 2012 war er Professor im Fachbereich Angewandte Informatik an der Hochschule Fulda. Seine Schwerpunkte in Lehre und Forschung waren: Rechnerkommunikation, Netzwerktechnologien und Multiservice Networking. Er hat u.a. auf den Gebieten: Netzwerktechnologien und Protokolle, VoIP und Next Generation Networking geforscht und verfolgt mit Engagement einige wichtige Entwicklungen weiter.
Prof. Badach ist Autor zahlreicher Veröffentlichungen und u.a. zahlreicher anderer Fachbücher, darunter *Voice over IP – Die Technik*, *Netzwerkprojekte* (Mitautor), *Web-Technologien* (Mitautor), *Integrierte Unternehmensnetze*, *Datenkommunikation mit ISDN*, *High Speed Internetworking* (Mitautor), *ISDN im Einsatz*. Seine Erfahrung vermittelt er weiter als Leiter/Referent bei Fachkongressen und -seminaren, Berater bei innovativen Projekten und Entwicklungen, Autor von Fachbeiträgen.

`http://www.competence-site.de/Anatol-Badach`

Prof. Dr. Erwin Hoffmann

Jahrgang 1958, Studium der Physik und Astrophysik an der Universität Bonn und 1989 Promotion an der TU München (Max-Planck-Institut für Physik und Astrophysik). Durch seine Tätigkeit in der experimentellen Teilchenphysik am CERN und Fermilab verschaffte er sich Kenntnisse über unterschiedlichste Rechnerbetriebssysteme. Beruflich war er zunächst im Bereich Hochgeschwindigkeitsnetze (FDDI) engagiert sowie mit der Implementierung von TCP/IP auf IBM-Großrechnern.
Ab 1994 war Prof. Hoffmann als Netzwerk- und Systemberater mit den Schwerpunkten Unix, IT-Prozessmanagement und ITIL tätig und trägt zur Weiterentwicklung der Software von D.J. Bernstein bei. Zwischen 2007 und 2013 Vertretungsprofessor an der Fachhochschule Frankfurt am Main für Rechnernetze, Betriebssysteme, IT-Security und IT-Projektmanagement. Seit 2014 ist er Professor an der Provadis Hochschule in Frankfurt/Höchst mit den Schwerpunkten IT-Governance und IT-Security.

`http://www.fehcom.de`

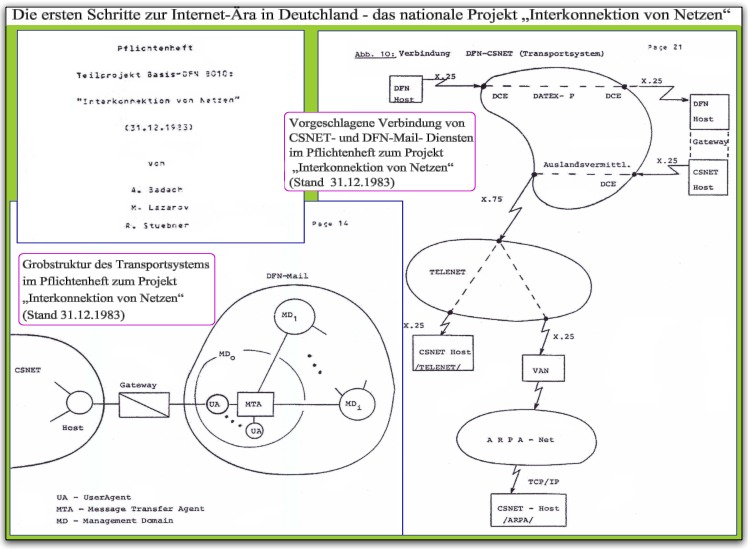
Erste Schritte zur Internet-Ära in Deutschland

Nationales Projekt B010 'Interkonnektion der Netze'

Die dritte Auflage des Buches 'Technik der IP-Netze' vermittelt die Prinzipien der *Internet-Kommunikation in Theorie und Einsatz* und erschien fast zeitgleich zum 30. Jahrestag der ersten Internet-E-Mail, die an der Universität Karlsruhe empfangen wurde, also de facto zum 30. Jahrestag der Internet-Ära im Deutschland.

In diesem Zusammenhang möchten wir auf die aus diesem Anlass von Prof. Dr.-Ing. Werner Zorn verfasste Festschrift verweisen. Prof. Zorn war Initiator und Koordinator

des nationalen Projekts 'Interkonnektion der Netze', mit dem zunächst die Universität Karlsruhe mit dem CSNET (*Computer Science Network*) verbunden wurde, und das der Grundstein des X-WiN beim DFN-Verein (*Deutsches Forschungsnetz*) in Deutschland werden sollte. Prof. Anatol Badach war auch an diesem Projekt beteiligt. Die hier gezeigten, aus der erwähnten Festschrift stammenden Abbildungen illustrieren die damaligen Ideen, die dazu beigetragen haben, dass die erste Internet-E-Mail nach Deutschland kam – und somit die Internet-Ära in Deutschland begann[1].



Danksagung

Ein so umfangreiches Buch kann ohne Anregungen von außen und einen entsprechenden Erfahrungsaustausch nicht geschrieben werden. An dieser Stelle danken wir deshalb allen Firmen und Personen, die uns mit ihren Anregungen unterstützt haben.

Ein besonderer Dank gilt Herrn Dipl. Inf. Benedikt Stockebrand, der mehrere, IPv6-betreffende Vorschläge geliefert hat. Für zeitraubendes Korrekturlesen möchten wir uns besonders beim Herrn Jürgen Dubau recht herzlich bedanken. Seine Bereitschaft und erbrachte Leistung war für uns eine große Hilfe. Für die gute und angenehme Zusammenarbeit mit dem Hanser Verlag möchten wir uns insbesondere bei Frau Margarete Metzger, Frau Brigitte Bauer-Schiewek und Frau Irene Weilhart recht herzlich bedanken. Nicht zuletzt danken wir unseren Familien für die unendliche Geduld, die sie uns während unserer Arbeit an den drei Auflagen dieses Buches entgegengebrachten.

Dieses Buch möchten wir all jenen widmen, die dank ihrer technischen Schöpfungen zur Entstehung des Internet beigetragen haben, und ebenso denen, die sich dafür engagieren das Internet weiterzuentwickeln, es offen und aufrecht zu erhalten.

Prof. Anatol Badach (Fulda)
Prof. Erwin Hoffmann (Höhn) – im Januar 2015

[1] `http://www.informatik.kit.edu/downloads/zu-30JahreInternet-EMail-V01-28Jul2014.pdf`.

1 Grundlagen der IP-Netze

Internet als IP-Netz

Die heutige Gesellschaft kann man sich ohne Internet kaum noch vorstellen. Das *Internet* ist ein weltweites Rechnernetz, in dem nicht nur die Daten, sondern auch alle digitalisierten Echtzeitmedien wie Sprache, Audio und Video mit dem *Internet Protocol* (IP) übermittelt werden. Das Internet und alle anderen Netze auf Grundlage von IP nennt man *IP-Netze*. Die Kommunikation zwischen zwei Rechnern über ein IP-Netz bedeutet aber nicht nur zwei Rechner und IP dazwischen, sondern dahinter verbergen sich sehr komplexe Kommunikationsregeln, die in Form von *Kommunikationsprotokollen* spezifiziert werden.

Protokollfamilie TCP/IP

In IP-Netzen bilden alle Kommunikationsprotokolle eine Protokollfamilie, die sogenannte Protokollfamilie TCP/IP. Diese Familie, die sich seit mehr als 40 Jahren entwickelt hat, enthält außer IP und TCP (*Transmission Control Protocol*) eine Vielzahl weiterer Protokolle. Um diese Protokolle systematisch erläutern zu können, ist ein anschauliches Modell sehr hilfreich. Es basiert auf dem *OSI-Referenzmodell* (*Open System Interconnection*), das bereits Ende der 70er Jahre eingeführt wurde.

Überblick über das Kapitel

Dieses Kapitel schildert in Abschnitt 1.1 kurz die bisherige und zukünftige Entwicklung des Internet und beschreibt in komprimierter Form die Hauptkomponenten des WWW (*World Wide Web*). Abschnitt 1.2 erläutert die grundlegenden Funktionen der *Kommunikationsprotokolle* und geht dabei insbesondere auf die Ideen der Fehlerkontrolle, Flusskontrolle und Überlastkontrolle. Dem Schichtenmodell für die Darstellung von Prinzipien der Rechnerkommunikation widmet sich Abschnitt 1.3. Allgemeine Prinzipien der Kommunikation in IP-Netzen erläutert Abschnitt 1.4. Die wichtigsten Komponenten der Protokollfamilie TCP/IP präsentiert kurz Abschnitt 1.5. Abschnitt 1.6 ist der Sicherung der zu übertragenden Datenpakete gewidmet und erläutert Grundlagen der Sicherheit in IP-Netzen. Schließlich geht Abschnitt 1.7 auf den Aufbau der Organisation IETF (*Internet Engineering Task Force*) und die Internet-Standards ein. Schlussbemerkungen in Abschnitt 1.8 runden dieses Kapitel ab.

Ziel dieses Kapitels

In diesem Kapitel werden u.a. folgende Fragen beantwortet:

- Wie sah die bisherige Entwicklung des Internet aus und welche aktuellen Trends gibt es?
- Welche Funktionen liegen den Kommunikationsprotokollen zugrunde und wie kann die Kommunikation in IP-Netzen mittels eines Schichtenmodells anschaulich dargestellt werden?
- Wie können die verbindungslose und die verbindungsorientierte Kommunikation in IP-Netzen interpretiert werden und welche Bedeutung hat die *Transportschicht* in IP-Netzen mit den Protokollen TCP, UDP und SCTP?
- Wie kann sichergestellt werden, dass die übertragenen Daten unverfälscht und ohne 'in fremde Hände zu gelangen' übertragen werden?
- Welche Sicherheitsziele werden bei der IP-Kommunikation verfolgt und wie können diese technisch umgesetzt werden?
- Wie koordiniert die IETF die technologische Internet-Weiterentwicklung und wie können wir diese verfolgen?

1.1 Entwicklung des Internet

Es begann in den 60er Jahren

Die ersten Spuren, die in indirekter Form zur Entstehung des Internet beigetragen haben, führen zurück in die 60er Jahre. In dieser Zeit wurde zum ersten Mal für die amerikanische Regierung eine Kommunikationsform für den Fall eines nuklearen Krieges erforscht. Die damaligen Überlegungen beinhalten bereits die noch heute geltenden Grundprinzipien der paketvermittelnden Kommunikation. Die Entwicklung des Internet lässt sich grob in folgende Phasen einteilen:

- Das Internet vor der Nutzung des WWW (*World Wide Web*): Aufbau- und Experimentierphase als ARPANET und Verbreitung des Internet vor allem als Forschungs- und Wissenschaftsnetz.
- Die *Schaffung des WWW*.
- Das *Internet nach der Etablierung des WWW* als weltweite Kommunikationsinfrastruktur für wissenschaftliche, private und kommerzielle Nutzung.

1.1.1 Internet vor der Nutzung des WWW

ARPANET als Vorläufer des Internet

Die Geschichte des Internet ist eng mit der Entstehung des ersten Rechnernetzes im Jahr 1969 verbunden. Die Entwicklung dieses Rechnernetzes wurde vom *US Defense Advanced Research Project Agency* (DARPA), einer *Organisation des Department of Defense* (DoD), initiiert und es trug den Namen ARPANET (*Advanced Research Project Agency Network*). Abb. 1.1-1 illustriert den Aufbau des ARPANET.

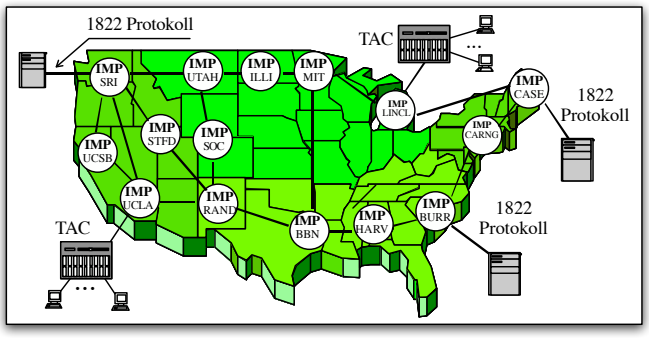


Abb. 1.1-1: Allgemeiner Aufbau von ARPANET – IMP dienen als Knoten
TAC: Terminal Access Controller, IMP: Internet Message Processor

Geburt von ARPANET

DARPA wollte zunächst digitale Telekommunikation auf Basis einer 'packet switching'-Methode über unterschiedliche Netze bereit stellen. Als erster Schritt hierzu wurde am 2. September 1969 am *University College of Los Angeles* (UCLA) ein Computer an einen *Internet Message Processor* (IMP) angeschlossen. Der IMP war auf der Basis eines Honeywell 516 Rechners der Firma *Bolt, Beranek & Newman* (BBN) gebaut worden.

70er Jahre

Anfang der 70er Jahre wurden die mittlerweile 15 zusammengeschalteten IMPs unter dem Namen *ARPANET* gehandelt. Das Kommunikationsprotokoll der IMPs trug die Bezeichnung BBN 1822 und kann als Vorläufer von IP gelten. Um ARPANET mit anderen Paketnetzen koppeln zu können, wurden 1974 ein Internetwork-Protokoll sowie Gateways entwickelt.

ICCB und NCP

Die weitere technische Entwicklung der zunächst NCP (*Network Control Program*) genannten Protokolle wurde vom DARPA entkoppelt und in die Obhut des *Internet Configuration Control Board* (ICCB) gegeben. Mit der 1983 von der *Defense Communication Agency* (DCA) vorgenommenen Trennung des militärisch genutzten Teils des Netzes *MILNET* vom ARPANET war ein weiterer wichtiger Schritt für die breite öffentliche Entwicklung des Internet gemacht.

BSD und Sockets

Diese Trennung hatte auch entscheidenden Einfluss auf das Betriebssystem UNIX, das von der Firma AT&T 1969/1970 entwickelt wurde. Wiederum am UCLA wurde in dieses Betriebssystem (genauer: unter UNIX System III) eine Netzwerk-Programmierschnittstelle *Sockets* implementiert, die es erlaubte, eine direkte Rechnerkommunikation mit dem ARPANET aufzunehmen. Dieses UNIX wurde als *Berkeley Software Distribution* (BSD) gegen eine geringe Gebühr abgegeben und fand daher schnellen Einzug in Lehre und Forschung. Die weitere Verbreitung von UNIX und Internet sowie ihre technische Fortführung waren die Folge. Nach der ersten Version BSD 4.0 folgte 4.2 und anschließend 4.3, wobei die spätere kommerzielle Weiterentwicklung durch die Firma Sun Microsystems als Betriebssystem Sun OS und später Solaris erfolgte.

IAB und TCP/IP

Eine 1983 stattfindende Reorganisierung des ICCB führte nicht nur zur Konstituierung des *Internet Activity Board* (IAB) anstelle des ICCB, sondern auch zur Festlegung der als Standard geltenden, nun TCP/IP genannten Protokollfamilie. Mit der weiteren Entwicklung wurde auch dieser organisatorische Rahmen zu eng. Das IAB wurde zum *Internet Architecture Board* umfirmiert und u.a. um folgende Gremien ergänzt:

- IETF *Internet Engineering Task Force* als offenes Gremium von Netzwerk-Architekten und -Designern, vor allem aus interessierten Firmen und Einzelpersonen gebildet, um die Entwicklung des Internet zu koordinieren `http://www.ietf.org`. Auf die Organisation der IETF geht Abschnitt 1.7 näher ein.
- IESG *Internet Engineering Steering Group* mit der Aufgabe, die Tagesaufgaben der IETF zu managen und eine erste technische Stellungnahme zu neuen Internet-Standards zu beziehen `http://www.ietf.org/iesg.html`.
- IRTF *Internet Research Task Force* als Gremium zur Grundlagendiskussion langfristiger Internet-Strategien und -Aufgaben `http://www.irtf.org`.
- IEPG *Internet Engineering and Planning Group*, eine offene Arbeitsgruppe von Internet- Systemadministratoren, die dem Ziel verpflichtet sind, einen koordinierten Internet-Betrieb zu gewährleisten `http://www.iepg.org`.
- ICANN *Internet Corporation for Assigned Names and Numbers* mit der Aufgabe, die Verwendung und die Konsistenz der im Internet benutzten Namen, Optionen, Codes und Typen zu regeln und zu koordinieren (`http://www.icann.org`).

NSFNet

Nach der Trennung des militärischen vom zivilen Teil des ARPANET wurde dieses zunächst zum Austausch wissenschaftlicher Informationen genutzt und von der *National Science Foundation* (NSF) betreut. Diese baute 1986 den zivilen Teil als nationales Backbone-Netz aus, das als NSFNet bekannt geworden ist. Drei Jahre später (1989) waren ca. 100 000 Rechner, die sich an Universitäten und Forschungslabors, in der US-Regierung und in Unternehmen befanden, am NSFNet angeschlossen. In nur einem Jahr (1990) hat sich die Anzahl der angeschlossenen Rechner verdoppelt, wobei das NSFNet ca. 3 000 lokale Netze umfasste. Dies war auch der Zeitpunkt, an dem das *Domain Name System* (DNS) eingeführt wurde.

EARN

Auch in Europa wurden die ersten Ansätze zur Vernetzung der Forschungsinstitute durch EARN (*European Academic Research Network*) durchgeführt, um die bislang nationalen Netze wie z.B. BitNet in England und das vom DFN-Verein (*Deutsches Forschungsnetz*) getragene WiN (*Wissenschafts-Netz*), miteinander zu koppeln.

USENET

Neben dem direkten, d.h. festgeschalteten und teuren Anschluss ans Internet, wie er bei Universitäten und Forschungseinrichtungen sowie auch bei Firmen üblich ist, wurde bald ein loser Verbund von Systemen – vor allem auf UNIX-Rechnern basierend – aufgebaut, die über Telefonleitungen und Modems gekoppelt waren: das USENET. Hier wurden die Rechner über das Protokoll UUCP (*UNIX to UNIX Copy*) miteinander verbunden und Nachrichten ausgetauscht. Hauptzweck des USENET war die Verbreitung von E-Mail sowie vor allem von *NetNews*, die in *Newsgroups* themenstrukturierte, virtuelle Nachrichtenbretter darstellen, in denen zunächst technische Fragen zu Rechnern, Programmiersprachen und dem Internet behandelt wurden. USENET war zeitweise so populär, dass es mit dem Internet selbst identifiziert wurde.

Cyberspace

Die 'kopernikanische Wende' des Internet vollzog sich mit der Schaffung des WWW [Abschnitt 1.1.2] durch Tim Berners-Lee. Damit wurde die Möglichkeit geschaffen, mittels eines einfachen 'Browsers' grafisch auf öffentlich verfügbare Internet-Ressourcen über Webserver zugreifen zu können.

'dot-com' Internet

Sehr schnell fand die 'kopernikanische Wende' Ergänzung in einer 'keplerschen Wende': Mit Realisierung einer allgemeinen nutzbaren Verschlüsselung des Datenverkehrs zunächst auf Grundlage des *SSLeay*-Protokolls, entwickelt durch die Firma Netscape Anfang der 90er Jahre, konnte nun das Internet auch für den kommerziellen Einsatz genutzt werden. Das Internet explodierte, was sowohl die Anzahl der Teilnehmer bzw. der Anwender, die Server und die Datenmenge betraf. Das Internet mutierte vom Wissenschaftsnetz zum multimedialen *Cyberspace* und zum kommerziellen, immer geöffneten Einkaufsparadies, der 'dot-com'-Ökonomie.

1.1.2 Die Schaffung des WWW

Der Aufschwung und die umfassende Verbreitung des Internet ist einer Errungenschaft des europäischen Labors für Elementarteilchenforschung CERN (*Conseil Européen pour la Recherche Nucléaire*) in Genf zu verdanken. Mit dem raschen Wachsen und der Internationalisierung der Forschergruppen stellte sich heraus, dass die bisherige Infrastruktur des Internet, das maßgeblich zum Austausch der Forschungsergebnisse genutzt wurde, nicht mehr adäquat war. So wurde nach einem Verfahren gesucht, mit dem die Informationsquellen mittels *Hyperlinks* untereinander direkt verknüpft werden konnten. Der CERN-Mitarbeiter Tim Berners-Lee hatte 1989/1990 die Idee,

HTML

- die Dokumente in einer speziellen Seitenbeschreibungssprache HTML (*Hypertext Markup Language*) aufzubereiten und diese untereinander durch Hyperlinks zu verbinden, wobei

URL

- die Dokumenten-Referenzen über einheitliche Adressen URL (*Uniform Resource Locator*) erfolgen sollten und

HTTP

- die Verknüpfung über ein neues, einfaches Protokoll HTTP (*Hypertext Transport Protocol*) abgewickelt werden sollte.

Diese Idee brachte den Vorteil, dass nun nicht mehr der Systemadministrator des Servers, sondern der Dokumenten-Eigentümer für die Verknüpfung der Informationen verantwortlich war [Abb. 1.1-1]. Das nach dieser Idee weltweit verteilte System stellt heute unter dem Namen *World Wide Web* (WWW) – auch kurz *Web* genannt – die wichtigste Informationsquelle dar. WWW bildet ein weltweites Geflecht (Web) von Rechnern, die als *Webserver* fungieren und verschiedene Informationen enthalten.

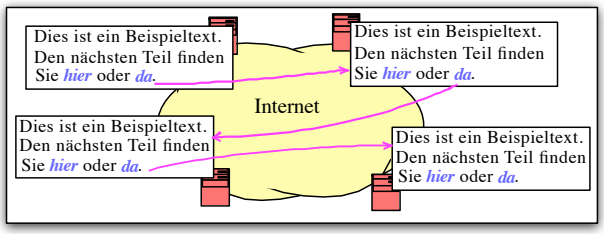


Abb. 1.1-2: Verknüpfung von Dokumenten auf unterschiedlichen Servern mittels Hyperlinks

WWW als Web-Dienst

Zusammen mit seinem Kollegen Robert Cailliau schrieb Tim Berners-Lee den ersten graphischen *Webbrowser* (als Software zur Darstellung der Web-Inhalte) sowie den ersten Webserver. Neben der graphischen Version wurde auch bald eine zeichenorientierte Browser-Version entwickelt, die weitgehend plattformunabhängig war. Mit der Verbreitung von Webbrowsern war der Siegeszug des WWW nicht mehr aufzuhalten. Heute spricht man in Bezug auf den Transport der verschiedenen Informationen im WWW vom *Web-Dienst*.

Hauptkomponenten des Webdienstes

Der Web-Dienst stellt einen Internetdienst auf grafischer Basis dar, der hauptsächlich zur Informationsabfrage verwendet wird. Die für die Realisierung des Webdienstes erforderlichen Komponenten zeigt Abb. 1.1-3.

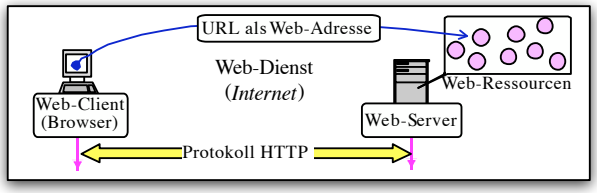


Abb. 1.1-3: Hauptkomponenten des Webdienstes – URL dient als Adresse

Die Grundkomponenten des Webdienstes sind:

- Eine Software für die Darstellung von Web-Inhalten in Form von *Webseiten* (*Web-Pages*) auf dem Bildschirm des Rechners. Diese Software stellt einen *Web-Client* dar und man bezeichnet sie als *(Web)-Browser*. Ein Browser zeigt die angeforderte Webseite an und bietet zahlreiche Funktionen für die Navigation im Web-Dienst.

  Webbrowser

- Einheitliche *Web-Adressen* zur Angabe der Lokation von Web-Inhalten, die man auch Web-Ressourcen nennt. Eine Web-Ressource stellt oft eine Datei in beliebigem Format (wie z.B. HTML, JPEG oder GIF) dar. Als einheitliche Web-Adressen wird ein *URL* (*Uniform Resource Locator*) verwendet. `http://www.hs-fulda.de/fb/ai` ist ein Beispiel hierfür. Die Adressierung von Web-Ressourcen wird noch näher erläutert.

  URLs als Web-Adressen

Webserver

- Webserver mit *Web-Inhalten* (Web-Ressourcen), auf die über das Internet zugegriffen werden kann. Die Web-Inhalte werden auch *Web-Content* genannt. Auf einem Webserver können auch herkömmliche Programme abgespeichert und an den Web-Dienst über eine Software-Schnittstelle, beispielsweise CGI (*Common Gateway Interface*), angebunden werden. Diese Programme können über das Internet aufgerufen werden.

HTML

- Eine abstrakte Sprache für die Beschreibung von *Webseiten*. Eine Webseite besteht in der Regel aus mehreren Web-Objekten und wird als Hypertext dargestellt. Für die Darstellung von Webseiten verwendet man die Seitenbeschreibungssprache HTML (*Hypertext Markup Language*), die in den Jahren 1989/1990 entwickelt wurde. HTML wird beständig weiterentwickelt und modifiziert, sodass es bereits mehrere HTML-Varianten (derzeit HTML 5) gibt. Ergänzt wird durch eine Formatierungssprache, *Cascading Style Sheets* (CSS) (aktuelle Version 3), durch die eine Trennung zwischen Inhalt und Format möglich wird.

Protokoll HTTP

- Ein Protokoll für den Transport von Web-Inhalten zwischen Browsern und Webservern. Hierfür dient das HTTP (*Hypertext Transport Protocol*). Hat ein Benutzer eine Webseite angefordert (z.B. indem er einen Hyperlink auf dem Bildschirm angeklickt hat), sendet sein Browser die Anforderung (d.h. einen HTTP-Request) an den durch die URL angegebenen Webserver. Dieser empfängt diese Anforderung und sendet eine Antwort (d.h. einen HTTP-Response), in der sich der angeforderte Web-Inhalt befindet, an den Browser zurück.

HTTP nutzt TCP

Für die Übertragung der Web-Inhalte zwischen Webserver und -browser nutzt HTTP das verbindungsorientierte Transportprotokoll TCP (*Transmission Control Protocol*). Dies bedeutet, dass eine TCP-Verbindung für die Übermittlung von Web-Inhalten zwischen Web-Client und -Server aufgebaut werden muss. Das Protokoll TCP wird in Abschnitt 3.3 detailliert beschrieben.

Adressierung von Web-Ressourcen

Um die Lokation einer gewünschten Web-Ressource im Internet anzugeben, braucht man die *Web-Adresse*. Was muss aber eine Web-Adresse angeben und wie sieht sie aus? Abb. 1.1-4 zeigt, was man beim Web-Dienst zu tun hat.

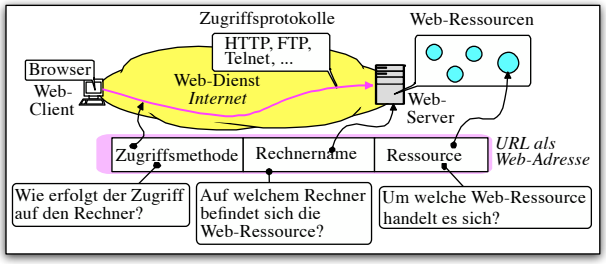


Abb. 1.1-4: Prinzip der Adressierung beim Web-Dienst

Was muss eine Web-Adresse enthalten?

Beim Zugriff auf eine Ressource muss folgendes angegeben werden [BRS03]:

- Die Art und Weise wie der Zugriff auf den Webserver erfolgt, also die Zugriffsmethode, d.h. welches Protokoll (HTTP, FTP, ...) verwendet wird.
- Der Rechner, auf dem sich die gewünschte Ressource befindet. Man muss auf den Rechner verweisen, um ihn eindeutig zu lokalisieren.
- Die Ressource, um die es sich handelt.

1.1.3 Internet nach der Etablierung des WWW

Internet-Strukturierung

Das Internet ist nach der Geburt des WWW ein so komplexes weltweites Rechnernetz geworden, dass es nicht möglich ist, hier die Struktur seiner physikalischen Vernetzung zu zeigen. Sie ist unbekannt und wächst ständig. Das Internet ist aber nach einem hierarchischen Prinzip aufgebaut. Wie Abb. 1.1-5 illustriert, stellt das Internet eine Vernetzung von Rechnern dar, in der man mehrere Schichten unterscheiden kann.

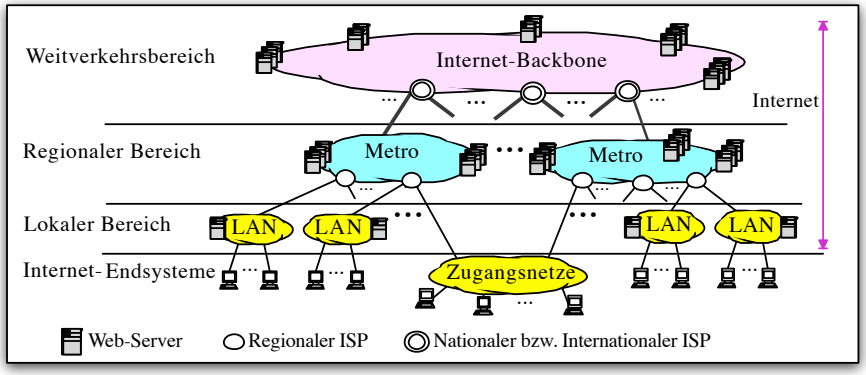


Abb. 1.1-5: Allgemeine Internet-Strukturierung – Aufbau als baumartige Struktur
ISP: Internet Service Provider; LAN: Local Area Network; Metro: Metro(politan)-Netz

Die untere Schicht bilden lokale Netzwerke (LANs) mit den Webservern, die den privaten Firmen, öffentlichen Institutionen, Hochschulen und anderen Organisationen gehören; sie können als *lokaler Internet-Bereich* angesehen werden. Die mittlere Schicht bilden regionale Netze mit regionalen IInternetdienstnbietern, sog. ISPs (*Internet Service Provider*). Diese Schicht stellt den regionalen Internet-Bereich dar. Bei den regionalen Netzen handelt es sich in der Regel um Hochgeschwindigkeitsnetze innerhalb von Großstädten, weshalb man sie als *Metro-Netze* bzw. *City-Netze* bezeichnet, die heute von häufig lokalen Netz-Providern zur Verfügung gestellt werden. Die obere Schicht, die den Internet-Weitverkehrsbereich darstellt, bilden nationale und internationale Hochgeschwindigkeitsnetze mit nationalen bzw. internationalen ISPs. Die nationalen und internationalen Hochgeschwindigkeitsnetze werden miteinander gekoppelt und bilden das *Internet-Backbone*, auch *Internet-Core* genannt.

Jeder ISP stellt einen Internetzugangspunkt dar, der auch als Einwahlknoten bzw. als POP (*Point of Presence*) bezeichnet wird.

1.1.4 Meilensteine der Internet-Entwicklung und Trends

Das Internet hat sich unmittelbar nach der Etablierung des WWW rasant entwickelt und adaptiert sich mit einem hohen Tempo an den Bedarf der Nutzer stetig weiter. Dies möchten wir jetzt in kurzer Form näher zum Ausdruck bringen. Hierfür zeigt Abb. 1.1-6 wesentliche Meilensteine bisheriger Internet-Entwicklung seit 1990 sowie wichtige Entwicklungstrends.

Bisherige Internet-Entwicklungen und Entwicklungstrends wurden in Abb. 1.1-6 zu sechs Schwerpunkten zusammengefasst; auf diese gehen wir jetzt kurz ein.

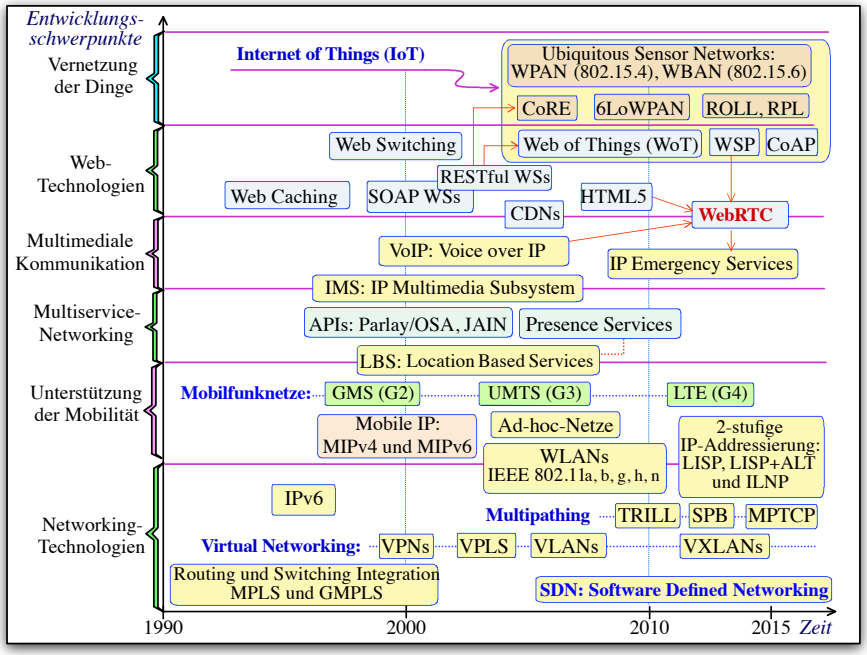


Abb. 1.1-6: Internet und IP-Netze; Meilensteile der bisherigen Entwicklung und Trends
CoRE: Constrained RESTful Environment, G: Generation, ROLL: Routing over Low power and Lossy neworks, RPL: Routing Protocol for Low power and Lossy networks, WPAN: Wireless Personal Area Network, WBAN: Wireless Body Area Network, WS: Web Services, WSP: WebSocket Protocol

Meilensteine bei Networking-Technologien

Integration von Routing und Switching

In lokalen Netzwerken wurden bereits zu Beginn der 90er Jahre sowohl Router als auch Switches eingesetzt und man hat damals von Routing und *Switching Integration* gesprochen. Diese Integration hat auch im Backbone des Internet und in großen privaten IP-Netzen stattgefunden. Folglich hat man versucht, die beiden Techniken Routing und Switching in einer Netzwerkkomponente (als Multi-Layer-Switch bezeichnet) zu integrieren. Ein besondere Art der Integration von Routing und Switching liegt dem Konzept MPLS (*Multi-Protocol Label Switching*) zugrunde [Abb. 1.4-4]. MPLS wird in Abschnitt 11.2 beschrieben.

MPLS, GMPLS

Bei MPLS werden die IP-Pakete über ein Netz als Gänsemarsch übermittelt [Abb. 11.1-1]. Dadurch entsteht die Möglichkeit, die Dienstgüte (*Quality of Service*) auf einem geforderten Level zu garantieren. Dies ist für die multimediale Kommunikation von enormer Bedeutung. Um das MPLS-Konzept in optischen Netzen, in denen man WDM (*Wavelength Division Multiplexing*) verwendet, einsetzen zu können, wurde GMPLS (*Generalized MPLS*) entwickelt [Abschnitt 11.3]. In den 90er Jahren hat man bei IP-Netzen mit (G)MPLS von *Next Generation IP Networks* gesprochen.

IPv6

Als Meilenstein in der Internet-Entwicklung kann das in 1994 spezifizierte Konzept von IPv6 angesehen werden. Es sei hervorgehoben, dass damals die Knappheit von offiziellen IPv4-Adressen die treibende Kraft der Entwicklung von IPv6 war. Mitte der 90er

Jahre wurde aber die als NAT (*Network Address Translation*) bezeichnete Möglichkeit 'entdeckt', die privaten IPv4-Adressen nutzen zu können [Abschnitt 2.3.3]. Das NAT-Konzept, insbesondere dessen Variante PAT (*Port Address Translation*), hat dazu beigetragen, dass man IPv6 in der Tat damals (und sogar bis Ende des ersten Jahrzehnts dies Jahrhunderts) noch nicht unbedingt gebraucht hat. Die Ära von IPv6 hat aber erst 'richtig' nach 2010 begonnen; unmittelbar nachdem die letzten offiziellen IPv4-Adressen vergeben wurden.

Virtual Networking

Schon in der zweiten Hälfte der 90er Jahre hat man mit *Virtual Networking* begonnen. Physikalische Leitungen im WAN-Bereich wurden durch virtuelle Verbindungen ersetzt, und es entstanden *Virtual Private Networks* (VPNs [Abschnitt 12.1]). Bereits zu Anfang dieses Jahrhunderts konnte man schon mehrere Ethernet-Segmente dank des (G)MPLS-Einsatzes über virtuelle Leitungen an einen zentralen Ethernet-Switch (Layer-2-Switch [Abb. 13.2-2]) anbinden und auf diese Weise ein standortübergreifendes, verteiltes Ethernet einrichten; damit wurde VPLS (*Virtual Private LAN Service* [Abschnitt 12.2-10]) geboren. Etwa zur gleichen Zeit bildet man IP-Subnetze in lokalen Netzwerken als beliebige Gruppen von Rechnern und bezeichnet man diese Gruppen als VLANs (*Virtual LANs*). Durch die Virtualisierung von Rechnern besteht heute – theoretisch gesehen – dank dem Konzept VXLAN (*Virtual Extensible LAN*) die Möglichkeit, aus virtuellen Rechnern (*Virtual Machines*) bestehende VLANs sogar weltweit zu bilden.

Multipathing

Um parallele, über mehrere Datenpfade verlaufende Kommunikation zwischen Rechnern, insbesondere in Datacentern, zu ermöglichen, wurden hierfür die Konzepte TRILL (*TRansparent Interconnection of Lots of Links*), SPB (*Shortest Path Bridging*) und MPTCP (*Multipath TCP* [Abschnitt 6.5]) entwickelt. Mit TRILL und SPB können standortübergreifende VLANs gebildet werden. Mittels von SPB kann auch ein verteilter, virtueller Ethernet-Switch auf Basis eines Ethernet-basierten Netzwerks – sogar eines standortübergreifenden Netzwerks – eingerichtet werden und die an diesem virtuellen Ethernet-Switch angeschlossenen Rechner können auch zu verschiedenen VLANs zugeordnet werden.

Technologien zur Unterstützung der Mobilität

Der Einsatz tragbarer Rechner (insbesondere Laptops und Smartphones) hat dazu geführt, dass von der IEEE mehrere Standards für WLANs (*Wireless LANs*) im ersten Jahrzehnt dieses Jahrhunderts spezifiziert wurden. Ein WLAN ermöglicht aber nur eine räumlich beschränkte Mobilität von tragbaren Rechnern. Heutzutage werden jedoch oft in Wirt-Servern mit zahlreichen virtuellen Rechnern mehrere, aus den in ihnen eingerichteten virtualisierten Rechnern bestehende, virtuelle Netzwerke gebildet; sie werden oft als *Clouds* bezeichnet. Dringlich sind die Konzepte hierfür nötig, eine aus mehreren virtuelle Rechnern enthaltene Cloud weltweit transferieren und an verschiedenen Standorten einsetzen zu können, ohne dass die IP-Adressen von Rechnern in der Cloud geändert werden müssen. Um diese Traumvorstellung zu verwirklichen, ist eine neue zweistufige, flexible IP-Adressierung notwendig. Wie diese zu realisieren und zu nutzen ist, beschreiben die in Abschnitt 13.7 präsentierten Konzepte LISP (*Locator/ID Separation Protocol*), LISP+ALT (*LISP Alternative Logical Topology*) und ILNP (*Identifier-Locator Network Protocol*).

Software Defined Networking

Die Virtualisierung von Rechnern und der zunehmende Bedarf an flexiblen, spontanen und an Geschäftsprozesse angepassten IT-Diensten verlangen neue Ideen zur variablen und raschen Bereitstellung von Netzwerkdiensten. *Software Defined Networking* (SDN) stellt eine solche Idee dar und ist als enorm wichtiger Entwicklungstrend im Internet und in Netzwerken mit IP zu betrachten. SDN ermöglicht die Bereitstellung universeller und programmierbarer Netzwerkknoten zur Weiterleitung von Daten. Diese Netzwerkknoten

können fast alle denkbaren Netzwerkfunktionen erbringen – und dies sogar parallel für die beiden Internetprotokolle IPv4 und IPv6. Dadurch können beim SDN verschiedene programmierbare Netzwerkdienste (*Programmable Network Services*) realisiert werden. Folglich kann man beim SDN sogar von Netzwerkprogrammierbarkeit (*Network Programmability*) sprechen.

Unterstützung der Mobilität

MIPv4 und MIPv6

Die Unterstützung der Mobilität im Internet und in IP-Netzen war ein großes Thema schon seit Beginn der Internet-Ära. Bereits Mitte 90er Jahre wurden die beiden Protokolle MIPv4 (*Mobile IPv4*) und MIPv6 (*Mobile IPv6*) konzipiert, um die Mobilität von Rechnern zwischen IP-Subnetzen zu ermöglichen. Kapitel 15 widmet sich der Unterstützung der Mobilität in IP-Netzen mit MIPv4 und MIPv6.

UMTS und LTE und WLANs

Erst die Mobilfunknetze der 3-ten und 4-ten Generation (G3 und G4), d.h. UMTS (*Universal Mobile Telecommunications System*) als G3 und LTE (*Long Term Evolution*) als G4, haben dazu beigetragen, dass verschiedene Arten von Smartphones heute als multifunktionelle Endgeräte am Internet dienen. Durch die breite Einführung von WLANs und die flächendeckende Verfügbarkeit von UMTS- bzw. von LTE-Diensten wurde das Problem 'Internet unterwegs mit Laptops, Tablets und Smartphones' gelöst. Als offenes Problem gilt aber noch die Mobilität kleiner Netzwerke und zwar so, dass die Adressen von Rechnern in diesen Netzwerken an jedem neuen Internetzugang nicht geändert werden müssen. Dieses Problem soll mit 2-stufiger IP-Adressierung gelöst werden [Abschnitt 13.7].

Ad-Hoc-Netzwerke

Es werden auch Konzepte entwickelt, um *Ad-Hoc-Netzwerke*, in denen sowohl die Knoten als auch die Endsysteme mobil sind, verwirklichen zu können. Diese Netzwerke haben eine große Bedeutung, da sie es spontan ermöglichen, fahrende Autos bis zu einer bestimmten Entfernung untereinander zu vernetzen: *Car-to-Car-Networks* (*C2C Networks*).

Multiservice-Networking

Konvergenz der Netze

Als wichtiger Trend bei IP-Netzen am Ende der 90er Jahre war das *Multiservice-Networking*, der die *Integration* (*Konvergenz*) der Netze aufgegriffen hat. Da verschiedene TK-Netze (wie PSTN, ISDN, GSM, UMTS) schon damals konvergiert haben, Stand der Wunsch im Raum, alle TK-Netze seitens des Internet als ein heterogenes TK-Netz zu nutzen, intelligente Netzdienste auf Basis des Protokolls IP zu entwickeln und diese den Teilnehmern an allen TK-Netzen über das Internet zugänglich zu machen.

Um intelligente Netzdienste auf Basis eines TK-Netzes zu entwickeln, muss man jedoch auf bestimmte Software-Schnittstellen, APIs (*Application Programming Interface*), im Netzkern zugreifen. Da diese noch in den 90er Jahren nur für den Netzbetreiber zugänglich waren, war es damals nicht möglich, dass die Netzdienste durch Dritte, also durch die Nicht-Netzbetreiber, konzipiert und entwickelt werden konnten. Um dies zu ändern, wurde zu mit Beginn dieses Jahrhunderts ein vom Netz unabhängiges API entwickelt, über das man auf die Dienste wichtiger TK-Netze zugreifen kann (siehe Abschnitt 1.4.4 in [Bad10]). Es handelt sich um Parlay/OSA (*Open Service Architecture*).

Parlay/OSA

Idee von Parlay/OSA: Die grundlegende Idee von Parlay/OSA besteht darin, dass man verschiedene TK-Netze als Kernnetz betrachtet. Die Dienste dieses Kernnetzes sind über Parlay/OSA API für die Nicht-Netzanbieter zugänglich. Somit können sie Netzanwendungen entwickeln und auf speziellen Application-Servern installieren, sodass man auf diese über das Internet zugreifen kann.

JAIN

Ein ähnliches Konzept wie bei Parlay/OSA wurde auch bei JAIN (*Java API for Integrated Networks*) von der Firma Sun Microsystems (jetzt Oracle) verfolgt.

LBS und Presence Services

Bei der Nutzung von Parlay/OSA kann man beliebige Netzdienste – z.B. auf Basis von UMTS bzw. von LTE – entwickeln und sie über das Internet zugänglich machen. Da die Lokation von mobilen Benutzern in Mobilfunknetzen UMTS und LTE mit einer bestimmten Genauigkeit bekannt ist, sind *Location Based Services* (LBS) realisierbar; und diese bilden die Grundlage für Presence Services. Die Einsatzmöglichkeiten von *Presence Services* sind sehr breit und haben in sozialen Netzwerken (z.B. Facebook) eine wichtige Funktion; sie ist aber nicht immer vorteilhaft.

IMS

Bei LBS und Presence Services spielt das IMS (*IP Multimedia Subsystem*) eine wichtige Rolle. IMS ermöglicht zusätzlich den Nicht-Netzanbieter, ihre Server am UMTS bzw. am LTE zu installieren und verschiedene Multimedia Services (Spiele, Filme, ...) zum Abruf per Internet anzubieten.

Multimediale Kommunikation

VoIP ist heute MMoIP

Mit Multiservice-Networking hängt auch die Realisierung der multimedialen Kommunikation zusammen. Die Entwickler haben seit geraumer Zeit davon geträumt, über ein Netz zu verfügen, über welches man alle Informationsarten (Audio, Video und Daten) übermitteln könnte. Die Konzepte und Protokolle für VoIP (*Voice over IP*) sind ein wichtiger Schritt in diese Richtung. In der Wirklichkeit ist VoIP mit dem Signalisierungsprotokoll SIP (*Session Initiation Protocol*) nicht nur VoIP, sondern *Multi-Media over IP* (MMoIP [Abb. 6.3-4]).

IP-Radio und IP-Fernsehen mit CDNs

Durch die Einführung der geeigneten Protokolle wie z.B. IP-Multicasting [Abb. 10.6-1] sind Dienste wie IP-Radio und IP-Fernsehen realisierbar. Bereits seit 2005 spricht man von *Triple Play*. Darunter versteht man das gebündelte Angebot der drei Dienste *Internet*, *IP-Telefonie* (VoIP) und *Fernsehen* für private Haushalte. Bei zeitversetztem Abruf der Echtzeitsendungen (wie Radio- bzw. Fernsehsendungen) spielen Web-basierte *Content Delivery Networks* (CDNs) mit zahlreichen Lieferungsservern eine Schlüsselrolle (siehe Kapitel 10 in [BRS03]). Und zwar bestimmt ein *Redirect-Router* – nach der Lokation des die Echtzeitsendung abrufenden Rechners – den Lieferungsserver, aus welchem (möglichst nicht weit gelegen vom abrufenden Rechner) die gewünschte Echtzeitsendung ausgeliefert werden soll, damit man eine gute Qualität gewährleisten kann.

IP Emergency Services

Eine wichtige Funktion herkömmlicher, öffentlicher Netze für die Sprachkommunikation ist die von ihnen angebotene Möglichkeit, in einem Notfall einen Notruf (*Emergency Call*) abzusetzen. Die Netze für die Sprachkommunikation bieten daher die Notrufdienste an; z.B. unter den Notrufnummern 110 für die Polizei und 112 für die Feuerwehr und die Rettungsdienste. Diese Dienste – und auch zahlreiche ähnliche – müssen zukünftig auch in öffentlichen VoIP-Systemen, also in der Tat im Internet, angeboten werden; hierbei spricht man von *IP Emergency Services* bzw. von *VoIP Emergency Services*.

LoST als 'Bruder' von DNS

Verschiedene Organisationen und Standardisierungsgremien sind in dieser Hinsicht aktiv. So wurde bei der IETF beispielsweise eine Arbeitsgruppe namens *Emergency Context Resolution with Internet Technologies* (ECRIT) ins Leben gerufen, um die Konzepte und Protokolle für die Realisierung von IP Emergency Services zu spezifizieren. Eine wichtige Funktion in Notrufsystemen im Internet besteht in der Ermittlung der IP-Adresse der richtigen Notrufleitstelle – und zwar aufgrund des in Form von URN (*Uniform Resource Name*) dargestellten Geschehens, d.h. was ist passiert, und des Standorts des Geschehens.

Um diese Funktion sicher zu realisieren, wurde das Konzept LoST (*Location-to-Service Translation*) entwickelt [RFC 5222]. LoST stellt eine hierarchische, baumartige Vernetzung von Rechnern dar (vergleichbar dem DNS) und kann folglich als jüngster Bruder vom DNS betrachten werden.

Web-Technologien

Web-Caching

In der ersten Phase der Web-Ära hat man hauptsächlich auf Webserver zugegriffen, um verschiedene Webinhalte in Form von Websites herunterzuladen. Um die Webinhalte, welche in der Zeit unverändert bleiben, nicht erneut über lange Strecken übermitteln zu müssen und schneller liefern zu können, hat man das in Rechnern gut bewährte Caching-Prinzip für das Internet übernommen – und so wurde Web-Caching 'geboren', siehe hierzu Kap. 8 in [BRS03]. Für das Web-Caching werden in der Regel spezieller Rechner als *Web-Cache-Server* eingesetzt. Große Internet Service Provider setzen mehrere Web-Cache-Server ein, sodass ein vernetztes Web-Caching-System entsteht. Für die Kommunikation zwischen Web-Caches wurde ICP (*Internet Cache Protocol*) entwickelt.

Web-Switching

Stark gefragter Webcontent wird schon lange nicht mehr nur auf einem Webserver abgespeichert, sondern auf mehreren Webservern, die sogar weltweit verteilt sein können. Diese Webserver bilden eine Gruppe von Servern, die unter einer IP-Adresse erreichbar sein muss und als *verteilter Webserver* angesehen werden kann. Als technische Basis dafür dient das Ende der 90er Jahre entwickelte *Web-Switching* und darunter versteht man die Verteilung von aus dem Internet kommenden Anfragen gemäß dem gefragten *Webcontent* auf mehrere Webserver. Web-Switching bildet heute die Grundlage für E-Commerce-Geschäfte; für Näheres darüber siehe Kap. 7 in [BRS03].

Web Services

Das Internet wurde zuerst hauptsächlich dafür benutzt, um per Browser den Zugriff auf verschiedene Informationen und Applikationen zu ermöglichen. Ende 90er Jahre hat man aber erkannt, dass Internet sich auch als universelle Plattform für die Kommunikation zwischen verteilten Anwendungen eignet und die Idee, webbasierte Dienste durch die Vernetzung von verteilten Anwendungen zu realisieren, hat zur Entstehung *Web Services* geführt. Ein Web Service ist ein Dienst auf Basis des Internet und des Protokolls HTTP, der durch die Vernetzung von verteilten Anwendungen und den Einsatz von XML (*eXtensible Markup Language*) zur Bildung von 'Nachrichten' – genauer von XML-Nachrichten – mit den zwischen Anwendungen zu übertragenden Daten erbracht wird. Jeder Web Service kann über einen Verzeichnisdienst veröffentlicht werden, um ihn bekannt, auffindbar und damit aufrufbar zu machen (vgl. Kapitel 11 in [BRS03]).

SOAP WS und RESTful WS

Es sei angemerkt, dass man zuerst bei *Web Services* (WS) das Protokoll SOAP (*Simple Object Access Protocol*) verwendet hat, um XML-Nachrichten in HTTP-Requests und -Responses zu übermitteln. Der Einsatz von SOAP bringt allerdings einen Nachteil mit: Die Web-Ressourcen (Objekte) können nicht direkt adressiert werden. Somit wurde später das Transferprinzip REST (*REpresentational State Transfer*) bei Web-Services eingesetzt, sodass Web-Ressourcen direkt mit URLs adressierbar sind; also in der Tat so, wie es ursprünglich bei Einführung des WWW vorgesehen wurde. Aus diesem Grund unterscheidet man zwischen SOAP-basiertem WS (*SOAP WS*) und REST-basiertem WS (*RESTful WS*).

CDNs, Request-, Content-Routing

Das klassische Modell der Webdienste, bei dem der Webcontent von einem Ursprungs-Server aus weltweit auf jede Webanfrage hin an den Benutzer geschickt wird, ist in einigen Situationen nicht mehr praktikabel; insbesondere bei zeitkritischem Content, also beim Streaming-Media (Video, Internet-TV, -Spiele etc.). Damit man Streaming-Media in

guter Qualität den Benutzern liefern konnte, ist Anfang dieses Jahrhunderts die Idee für *Content Delivery Networks* (CDNs) entstanden. Die Webserver mit dem gleichen zeitkritischen Content, sind jetzt nicht immer an einem Standort, sondern werden weltweit verteilt. In diesem Falle muss der 'günstigste' Webserver ausgewählt und die Anfrage an ihn gerichtet werden. Diesen Vorgang nennt man *Request-Routing* oder auch *Content-Routing*. Näheres über CDN findet sich in Kapitel 10 von [BRS03].

WebRTC – eine richtungsweisende Idee

Mit dem zweiten Jahrzehnt in diesem Jahrhundert versucht man eine richtungsweisende Idee zu verwirklichen, die sog. WebRTC (*Web Real-Time Communication*), eine Art *Web Video Telephony*. Diese Idee besteht darin, multimediale Echtzeitkommunikation mithilfe von HTLM5-fähigen Webbrowsern einfach zu realisieren, ohne dafür zusätzliche Softwaremodule installieren zu müssen. Zur Unterstützung von WebRTC können bei Bedarf von einem Webserver verschiedene RTC-spezifische Funktionsmodule heruntergeladen werden, und im Webbrowser diese (quasi automatisch) einzubauen. Bei WebRTC kann ein spezieller Server um RTC-Funktionen erweitert werden (hierzu eignet sich jeder Webserver) und als Manager von multimedialen Verbindungen zwischen Browsern dienen. Es ist zu erwarten, dass eine große Akzeptanz von WebRTC in Smartphones, Tablets und in Smart-TV in der Zukunft zu großen Veränderungen der heutigen Kommunikationslandschaft führen wird und die Videotelefonie per Smart-TV nur eine Frage der Zeit ist. Ferner besitzt WebRTC einen bedeutenden Einfluss auf die Realisierung von *IP Emergency Services*.

WebSockets

Für die Realisierung von WebRTC wurde das *WebSocket Protocol* (WSP) entwickelt [RFC 6455]. WSP wird auch beim Einsatz von Webtechnologien zur Vernetzung verschiedener 'Dinge' mit dem Internet eingesetzt; man spricht hierbei von *Web of Things* (WoT).

Vernetzung der Dinge – Internet of Things

USNs, IoT und WoT

Das Streben insbesondere nach mehr Energieeffizienz, mehr Lebensqualität und besserer Umweltüberwachung führt dazu, dass verschiedene Systeme zur drahtlosen Vernetzung von Sensoren und Aktoren – in der Tat zur Vernetzungen aller möglichen 'Dinge' — ständig und immer mehr an Bedeutung und an Verbreitung gewinnen; folglich entstehen *Sensornetze/Sensornetzwerke*. Weil Sensornetze überall und jederzeit zum Einsatz – z.B. zur Industrie-/Gebäude-/Heimautomation, Gesundheits-/Umweltüberwachung – kommen können, werden sie als *ubiquitäre Sensornetze* bzw. kurz als USNs (*Ubiquitous Sensor Networks*) bezeichnet[1]. USNs sind sehr stark ressourcenbeschränkt, insbesondere *energiearm* und *verlustbehaftet*; demzufolge spricht man von *Constrained Networks* bzw. von LLNs (*Low power and Lossy Networks*).

Internet of Things, Web of Things

Die Anbindung von USNs an das 'heutige' Internet führt zur Entstehung eines neuen Internetteils, welcher als *Internet of Things* (IoT) – also *Internet der Dinge* – bezeichnet wird [ITU-T Y.2060]. Im IoT kommen auch einige Web-Technologien zum Einsatz, sie müssen an die Besonderheiten von Sensornetzen angepasst werden und man spricht in diesem Zusammenhang von *Web of Things* (WoT); siehe hierzu ITU-T Y.2063.

Sensornetze werden oft nach IEEE-Standards 802.15.4 (WPAN, *Wireless Personal Area Network*) und 802.15.6 (WBAN, *Wireless Body Area Network*) aufgebaut und dabei wird IPv6 eingesetzt. Dadurch kann eine global eindeutige IPv6-Adresse jedem Sensor/Aktor zugewiesen werden und er ist dann über das Internet zugreifbar.

6LoWPAN

Um IPv6 in Sensornetzen auf Basis von WPANs nach IEEE 802.15.4 einsetzen zu können, wurde das Konzept 6LoWPAN (*IPv6 over Low-power PAN*) bei der IETF in RFC 4944 spezifiziert. 6LoWPAN gilt bereits als ein effizientes Konzept für den Einsatz von IPv6 in Sensornetzen.

CoRE und CoAP

Im IoT sollen REST-basierte Web Services, auch als *RESTful WSs* bezeichnet, zum Einsatz kommen. Diese müssen jedoch an ressourcenbeschränkte Umgebungen (*constrained environments*) in Sensornetzen adaptiert werden. Die Umsetzung dieser Anforderung hat sich die IETF Working Group CoRE (*Constrained RESTful Environments*) vorgenommen. Da das normale Webprotokoll HTTP in Sensornetzen praktisch nicht einsetzbar ist, wird dieses in Sensornetzen durch das neue, von der Working Group CoRE entwickelte Webprotokoll CoAP (*Constrained Application Protocol*) ersetzt.

ROLL und RPL

Sensornetze als LLNs können auch beliebig verteilte Strukturen mit intelligenten Knoten bilden, daher ist auch ein Routing-Protokoll in diesen Netzen nötig. Auf dem Gebiet *Routing over LLNs* ist die IETF Working Group ROLL (*Routing Over Low power und Lossy networks*) aktiv, und das Ergebnis ist u.a. das Routing-Protokoll RPL (*IPv6 Routing Protocol for Low-Power und Lossy Networks*).

1.2 Funktionen der Kommunikationsprotokolle

Fehlerursachen

In einem Netz können die zu übertragenden Daten verfälscht werden. Die Ursachen dafür sind meist auf die schlechte Qualität des Übertragungsmediums zurückzuführen. Eine Verfälschung der Daten kann auch durch äußere Einflüsse wie etwa starke elektromagnetische Felder in der Umgebung oder durch das *Nebensprechen* entstehen. Übertragungsstörungen führen nicht nur zu einer Datenverfälschung, sondern sogar zu einem Datenverlust. Um dies zu vermeiden, müssen entsprechende Funktionen in den Kommunikationsprotokollen enthalten sein. Diese Funktionen lassen sich in drei Gruppen aufteilen:

- **Fehlerkontrolle** (*Fault Control*),
- **Flusskontrolle** (*Flow Control*) und
- **Überlastkontrolle** (*Congestion Control*).

Datenverfälschungen und -verluste

Die *Fehlerkontrolle* umfasst alle Maßnahmen in einem Kommunikationsprotokoll, mit denen Datenverfälschungen und -verluste während der Übertragung entdeckt und beseitigt werden können. Die *Flusskontrolle* bedeutet eine gegenseitige Anpassung der Sende- und der Empfangsseite in Bezug auf die übertragene Datenmenge. Die *Überlastkontrolle* betrifft alle Vorkehrungen, die dazu dienen, ein Netz nicht zu überlasten. Bei der Überlastung eines Netzes müssen die übertragenen Datenblöcke oft verworfen werden und die Verweilzeit von Datenblöcken im Netz durch 'Staus' in Knoten nimmt stark zu. Im Folgenden werden diese Funktionen näher erläutert.

[1] Da in Sensornetzen sowohl (passive) Sensoren wie auch als Ausführungsorgane aktiv fungierende Aktoren vorhanden sind, sprechen wir allgemein von *Sensor-Aktor-Netzen/Netzwerken*.

1.2.1 Prinzipien der Fehlerkontrolle

Die Fehlerkontrolle hat die Aufgabe, jede fehlerhafte Situation während der Datenübertragung zu entdecken und zu beseitigen. Sie ist Bestandteil jedes Kommunikationsprotokolls und wird beim Empfänger mittels festgelegter Quittungen (Bestätigungen) und beim Sender durch die Zeitüberwachung realisiert. Im Weiteren werden alle möglichen Fehlersituationen dargestellt und notwendige Maßnahmen zu ihrer Beseitigung aufgezeigt.

Allen Kommunikationsprotokollen liegen zwei 'eiserne Regeln' zugrunde:

Erste 'eiserne Regel'

> Datenblöcke können während der Übertragung verfälscht werden. Deshalb muss nach dem Absenden jedes Datenblocks dieser im Speicher der Quellstation für den Fall gehalten werden, falls eine wiederholte Übermittlung notwendig ist.

Negative Auswirkungen infolge der Verfälschung von übertragenen Datenblöcken können durch die Umsetzung dieser Regel und durch eine wiederholte Übermittlung ausgeglichen werden. Abb. 1.2-1a zeigt die fehlerlose Übermittlung eines Datenblocks. Diese wird von der Empfangsseite positiv quittiert (bestätigt) und eine Kopie des Datenblocks in der Quellstation gelöscht. Auch eine Quittung stellt einen kurzen, vom Protokoll festgelegten Datenblock dar.

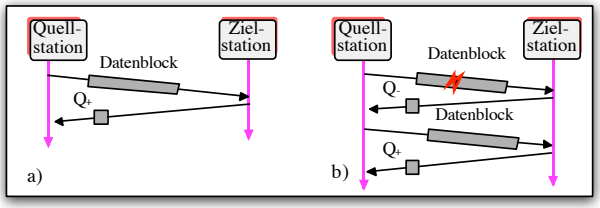


Abb. 1.2-1: Übermittlung eines Datenblocks: a) fehlerlos, b) fehlerhaft
Q+: positive Quittung, Q–: negative Quittung

In Abb. 1.2-1b tritt bei der Übermittlung des Datenblocks eine Störung auf, was eine negative Quittierung zur Folge hat. Der gestörte Datenblock wird durch die Zielstation einfach verworfen. Da in der Quellstation eine Kopie des betreffenden Datenblocks gehalten wird, sendet die Quellstation den gleichen Datenblock noch einmal – diesmal fehlerfrei – zu der Zielstation, die ihn positiv quittiert. Die Kopie des übertragenen Datenblocks kann nun in der Quellstation gelöscht werden.

Negative Quittung bei Störungen

Eine besondere Situation entsteht dadurch, dass nicht nur die Datenblöcke während der Übertragung verfälscht werden können, sondern auch die Quittungen. Wird eine positive Quittung so verfälscht, dass die Quellstation sie als negative Quittung interpretiert, führt dies zu einer unnötigen wiederholten Übermittlungen des betreffenden Datenblocks und zur Verdoppelung von Daten am Ziel.

Verfälschung von Quittungen

Der schlimmste Fall (*worst case*) bei der Übermittlung eines Datenblocks entsteht dann, wenn sowohl der übertragene Datenblock als auch dessen negative Quittung verfälscht werden. Wie Abb. 1.2-2 zeigt, empfängt die Quellstation in diesem Fall eine positive Quittung und könnte deshalb die Kopie des Datenblocks löschen. Dies würde aber zum Verlust des Datenblocks führen. Um einen solchen Fall zu bewältigen, müssen die Kommunikationsprotokolle zwei Stufen der Fehlerkontrolle realisieren. Die hier angesprochene Fehlerkontrolle bezieht sich nur auf die Übermittlung einzelner Datenblöcke, die oft auf-

grund der Segmentierung von zu übertragenden Dateien entstehen. Die Fehlerkontrolle muss auch auf Dateiniveau realisiert werden. Die einzelnen Datenblöcke, die zu einer Datei gehören, werden in der Zielstation zu einer Datei zusammengesetzt. Ist ein Datenblock der Datei in der Zielstation nicht vorhanden, sendet sie eine negative Quittung, die sich auf diese Datei bezieht. Die Quellstation muss dann entweder den verloren gegangenen Datenblock oder sogar die ganze Datei nachsenden.

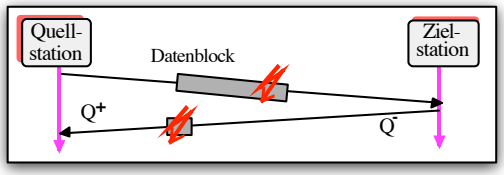


Abb. 1.2-2: Worst case einer Datenmittlung: Daten und Quittung werden verfälscht

Zweite 'eiserne Regel'

Die zweite 'eiserne Regel', die bei allen Kommunikationsprotokollen realisiert werden muss, um Datenverluste während der Übertragung zu erkennen, lautet:

> Datenblöcke können bei der Übertragung verloren gehen, sodass man nur eine begrenzte Zeit auf eine positive oder negative Quittung für einen Datenblock warten soll.

Dies muss über eine Zeitüberwachung realisiert werden, um Verluste von Datenblöcken während der Übertragung zu erkennen. Dazu ist im Protokoll eine maximale Wartezeit auf eine Quittung festzulegen. Eine solche Wartezeit wird auch als *Timeout* bezeichnet und kann als 'Geduldzeit' interpretiert werden. Nach dem Absenden eines Datenblocks muss die Überwachung der maximalen Wartezeit auf die Quittung aktiviert werden. Es stellt sich die Frage, wann die Datenblöcke während der Übermittlung eigentlich verloren gehen. Dass ein übertragener Datenblock bei einem plötzlichen Bruch der Leitung verloren geht, ist selbstverständlich, doch das ist selten der Fall. Die häufigste Ursache für den Verlust eines Datenblocks ist eine Verfälschung in seinem Header oder Trailer, sodass er auf der Leitung nicht vollständig erkannt und damit in der Zielstation nicht aufgenommen werden kann.

Geduldzeit

Abb. 1.2-3 illustriert die fehlerhafte Situation, in der ein Datenblock verloren gegangen ist. Nach dem Absenden des Datenblocks wird die 'Geduldzeit' überwacht. Kommt innerhalb dieser Zeit keine Quittung an, interpretiert dies die Quellstation als verloren gegangenen Datenblock und wiederholt die Übermittlung. Nach dem wiederholten Absenden kommt eine positive Quittung noch während der 'Geduldzeit' an und die Kopie des Datenblocks kann dann gelöscht werden.

Nummerierung von Datenblöcken

Auch eine Quittung kann verloren gehen. Wie Abb. 1.2-3b zeigt, wird dies ebenfalls mit Hilfe der Zeitüberwachung erkannt. In einem solchen Fall kann ein Datenblock in der Zielstation doppelt vorhanden sein. Deswegen muss für die Zielstation klar werden, dass es sich nicht um einen neuen Datenblock handelt, sondern um eine wiederholte Übermittlung. Werden die transferierten Datenblöcke nicht nummeriert, kann das zur Verdopplung von Daten am Ziel führen. Derartige Datenverdopplungen lassen sich mit

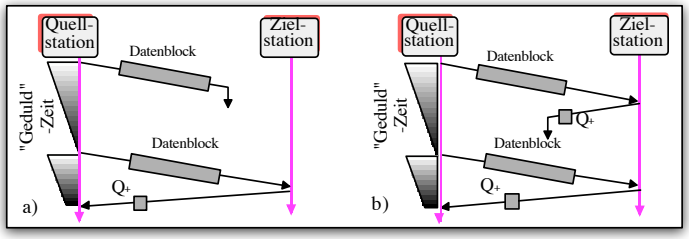


Abb. 1.2-3: Fehlerhafte Übermittlung: a) Datenblockverlust, b) Quittungsverlust

der Nummerierung von Datenblöcken ausschließen. Aus diesem Grund werden bei allen Kommunikationsprotokollen die übertragenen Datenblöcke nummeriert.

Modulo-Verfahren

Anmerkung: Bei einer fortlaufenden Nummerierung der übertragenen Datenblöcke kann die Zielstation erkennen, ob es sich um eine wiederholte Übermittlung handelt und somit einen doppelt vorhandener Datenblock entdecken. Eine Nummerierung der übertragenen Datenblöcke besteht darin, dass jedem Datenblock eine bestimmte Sequenznummer zugeteilt wird. Diese Nummerierung kann aber nicht beliebig fortgesetzt werden. Ursache hierfür ist die begrenzte Anzahl von Bit für die Nummernabspeicherung im Header des Datenblocks und deshalb werden die Datenblöcke nach dem Modulo-Verfahren nummeriert. In den meisten Fällen wird die Nummerierung nach dem Modulo 8 oder 128 realisiert. Bei Modulo 8 werden die einzelnen Datenblöcke von 0 bis 7 gekennzeichnet und verschickt. Ist die 7 als letzte Nummer vergeben worden, wird der Zähler zurückgesetzt und die Nummerierung startet bei 0. Äquivalent dazu funktioniert die Nummerierung nach dem Modulo-128-Verfahren, bei dem die Nummern bis 127 vergeben werden.

Bei der Nummerierung von Datenblöcken kann eine Gruppe von empfangenen Blöcken gleichzeitig durch die Zielstation quittiert werden, um damit die Verkehrslast im Netz durch eine geringere Anzahl von Quittungen zu reduzieren.

Nummerierungsfenster (Window)

Beim Aufbau einer Verbindung muss sichergestellt sein, dass die Quellstation den Datenblöcken jene Sequenznummern zuteilt, die auch von der Zielstation erwartet werden. Aus diesem Grund ist zu vereinbaren, welchen Zahlenwert das Nummerierungsfenster hat und mit welcher Sequenznummer bei der Übermittlung der Datenblöcke begonnen wird.

1.2.2 Realisierung der Flusskontrolle

Bedeutung der Flusskontrolle

Bei der Datenkommunikation tritt häufig der Fall ein, dass die Daten beim Sender rascher 'produziert' werden, als der Empfänger sie 'konsumieren' kann. So ist eine Situation vorstellbar, in der ein Großrechner im Netz eine große Menge von Daten an einen entfernten kleinen Drucker übermittelt. Der Großrechner muss, um ein Überfließen des Druckerspeichers zu verhindern, die Menge der zu übertragenden Daten der Aufnahmefähigkeit des Druckers anpassen. Die Anpassung muss durch entsprechende Kommandos vom Drucker gesteuert werden. Dieses einfache Beispiel weist auf die Bedeutung der gegenseitigen Abstimmung zwischen Quell- und Zielstation in Bezug auf die Menge der zu übertragenden Daten hin.

Ziel der Flusskontrolle

Unter *Flusskontrolle* versteht man alle Maßnahmen, die zur Anpassung der gesendeten Datenmenge der Quellstation an die Aufnahmekapazität der Zielstation führen. Die Flusskontrolle kann realisiert werden

- mittels der Meldungen *Halt* und *Weitersenden*,
- über einen *Kreditmechanismus* (Kredite), oder
- vermöge eines *Fenstermechanismus* (Window).

Meldungen: Halt, Weitersenden

Die einfache Flusskontrolle mit Hilfe der Meldungen *Halt* und *Weitersenden* verläuft wie folgt: Stellt der Empfänger fest, dass er nicht mehr in der Lage ist, die empfangenen Daten aufzunehmen, schickt er dem Sender die Meldung *Halt*. Der Sender ist nach dem Empfang von Halt verpflichtet, das Senden von Daten einzustellen, bis der Empfänger die Meldung *Weitersenden* übermittelt und damit den *Halt*-Zustand aufhebt. Ein Nachteil dieses einfachen Verfahrens besteht darin, dass eine Verfälschung der Meldungen *Halt* oder *Weitersenden* besondere Konsequenzen hat: Wird *Halt* während der Übertragung verfälscht und vom Sender als *Weitersenden* empfangen, so sendet er die Daten weiter. Kommt *Weitersenden* beim Sender als *Halt* an, wird der Sendeprozess auf Dauer gestoppt.

Flusskontrolle mittels Krediten

Bei einer Flusskontrolle mit Hilfe von Krediten erteilt der Empfänger dem Sender einige Kredite für die Übermittlung von Datenblöcken. Sind diese Kredite aufgebraucht, muss der Sender die Übermittlung einstellen. Ein Kredit definiert eine Anzahl von Datenblöcken, d.h. deren Sequenznummer, die der Sender abschicken darf, ohne auf eine Quittung vom Empfänger warten zu müssen. Hierbei ist die maximale Länge der Datenblöcke festgelegt. Im Normalfall werden die Kredite laufend erteilt, sodass ein ununterbrochener Datenverkehr aufrechterhalten werden kann. Die Übermittlung von Krediten muss vor Störungen geschützt werden. Bei der Störung einer Kreditmeldung könnte der Sender ohne weitere Kredite bleiben und der Empfänger auf weitere Datenblöcke warten. Damit wäre die Datenübermittlung blockiert. Es muss sichergestellt sein, dass eine Kreditmeldung nicht verdoppelt wird. Wäre dies nicht der Fall, könnte der Sender weitere Datenblöcke senden, die vom Empfänger nicht aufgenommen werden könnten.

Flusskontrolle über Fenstermechanismus

Die Flusskontrolle über einen Fenstermechanismus stützt sich auf eine Sequenznummer der übertragenen Datenblöcken. Vor der Datenübermittlung sprechen sich Quell- und Zielstation über ein *Fenster* innerhalb des Wertebereiches der Sequenznummern ab. Die Fenstergröße `W` bedeutet:

> Die Quellstation darf maximal `W` Datenblöcke absenden, ohne auf eine Quittung von der Zielstation warten zu müssen, d.h. `W` ist bei der Quellstation als Anzahl der Kredite zu interpretieren.

Bei der Zielstation stellt `W` die Kapazität des Empfangspuffers für die ankommenden Datenblöcke dar.

Beispiel: Abb. 1.2-4 zeigt den Fall, in dem die Fenstergröße `W = 3` und die Datenblöcke nach dem Modulo-8-Verfahren nummeriert werden. Bei `W = 3` darf die Quellstation drei Datenblöcke absenden, ohne auf eine Quittung warten zu müssen. Abb. 1.2-4 zeigt gleichzeitig die freie Sequenznummer, die der Sender für die Nummerierung verwenden darf. Da `W = 3`, sind maximal drei Nummern zu vergeben. Wie hier ersichtlich, ist während der Übermittlung der ersten drei Datenblöcke keine Quittung angekommen, also muss die Quellstation den Sendeprozess unterbrechen. Dies führt zu einer Senderblockade. Nach der ersten positiven Quittung, mit der die Datenblöcke mit den Nummern 0 und 1 positiv quittiert wurden, darf sie zwei weitere Datenblöcke senden. Dieses Beispiel zeigt,

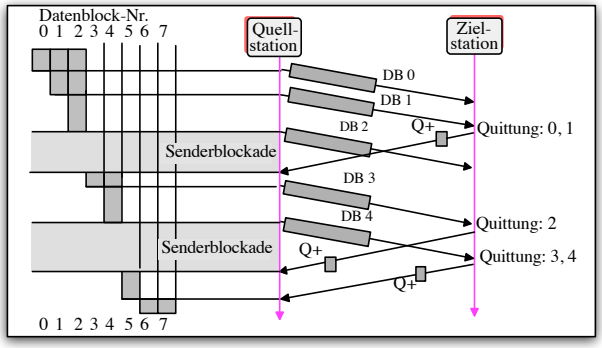


Abb. 1.2-4: Veranschaulichung der Flusskontrolle über den Fenstermechanismus

welche Auswirkungen die Fenstergröße auf die Auslastung des Übertragungsmediums hat. Insbesondere im Fall `W = 1` muss man nach dem Absenden jedes Datenblocks den Sendeprozess stoppen. Dies führt selbstverständlich zu einer schlechten Ausnutzung des Übertragungsmediums.

Die Fenstergröße kann als Kredit für die Vergabe von Nummern für die abzusendenden Datenblöcke interpretiert werden. Die meisten Kommunikationsprotokolle realisieren die Flusskontrolle nach dem Fenstermechanismus.

1.2.3 Überlastkontrolle

Ein Netz hat eine bestimmte Aufnahmekapazität, d.h. zu jedem Zeitpunkt kann sich darin nur eine begrenzte Anzahl von Datenblöcken befinden. Wird diese Anzahl überschritten, hat dies die folgenden negativen Auswirkungen:

- *Die Aufnahmepuffer im Netz (in Knoten) sind voll*; dies führt dazu, dass die im Netz eintreffenden Datenblöcke verworfen werden müssen.
- *Es bilden sich Warteschlangen von Datenblöcken vor den Übertragungsleitungen*; durch die so verursachten großen Verweilzeiten der Datenblöcke im Netz entstehen große Verzögerungen der übertragenen Datenblöcke.

Die Maßnahmen, mit denen eine Überlastung des Netzes verhindert wird, bezeichnet man als *Überlastkontrolle* (Congestion Control). Die wichtigsten Kriterien für die Beurteilung der Überlastung von Netzen sind:

Congestion Control

- Durchsatz (*Throughput*) und
- Datenverweilzeit (*Latenzzeit*, *Delay*) im Netz.

Unter dem *Durchsatz eines Netzes* versteht man den Anteil des Datenverkehrs, der von dem Netz akzeptiert wird. Den Verlauf des Durchsatzes in Abhängigkeit vom Gesamtdatenverkehr zeigt Abb. 1.2-5a. Ist der Datenverkehr im Netz klein (kleine Belastung), werden alle ankommenden Daten durch das Netz aufgenommen; dabei müssen normalerweise keine Vorkehrungen gegen die Überlast ergriffen werden. Bei hoher Netzbelastung

Durchsatz

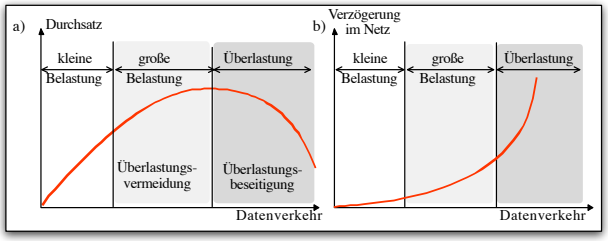


Abb. 1.2-5: Auswirkungen der Netzüberlastung: a) auf den Durchsatz, b) auf die Datenverweilzeit im Netz

dagegen müssen bestimmte Maßnahmen getroffen werden, um eine Überlastung zu vermeiden, und sie führen zur Einschränkung der Datenmenge, die ins Netz gesendet wird.

Ist der Datenverkehr im Netz so groß, dass das Netz überlastet ist, müssen andere Aktionen eingeleitet werden, um die bestehende Überlastung zu beseitigen. Wie in Abb. 1.2-5a ersichtlich, nimmt der Durchsatz in der Überlastsituation mit zunehmendem Datenverkehr sehr stark ab.

Verweilzeit im Netz

Abb. 1.2-5b veranschaulicht, welche Auswirkungen die Netzbelastung auf das Verhalten der Datenverweilzeit (Latenzzeit) im Netz hat. In einer Überlastsituation muss also mit großen Verzögerungen für die Datenübertragung im Netz gerechnet werden. Die wichtigste Maßnahme für die Vermeidung von Überlasten besteht in der Einschränkung der Datenströme, die ins Netz fließen. Welche Maßnahmen gegen die Überlastung in einzelnen Netzen und Kommunikationsprotokollen ergriffen werden, hängt auch von der Realisierung der Flusskontrolle ab. Komplexere Verfahren der Flusskontrolle können z.B. mittels *Explicit Congestion Control* (ECN) realisiert werden, wie dies in Abschnitt 3.5 besprochen wird.

1.3 Schichtenmodell der Kommunikation

Was ist OSI?

Als man Mitte der 70er Jahre versuchte, die Rechner unterschiedlicher Hersteller miteinander zu vernetzen, hat sich folgendes Problem ergeben: Es sind dringend Kommunikationsregeln nötig, damit ein Rechner des Herstellers *X* mit einem Rechner des Herstellers *Y* kommunizieren kann. Es sollte möglich sein, dass jeder Rechner für die Kommunikation mit allen anderen Rechnern *offen* (bereit) ist. In diesem Zusammenhang wurde bereits damals von der Vernetzung offener Systeme – also von *Open System Interconnection* (OSI) – gesprochen und nach einem Modell für ihre Verwirklichung gesucht.

OSI-Referenzmodell

Daher wurde ein Schichtenmodell eingeführt, dass die Prinzipien der Kommunikation zwischen verschiedenen Systemen beschreibt und die OSI-Vorstellung ermöglicht. Es wird deshalb *OSI-Referenzmodell* genannt. Standardisiert wurde es von ISO (*International Organization for Standardization*) und es wird auch als *ISO/OSI-Referenzmodell* bzw. kurz als *ISO/OSI-Modell* bezeichnet.

1.3.1 Konzept des OSI-Referenzmodells

Idee von OSI

Die Idee von OSI illustriert Abb. 1.3-1. Gemäß OSI wird ein Rechner als *offenes System* angesehen. Diese Systeme werden durch Übertragungsmedien untereinander verbunden und enthalten entsprechende *Kommunikationsprotokolle*, nach denen *logische Verbindungen zwischen Applikation* in den einzelnen Systemen nach Bedarf aufgebaut und *Nachrichten* bzw. allgemein *Daten* übertragen werden können.

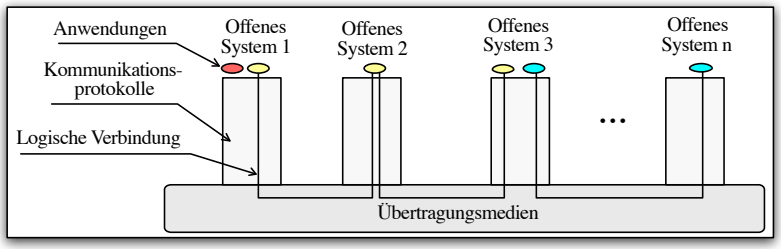


Abb. 1.3-1: Idee von OSI: Jedes System soll mit jedem anderen kommunizieren können

Um die Kommunikationsprotokolle für die Verwirklichung der Zielvorstellung von OSI zu entwickeln, wurde die komplexe Aufgabe der Kommunikation zwischen verschiedenen Systemen so auf sieben Teilaufgaben verteilt, dass diese den einzelnen Schichten, die in einer Hierarchie zueinander stehen, zugeordnet werden. Dadurch ist ein OSI-Referenzmodell mit sieben Schichten entstanden; man spricht hier auch vom *OSI-Schichtenmodell*.

Allgemeines OSI-Schichtenmodel

Ein Rechnernetz enthält aber nicht nur die Rechner als Endsysteme, sondern auch die Netzknoten (Router, Switches) als *Zwischensysteme*. Die Aufgabe der Kommunikation in Zwischensystemen kann aber zu drei untereinander liegenden Schichten zusammengefasst werden. Daher enthalten die Zwischensysteme nur die ersten drei Schichten. Abb. 1.3-2 zeigt die allgemeine Struktur des OSI-Referenzmodells. Die unterste Schicht 1 repräsentiert die physikalische Netzanbindung, also die Übertragungstechnik. Die Schichten von 2 bis 6 repräsentieren bestimmte Funktionen der Kommunikation. Schicht 7 enthält die Anwendungen (*Applikationen*).

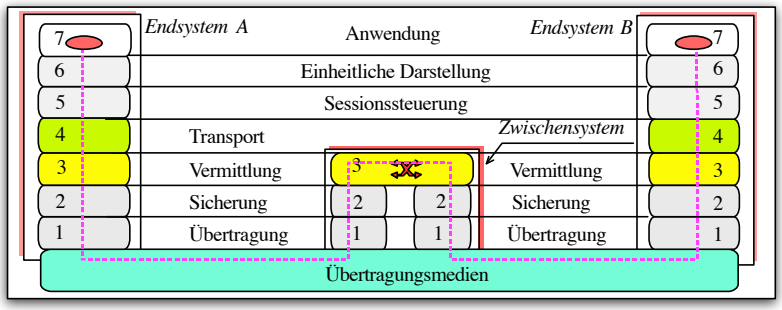


Abb. 1.3-2: OSI-Referenzmodell: Zerlegung der Kommunikationsaufgabe in 7 Schichten

Funktionen der Schichten

Die einzelnen Schichten im OSI-Referenzmodell sind:

1. **Physikalische Schicht** (Übertragungsschicht, *Physical Layer*)
   Sie definiert die mechanischen und elektrischen Eigenschaften sowie die Funktionen und die Abläufe bei der Bitübertragung.
2. **Sicherungsschicht** (*Data-Link Layer*)
   Diese Schicht garantiert eine sichere Übertragung zwischen zwei direkt benachbarten Stationen (Knoten). Dazu werden die übertragenen Bit in *Frames* (Rahmen) zusammengefasst und am Ende mit einer Prüfsumme versehen. Dadurch ist eine Fehlererkennung möglich. In LANs wird die Schicht 2 in zwei Teilschichten aufgeteilt: *Schicht 2a* als *MAC-Schicht* (*Media Access Control*), die den Zugriff auf das Übertragungsmedium regelt, und *Schicht 2b* als *LLC-Schicht* (*Logical Link Control*) [Abb. 12.1-1], die eine Sicherungsschicht darstellt.
3. **Netzwerkschicht/Vermittlungsschicht** (*Network Layer*)
   Diese Schicht hat die Aufgabe, die Daten blockweise zwischen Endsystemen zu übermitteln. Die innerhalb dieser Schicht übertragenen Datenblöcke werden oft *Pakete* genannt. Schicht 3 stellt eine *Paketvermittlungsschicht* dar.
4. **Transportschicht** (*Transport Layer*)
   Die Transportschicht hat u.a. die Aufgabe, eine gesicherte *virtuelle Ende-zu-Ende-Verbindung* für den Transport von Daten zwischen den Endsystemen bereitzustellen. Die Aufgaben der Transportschicht bestehen vor allem in der Korrektur der Übermittlungsfehler und sind von den Protokollen der Schicht 2 und 3 sehr stark abhängig.
5. **Sitzungsschicht** (*Session Layer*)
   Sie ist die unterste anwendungsorientierte Schicht und regelt den Auf- und Abbau von Kommunikationsbeziehungen (Sitzungen, Sessions) sowie deren Wiederherstellung nach Störungen im Transportsystem. Hier findet die *Synchronisation* und somit der *geregelte Dialogablauf* zwischen zwei Kommunikationsprozessen statt.
6. **Darstellungsschicht** (Präsentationsschicht, Presentation Layer)
   Die Umsetzung verschiedener Darstellungen der Information (z.B. die Zeichensätze ASCII und EBCDIC) auf ein einheitliches Format auf der Senderseite ist die Aufgabe der Darstellungsschicht. Diese Schicht kann auch Funktionen enthalten, mit denen Daten komprimiert, konvertiert und verschlüsselt werden können. Vor der Web-Ära war das ASN.1-Konzept (*Abstract Syntax Notation*) für diese Schicht von großer Bedeutung. Inzwischen wurde die Aufgabe von ASN.1 durch XML (*eXtensible Markup Language*) übernommen.
7. **Anwendungsschicht** (*Application Layer*)
   In dieser Schicht sind die sog. OSI-Anwendungsprogramme angesiedelt. Zu den wichtigsten OSI-Standardanwendungen gehörten in den 90er Jahren E-Mail (X.400) und verteilter Verzeichnisdienst (X.500) und Filetransfer (FTAM).

Im Allgemeinen kann die Schicht `n-1` im OSI-Referenzmodell als Erbringer bestimmter Kommunikationsdienste für die Schicht `n` angesehen werden. Die mit der Kommunikation verbundenen Aufgaben in den Endsystemen können bestimmten Klassen von Aufgaben zugeordnet werden. Abb. 1.3-3 bringt dies deutlicher zum Ausdruck.

Übermittlungsdienste

Die ersten drei Schichten realisieren die *Übermittlungsdienste*. Schicht 1 realisiert die Übermittlung von Daten bitweise zwischen zwei direkt verbundenen Stationen (d.h. zwischen einem Endsystem und seinem Netzknoten bzw. zwischen zwei benachbarten Netzknoten). Die ersten zwei Schichten realisieren die gesicherte Übermittlung von Daten in Form von Frames zwischen zwei direkt verbundenen Stationen [Abb. 1.3-5]. Die ersten

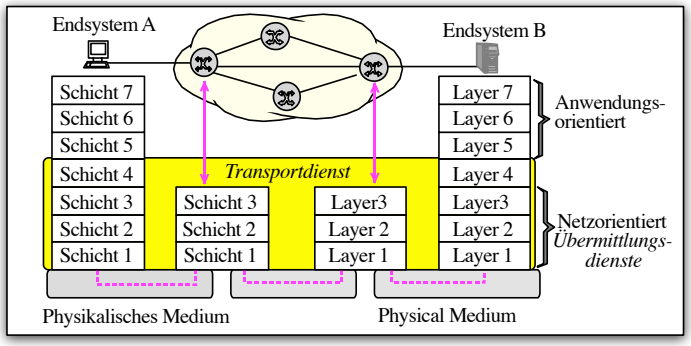


Abb. 1.3-3: Klassen von Aufgaben im OSI-Referenzmodell

drei Schichten realisieren in der Regel eine ungesicherte Übermittlung von Datenpaketen zwischen zwei Endsystemen, also z.B. über mehrere Zwischensysteme.

Schicht 4 und Schicht 2

Eine besondere Rolle hat die Schicht 4 (Transportschicht). Sie hat insbesondere die Aufgabe, die unzuverlässige Übermittlung von Datenpaketen zwischen zwei Endsystemen – als den Dienst der ersten drei Schichten – zuverlässig (fehlerfrei) zu machen. Die Aufgabe von Schicht 4 ist somit mit der Aufgabe von Schicht 2 vergleichbar. Diese beiden realisieren die Sicherung der Datenübermittlung. Schicht 2 kümmert sich um die Datenübermittlung über eine 'Leitung' und Schicht 4 kümmert sich um die Übermittlung von Daten zwischen zwei Endsystemen, die in der Regel nicht direkt (physikalisch) verbunden sind.

Transportdienst

Die ersten vier Schichten können daher als *Transportdienst* angesehen werden, der einen gesicherten Datenaustausch zwischen zwei Endsystemen garantiert. Diesen Dienst nutzen die Schichten 5, 6 und 7, die anwendungsorientiert sind.

Dienst der Schichten: Paketierung

PDU und SDU vs. Frame, Paket, Segment, Nachricht

Ein zentrales Konzept der Rechnerkommunikation ist die *Paketierung* der Daten: Die Nutzdaten der Schicht $n+1$ werden um die Kontrollinformationen der Schicht n angereichert, bzw. bei den auf Schicht n vorliegenden Datenpakete wird der *Payload* entnommen und dieser der Schicht $n+1$ übergeben. Wie in Abb. 1.3-5 gezeigt, wird dieses Konzept auf allen Schichten realisiert:

- Die zu übertragenen Daten werden in 'Pakete' variabler Länge geschnürt und mit einem *Protokollkopf* (*Header*) versehen, der das Ziel (*Destination*) und die Herkunft (*Source*) sowie die Verwendung (*Protocol-Identifier*) angibt.
- Das Gesamtpaket, also die *Nutzlast* (*Payload*) und der Protokollkopf, wird als *Protocol Data Unit* (PDU) bezeichnet, die Nutzlast als *Service Data Unit* (SDU).
- Wird zusätzlich das Gesamtpaket durch einen *Trailer* ergänzt, der Informationen zum Schutz vor Datenverfälschung enthält, wird die Bezeichnung *Frame* genutzt.

Während der Begriff *PDU* für alle Schichten des Kommunikationsmodells verwendet werden kann, deutet die Bezeichnung *Paket* an, dass Bezug auf die Netzwerkschicht genommen wird; *Frames* sind vor allem auf der Sicherungsschicht (Schicht 2) im Einsatz; die TCP-Pakete werden *Segmente* genannt und die Applikationsdaten häufig einfach

Nachrichten. Gelegentlich wird (besonders bei TLS) der Begriff *Records* genutzt, der aber Synonym zu *Frames* zu verstehen ist, sich allerdings nun nicht mehr auf die Sicherungsschicht bezieht.

Verbindungsorientiert, Verbindungslos

Verbindungslose vs. verbindungsorientierte Kommunikation

Protokollieren zwei Kommunikationsinstanzen auf der jeweiligen Schicht *Zustandsinformationen* über ihren Partner, und tauschen sich beide diese Kontrollinformation gegenseitig aus, sprechen wir von *verbindungsorientierter Kommunikation*; im anderen Falle von *verbindungsloser Kommunikation*. In Bezug auf die Schichten des Kommunikationsmodells haben sich folgende Bezeichnungen eingebürgert:

- Eine *verbindungsorientierte* Kommunikation auf den anwendungsorientierten Schichten (7 bis einschließlich 5) wird in der Regel als *Sitzung* (*Session*) bezeichnet, insbesondere dann, wenn dies die Applikation selbst betrifft.
- Auf den Paket- bzw. Frame-übertragenden Schichten 4, 3 und 2 sprechen wird in der Regel bei einer *verbindungsorientierten Übermittlung* schlichtweg von einer *Verbindung* und
- auf der Schicht 1 wird eine kontinuierlich bestehende und überwachte, physikalische Signalverbindung kurz als *Link* bezeichnet.

1.3.2 Schichtenmodell der Protokollfamilie TCP/IP

Auch die Protokollfamilie TCP/IP kann in einem Schichtenmodell dargestellt werden. Dieses Modell ist eine vereinfachte Variante des OSI-Referenzmodells, in der die anwendungsorientierten Schichten 5, 6 und 7 aus dem OSI-Referenzmodell [Abb. 1.3-3] zu einer Schicht zusammengefasst sind. Abb. 1.3-4 zeigt das Schichtenmodell der Protokollfamilie TCP/IP.

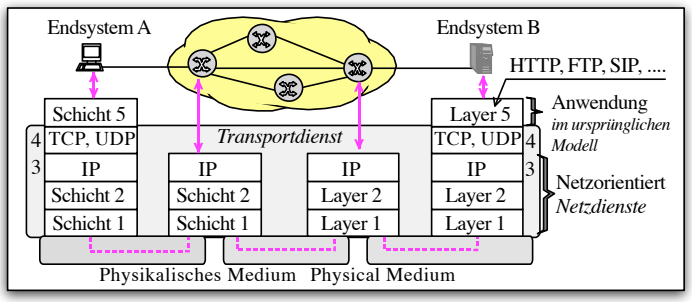


Abb. 1.3-4: Ursprüngliches Schichtenmodell der Protokollfamilie TCP/IP

Im Allgemeinen entsprechen die Funktionen der Schichten 1, 2, 3 und 4 im Schichtenmodell der Protokollfamilie TCP/IP den Funktionen der gleichen Schichten im OSI-Referenzmodell. Die Protokolle der Schichten 1 und 2 in den beiden Schichtenmodellen – d.h. von OSI und von TCP/IP – können auch identisch sein. Innerhalb der Schicht 3 im Modell von TCP/IP wird das Protokoll IP (*Internet Protocol*) angesiedelt. Innerhalb der Schicht 4 werden zwei in der Regel die Transportprotokolle TCP (*Transmission Control Protocol*) und UDP (*User Datagram Protocol*) eingesetzt. TCP ist ein verbin-

dungsorientiertes; UDP hingegen ein verbindungsloses Transportprotokoll. Als weitere Transportprotokolle fungieren SCTP (*Stream Control Transmisson Protocol*) [Abb. 3.6-1] und DCCP (*Datagram Congestion Control Protocol*). Auf die Unterschiede zwischen TCP und UDP geht Abschnitt 1.4.4 näher ein.

Schicht 5 als ursprüngliche Anwendungsschicht

Vergleicht man die Schichtenmodelle von OSI mit dem ursprünglichen Ansatz von TCP/IP, d.h. Abb. 1.3-3 und Abb. 1.3-4, stellt man fest, dass die oberen anwendungsorientierten Schichten 5, 6 und 7 aus dem OSI-Referenzmodell beim Schichtenmodell für TCP/IP zu einer Anwendungsschicht zusammengefasst sind.

Support-Protokolle

Mit dem heutigen *Internet der Dinge* (*Internet of Things*), dem *ubiquitous Computing*, den Anforderungen an die Echtzeitkommunikation, der Notwendigkeit für Verschlüsselung und den hiermit einhergehenden sehr unterschiedlichen Anwendungen ergibt sich als Anforderung für die Internetprotokolle eine ergänzende Unterstützung in Form von *Application-Support-Protokollen*, die Bedarfsweise eingesetzt werden und quasi eine zusätzliche Kommunikationsschicht darstellen. Zu diesen Protokollen zählen speziell die Protokolle TLS (*Transport Layer Security*) und DTLS (*Datagram Transport Layer Security*).

Abb. 1.3-5 zeigt der Aufbau von Daten, die zwischen den kommunizierenden Instanzen innerhalb einzelner Schichten übermittelt werden. Die Funktion der *(Application-)Support-Protokolle* lässt sich durchaus mit den Eigenschaften identifizieren, die im OSI-Modell für die *Präsentations-Schicht* vorgesehen war und die in Konsequenz zu einer Erweiterung des ursprünglichen TCP/IP-Schichtenmodells geführt hat.

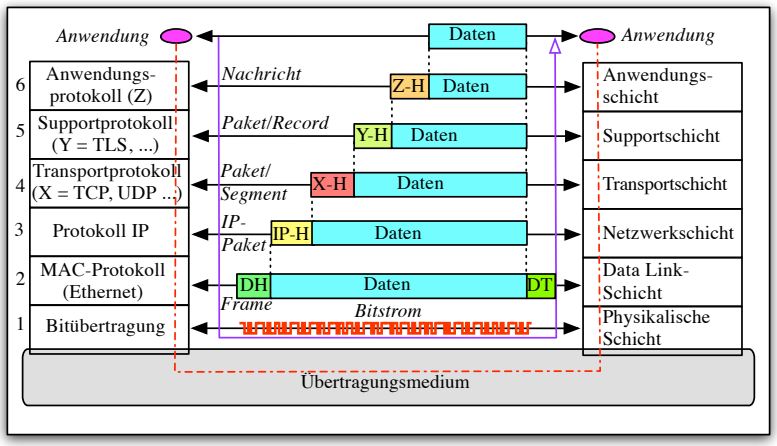


Abb. 1.3-5: Erweitertes Schichtenmodell der Protokollfamilie TCP/IP – Strukturen von zwischen den kommunizierenden Instanzen übermittelten Daten DH: Data-Link Header, DT: Data-Link Trailer

Strukturierung der übermittelten Daten

Vereinfacht kann man sich die Übermittlung von Daten zwischen zwei Anwendungen folgendermaßen vorstellen: Dem zu sendenden Datenblock `Daten` wird ein Header `Z-H` mit bestimmten Angaben des Anwendungsprotokolls `Z` (z.B. `Z` = HTTP) im Quellrechner vorangestellt. Dies stellt sicher, dass das Paar [`Z-H`, `Daten`] immer an die gleiche Instanz des Anwendungsprotokolls `Z` – nun aber im Zielrechner – übergeben wird. Bei Bedarf wird ein Protokoll der Application-Support-Schicht `Y` (`Y` = TLS oder DTLS) angefordert, um die Nutzdatenübertragung zu sichern. Hierdurch weiß der Zielrechner, wie er die-

ses Paket zu verarbeiten und ggf. zu entschlüsseln hat. Das resultierende Paket [Y-H, [Z-H, Daten]] muss nun an die Instanz des gleichen Supportprotokolls Y im Zielrechner übermittelt werden. Hierfür wird es an das Transportprotokoll X (X = TCP bzw. UDP) übergeben. Nun wird ein Header X-H des Transportprotokolls X vorangestellt, sodass eine Dateneinheit [X-H[Y-H[Z-H,Daten]]] des Transportprotokolls entsteht. Diese Dateneinheit wird nun an die IP-Instanz übergeben, wo ihr ein IP-Header (IP-H) hinzugefügt wird. So entsteht ein *IP-Paket*, das als Payload in einen *Data-Link Frame* (*DL-Frame*) eingebettet und durch den Data-Link-Header (DLH) und den Data-Link-Trailer (DLT) ergänzt wird. Dieses DL-Frame wird nun zum Zielrechner übertragen. Dort müssen die empfangenen Daten aus Schicht 1 an die Anwendung (Schicht 6) übergeben werden.

Übermittlungsvorgang

Abb. 1.3-5 zeigt den zusammengefassten Übermittlungsvorgang:

1. Quellrechner: *Vorbereitung von Daten zum Senden*

`Daten`		⇒
Anwendungsprotokolleinheit	`[Z-H,Daten]`	⇒
Supportprotokolleinheit	`[Y-H[Z-H,Daten]]`	⇒
Transportprotokolleinheit	`[X-H[Y-H[Z-H,Daten]]`	⇒
IP-Paket	`[IP-H[X-H[Y-H[Z-H,Daten]]]]`	⇒
DL-Frame	`[DLH[IP-H[X-H[Y-H[Z-H,Daten]]]]DLT]`.	

2. DL-Frame wird *bitweise* übertragen.
3. Zielrechner: *Übergabe von Daten an die Anwendung*

DL-Frame	`[DLH[IP-H[X-H[Y-H[Z-H,Daten]]]]DLT]`	⇒
IP-Paket	`[IP-H[X-H[Y-H[Z-H,Daten]]]]`	⇒
Transportprotokolleinheit	`[X-H[Y-H[Z-H,Daten]]]`	⇒
Supportprotokolleinheit	`[Y-H[Z-H,Daten]]`	⇒
Anwendungsprotokolleinheit	`[Z-H,Daten]`	⇒`Daten`.

Bemerkung: Abb. 1.4-2 illustriert eine vereinfachte Situation, bei der die zu sendenden Datenmenge so groß ist, dass man sie nicht in einem IP-Paket übermitteln kann. Hier kommt TCP zum Einsatz, und die Daten werden auf mehrere IP-Pakete aufgeteilt. Man spricht hierbei von *Segmentierung der Daten*. Ein IP-Paket enthält damit ein Datensegment.

1.4 Allgemeine Prinzipien der IP-Kommunikation

Die wichtigen Prinzipien der Kommunikation in IP-Netzen können weitgehend aus dem in Anschnitt 1.3.2 dargestellten Schichtenmodell abgeleitet werden. Hierbei spielen die Schichten *Netzwerkschicht* mit dem Protokoll IP und *Transportschicht* mit den Protokollen TCP und UDP eine dominierende Rolle. Bevor auf diese beiden Schichten eingegangen wird, wird zunächst die Bildung von IP-Paketen kurz vorgestellt.

1.4.1 Bildung von IP-Paketen

Nutzung von UDP

Bei der Bildung von IP-Paketen ist zu unterscheiden, ob TCP oder UDP als Transportprotokoll eingesetzt wird. Beim Einsatz des verbindungslosen Transportprotokolls UDP

werden die Daten bzw. eine Nachricht einer Anwendung – als Nutzlast – um den UDP-Header ergänzt, sodass eine UDP-Dateneinheit entsteht. Wie Abb. 1.4-1 zeigt, wird aus jeder UDP-Dateneinheit durch das Voranstellen eines IP-Header ein *IP-Paket* gebildet. Da die IP-Pakete keine Angaben zur Synchronisation enthalten, um sie auf der Leitung zu 'markieren', müssen sie in *Data-Link Frames* (*DL-Frames*) eingebettet werden.

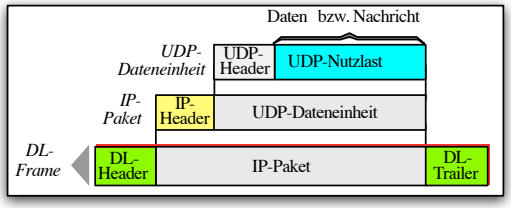


Abb. 1.4-1: Kapselung der Nutzlast beim UDP-Einsatz

MAC-Frames in LANs

In LANs bildet die sog. MAC-Funktion (*Media Access Control*) den Kern der Data-Link-Schicht. Wird ein IP-Paket in einem LAN übermittelt, wird es in einen MAC-Frame eingebettet. Bei der Übermittlung der IP-Pakete über eine Leitung bzw. über eine Punkt-zu-Punkt-Verbindung wird innerhalb der Schicht 2 häufig das Protokoll PPP (*Point-to-Point Protocol*) verwendet. In diesem Fall stellen die DL-Frames *PPP-Frames* dar (siehe Abschnitt 12.2).

Bedeutung von DL-Frames

Jedes zu übertragende IP-Paket muss immer in einen DL-Frame eingebettet werden. Dies bedeutet, dass jedem IP-Paket ein DL-Header vorangestellt wird und nach dem Ende des IP-Pakets folgt ein DL-Trailer. Diese beiden enthalten bestimmte *Synchronisationsangaben* (oft die Bitfolge `01111110`), um den Beginn und das Ende des DL-Frames auf einer Leitung zu erkennen. Abb. 1.4-2 illustriert, wie die IP-Pakete aus den Daten bzw. aus der langen Nachricht eines Anwendungsprotokolls bei der Nutzung des verbindungsorientierten Transportprotokolls TCP gebildet werden.

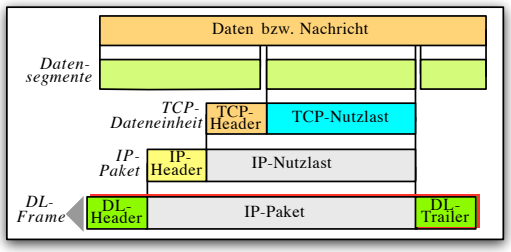


Abb. 1.4-2: Verkapselung der Nutzlast beim TCP-Einsatz

Anders als bei UDP entstehen aus den zu übermittelnden Daten bei TCP mehrere *Datensegmente*. Jedes Datensegment wird dann um einen TCP-Header erweitert, sodass eine TCP-Dateneinheit entsteht. Aus jeder TCP-Dateneinheit wird im nächsten Schritt ein IP-Paket gebildet. Zum Senden wird das IP-Paket in einen DL-Frame eingekapselt.

Wie aus Abb. 1.4-1 und Abb. 1.4-2 ersichtlich ist, werden die IP-Pakete zum Senden immer in entsprechende DL-Frames der zweiten Schicht eingekapselt, die vom Übermittlungsnetz abhängig sind. Erst in einem DL-Frame kann ein IP-Paket über ein physikalisches Netz gesendet werden.

1.4.2 Netzwerkschicht in IP-Netzen

Arten der Netzwerkschicht

Die Netzwerkschicht in IP-Netzen hat die Aufgabe, die Daten in Form von IP-Paketen zwischen Endsystemen zu übermitteln. Hierbei unterscheidet man zwischen der *verbindungslosen* und der *verbindungsorientierten* Netzwerkschicht:

Verbindungslos

- Wird *keine* Route über das Netz für einen Strom der von einem Quellrechner zu einem Zielrechner zu übermittelnden IP-Pakete festgelegt, sondern jedes einzelne Paket aus diesem Strom nach einem eigenen Weg über das Netz zum Zielrechner übermittelt, handelt es sich um die *verbindungslose Netzwerkschicht*.

Verbindungs-orientiert

- Wird *eine* Route über das Netz für einen Strom der von einem Quellrechner zu einem Zielrechner zu übermittelnden IP-Pakete festgelegt und werden alle Pakete aus diesem Strom nach dem gleichen Weg über das Netz, der eine logische Verbindung darstellt, zum Zielrechner übermittelt, handelt es sich um die *verbindungsorientierte Netzwerkschicht*.

Verbindungslose Netzwerkschicht

Verbindungslose Netzwerkschicht

Die verbindungslose Netzwerkschicht bedeutet, dass die Vermittlungsnetzknoten im IP-Netz die Router darstellen und die einzelnen IP-Pakete als *Datagrams* voneinander unabhängig über das Netz übermittelt werden. Diese Übermittlungsart entspricht dem Versand von Briefen bei der Post. Jedes IP-Paket kann daher mit einem Brief verglichen werden. Der Router würde einer Briefverteilungsstelle entsprechen. Abb. 1.4-3 illustriert die Struktur der verbindungslosen Netzwerkschicht in IP-Netzen.

Interpretation der IP-Adresse

Die ersten drei unten liegenden Schichten realisieren also beim Einsatz von Routern einen *verbindungslosen Übermittlungsdienst*. Dieser entspricht dem Briefpostdienst und eine IP-Adresse ist mit einer postalischen Adresse vergleichbar. Die IP-Adresse stellt auch einen Zugangspunkt zum Dienst für die Übermittlung der IP-Pakete dar und ist oberhalb der Schicht 3 – also an der Grenze zu Schicht 4 – anzusiedeln.

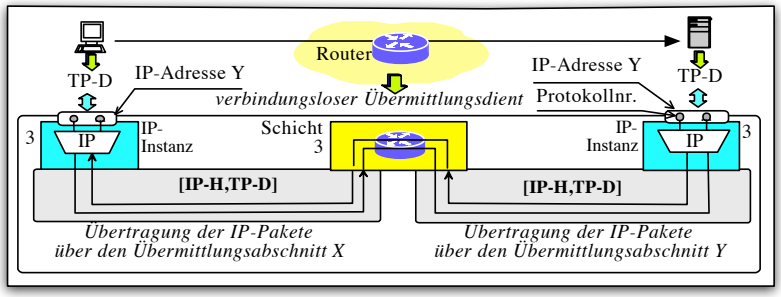


Abb. 1.4-3: Struktur der verbindungslosen Netzwerkschicht in IP-Netzen
IT-H: IP-Header, TP-D: Transportprotokolldateneinheit

Die IP-Instanz kann als ein IP-Multiplexer angesehen werden. Zwischen den IP-Instanzen werden die IP-Pakete übermittelt. Jedes IP-Paket setzt sich aus einem IP-Header `IP` und einer Transportprotokolldateneinheit `TP-D` zusammen, d.h, es hat die Struktur `[IP-H,TP-D]`.

Die Ports des IP-Multiplexers repräsentieren die Nummern der Protokolle von Schicht 4, die auf die Übermittlungsdienste direkt zugreifen können (vgl. Abb. 1.4-7 und Abb. 1.4-8). Die Protokollnummer wird im IP-Header übermittelt [Abb. 2.2-1] und informiert, von

welchem Protokoll die Dateneinheit im IP-Paket stammt. Jedem Protokoll der Schicht 4 wird daher von der IANA (*Internet Assigned Numbers Authority*) eine feste und weltweit eindeutige Nummer zugewiesen.

Verbindungsorientierte Netzwerkschicht

Verbindungsorientierte Netzwerkschicht

Der Einsatz von MPLS (*Multi-Protocol Label Switching*) bzw. von GMPLS (*Generalized MPLS*) führt zur verbindungsorientierten Netzwerkschicht in IP-Netzen [Kapitel 11]. In diesem Fall fungieren die *(G)MPLS-Switches* als Vermittlungsnetzknoten. Bei der verbindungsorientierten Netzwerkschicht wird zuerst eine Route über das Netz für die Übermittlung eines Stroms der IP- Pakete festgelegt und danach werden alle IP-Pakete aus diesem Strom als 'Gänsemarsch' über das Netz vom Quellrechner zum Zielrechner übermittelt. Diese Übermittlungsart wird heute hauptsächlich in IP-Netzen von großen Netzdienstanbietern realisiert.

Abb. 1.4-4 zeigt die Struktur der verbindungsorientierten Netzwerkschicht in IP-Netzen beim MPLS-Einsatz.

MPLS-Multiplexer

Die ersten drei unten liegenden Schichten realisieren beim MPLS-Einsatz einen verbindungsorientierten Übermittlungsdienst. Die IP-Adresse stellt einen Zugangspunkt zu diesem Dienst dar. Die IP-Instanz enthält hier – im Vergleich zur IP-Instanz in Abb. 1.4-3 – zusätzlich einen *MPLS-Multiplexer*.

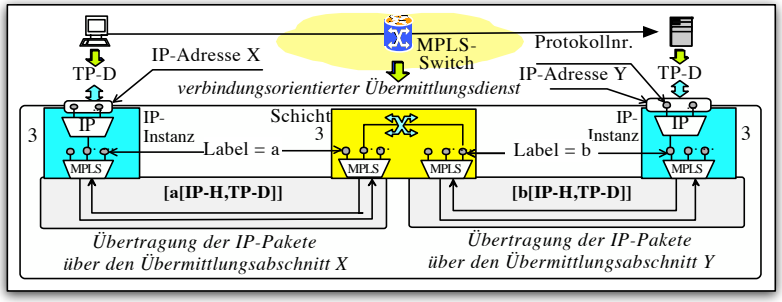


Abb. 1.4-4: Struktur der verbindungsorientierten Netzwerkschicht in IP-Netzen
IT-H: IP-Header, TP-D: Transportprotokolldateneinheit

Virtuelle Verbindung

Einem Strom von IP-Paketen wird ein Port im MPLS-Multiplexer zugeordnet. Somit können mehrere Datenströme parallel übermittelt werden. Die Portnmmern des MPLS-Multiplexers stellen die *Labels* dar. Ein Label wird immer den zu übermittelnden IP-Paketen eines Stroms vorangestellt. Abb. 1.4-4 bringt dies zum Ausdruck. Ein Label informiert, von welchem Port im MPLS-Multiplex ein IP-Paket stammt bzw. welchem Port es übergeben werden muss. Ein MPLS-Switch leitet – im Allgemeinen – ein empfangenes IP-Paket nach einer Switching-Tabelle von einem Port zu einem anderen zum Außenden weiter. Daher kann ein anderes Label den IP-Paketen eines Stroms auf einem anderen Übermittlungsabschnitt vorangestellt werden. Zwischen den Ports im MPLS-Multiplexer entsteht entsprechend im Quell- und im Zielrechner eine logische Verknüpfung, die als *virtuelle (logische) Verbindung* interpretiert wird.

1.4.3 Verbindungslose IP-Kommunikation im Internet

Nachbildung des Briefdienstes

Das Internet stellt eine weltweite Kopplung von physikalischen Netzen dar, in denen das Protokoll IP eingesetzt wird. Somit kann das Internet als heterogenes IP-Netz angesehen werden. Als IP-Netz setzt sich das Internet aus einer Vielzahl von IP-Subnetzen zusammen, die mit Hilfe von Routern miteinander vernetzt sind. Daher ist die Netzwerkschicht im heutigen Internet verbindungslos [Abb. 1.4-3]. Ein Router leitet jedes empfangene IP-Paket unabhängig von der aktuellen Lage im Netz und von anderen Paketen weiter.

Abb. 1.4-5 illustriert das Prinzip der Kommunikation im Internet an einem Beispiel, in dem eine Folge von TCP-Dateneinheiten gesendet wird. Jede dieser Dateneinheiten wird als ein IP-Paket gesendet. Im Zielrechner setzt TCP die in den IP-Paketen empfangenen Daten wieder zusammen. Gehen einige TCP-Dateneinheiten bei der Übertragung verloren bzw. werden sie verfälscht, so fordert TCP im Zielrechner vom Quellrechner eine wiederholte Übertragung an [Abschnitt 3.3].

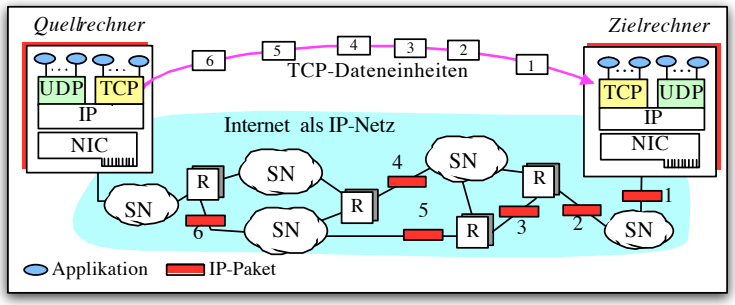


Abb. 1.4-5: Prinzip der Kommunikation im Internet – Datagramm-Prinzip
R: Router, SN: IP-Subnetz, NIC: Network Interface Card (Controller)

IP-Pakete wie Briefe

Beim Einsatz von Routern werden die IP-Pakete als *Datagrams* (also wie Briefe) unabhängig voneinander zum Zielrechner gesendet. Die wichtigsten Angaben in IP-Paketen sind die IP-Adressen von Quell- und Zielrechner. Da die einzelnen IP-Pakete unabhängig voneinander abgeschickt werden, können sie am Ziel in einer anderen Reihenfolge ankommen, als sie abgeschickt wurden. Für die Wiederherstellung von Daten aus so empfangenen IP-Paketen ist TCP verantwortlich.

Bedeutung von TTL

Da die IP-Pakete im Netz zirkulieren können, ist es nötig, ihre Verweilzeit im Netz zu kontrollieren. Der Quellrechner gibt als TTL-Angabe (*Time To Live*) im IP-Header [Abb. 2.2-1] an, wie lange das IP-Paket im Netz verweilen darf. Weil der TTL-Wert in jedem Router um 1 verringert wird, ist er identisch mit der maximalen Anzahl von Routern, die ein IP-Paket durchlaufen darf. Fällt der TTL-Wert auf 0, wird das IP-Paket im Router verworfen. Der Quellrechner wird dann mit einer Meldung des Protokolls ICMP (*Internet Control Message Protocol*) darüber informiert.

1.4.4 Transportschicht in IP-Netzen

Interpretation der IP-Adresse

Um die Bedeutung der Transportschicht in IP-Netzen näher zu erläutern, zeigt Abb. 1.4-6 die vereinfachte Struktur von Rechnern am IP-Netz. Die IP-Adresse eines Rechners kann einem Kommunikationspuffer zugeordnet werden, der einen Zugangsport zum Protokoll

IP darstellt. Dieser Kommunikationspuffer befindet sich an der Grenze zwischen der Schicht 3 mit dem Protokoll IP und der Schicht 4 mit den Transportprotokollen TCP und UDP.

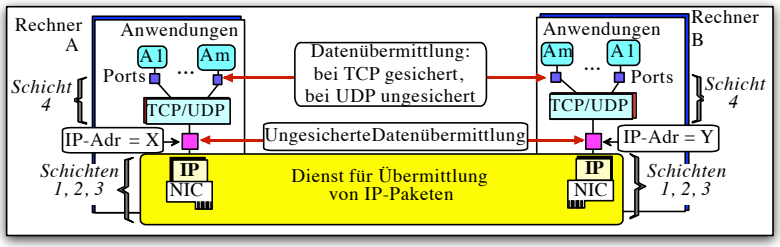


Abb. 1.4-6: Vereinfachte Struktur von Rechnern am IP-Netz
A: Applikation, Adr: Adresse, NIC: Network Interface Controller

Die drei Schichten 1, 2 und 3 stellen einen Dienst für die Übermittlung der IP-Pakete zwischen den Rechnern zur Verfügung. Es handelt sich hier um eine ungesicherte Übermittlung von IP-Paketen zwischen IP-Adressen. Eine IP-Adresse stellt einen Zugangspunkt zu diesem Übermittlungsdienst für die Protokolle TCP und UDP der Transportschicht (Schicht 4) dar.

Arten der Kommunikation

Die Transportschicht regelt den Verlauf der Datenübermittlung zwischen Anwendungen – genauer gesagt zwischen Ports dieser Anwendungen – in verschiedenen Rechnern. Hierbei sind zwei Arten der Kommunikation zu unterscheiden:

- *verbindungslose* Kommunikation beim UDP-Einsatz,
- *verbindungsorientierte* Kommunikation beim TCP-Einsatz

UDP-Multiplexer

Abb. 1.4-7 zeigt die Transportschicht mit UDP. Eine UDP-Instanz kann als UDP-Multiplexer angesehen werden. Die Eingangsports zu diesem Multiplexer stellen die Kommunikationspuffer einzelner UDP-Anwendungen dar, die kurz als *Ports* bezeichnet werden. Der Ausgangsport des UDP-Multiplexers führt zu einer IP-Adresse. Damit können mehrere UDP-Anwendungen parallel auf den Dienst für die Übermittlung der IP-Pakete zugreifen.

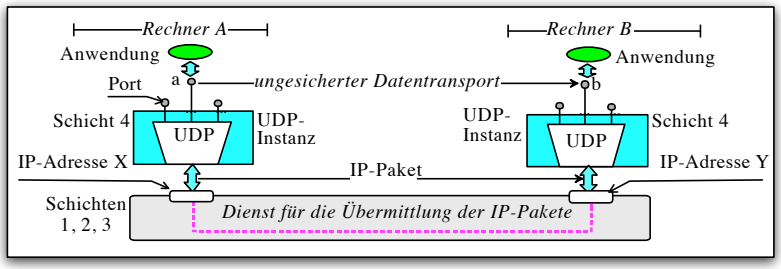


Abb. 1.4-7: Transportschicht mit UDP; ungesicherter Datentransport

Verbindungslose Kommunikation

Beim UDP-Einsatz ist die Kommunikation zwischen zwei Anwendungen verbindungslos, d.h, es wird keine Vereinbarung über den Verlauf der Kommunikation zwischen ihnen getroffen. Der Quellrechner als Initiator der Kommunikation übermittelt ein UDP-Paket an den Zielrechner, ohne ihn zu 'fragen', ob er in der Lage ist, dieses Paket zu empfan-

gen. Bei derartiger Kommunikation findet daher keine Fehler- und Flusskontrolle statt [Abschnitt 1.2].

Verbindungs-orientierte Kommunikation

Bei der verbindungsorientierten Kommunikation zwischen zwei Anwendungen beim TCP-Einsatz vereinbaren die beiden kommunizierenden Rechner zuerst, wie die Kommunikation zwischen ihnen verlaufen soll, d.h. wie die zu übertragenden Daten zu nummerieren sind und wie die Fehler- und die Flusskontrolle ablaufen sollen. Eine Vereinbarung zwischen zwei Rechnern in Bezug auf den Verlauf der Kommunikation zwischen ihnen wird als TCP-Verbindung bezeichnet [Abb. 1.4-8].

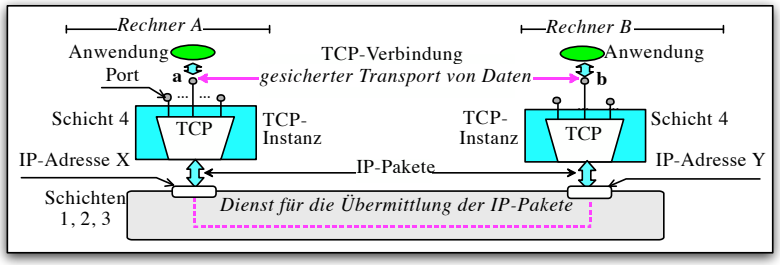


Abb. 1.4-8: Transportschicht mit TCP; gesicherter Datentransport

TCP-Multiplexer

Eine TCP-Instanz ist auch ein TCP-Multiplexer. Die Eingangsports zu diesem Multiplexer stellen die Ports einzelner TCP-Anwendungen dar. Der Ausgangsport des TCP-Multiplexers führt wie bei UDP zu einer IP-Adresse, sodass mehrere TCP-Anwendungen parallel auf den Übermittlungsdienst für IP-Pakete zugreifen können.

Well-known Ports

Die TCP- und UDP-Anwendungen wie z.B. HTTP, FTP bzw. SIP sind feste Standardanwendungen, die unter den allgemein bekannten und weltweit eindeutigen Portnummern (in Zielrechnern!) erreichbar sind. Eine derartige Nummer wird in der TCP/IP-Welt als *Well-known Port* bezeichnet. Eine Zusammenstellung von Standardanwendungen und deren Portnummern kann in UNIX-Rechnern in der Datei `/etc/services` eingesehen werden. Unter der Adresse `http://www.iana.org/assignments/port-numbers` befindet sich die Auflistung aller Well-known Ports.

Lokation von Anwendungen

Um eine TCP- und eine UDP-Anwendung eindeutig weltweit zu lokalisieren, muss man Folgendes angeben:

- auf welchem Rechner die Anwendung läuft; das bestimmt eindeutig die IP-Adresse des Rechners.
- auf welchen Port im UDP- bzw. TCP-Multiplexer die Anwendung zugreift; das bestimmt die UTP- bzw. TCP-Portnummer.

Bedeutung von Socket

Eine TCP- und UDP-Anwendung lokalisiert man daher durch die Angabe (IP-Adresse, Port). Dieses Paar hat eine fundamentale Bedeutung bei der Rechnerkommunikation und wird als *Socket* bezeichnet. Die Rechnerkommunikation bei TCP/IP kann mit Hilfe von Sockets sehr anschaulich dargestellt werden. Abb. 1.4-9 illustriert dies.

Socket als Software-Steckdose

Sockets dienen somit als Zugangspunkte zu einer Wolke, die ein IP-Netz bzw. das ganze Internet repräsentiert. Ein Socket kann auch als 'Software-Steckdose' für den Anschluss einer Anwendung an das IP-Netz angesehen werden. Jedem Socket steht im Rechner ein reservierter Speicherplatz als Kommunikationspuffer zur Verfügung. Die zu übertragenden und zu empfangenden Daten einer Anwendung werden jeweils in dem für das Socket

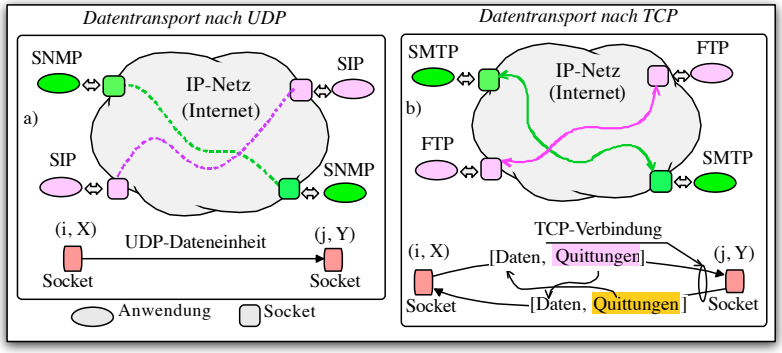


Abb. 1.4-9: Datentransport zwischen Anwendungen: a) beim UDP-Einsatz, b) beim TCP-Einsatz

reservierten Kommunikationspuffer abgelegt. Sockets sind somit auf die Zeitdauer der Verbindung beschränkt.

Wie Abb. 1.4-9a zeigt, wird bei UDP keine Verknüpfung von Sockets hergestellt, sondern eine UDP-Dateneinheit direkt an den Zielrechner gesendet und ihr Empfang vom Zielrechner nicht bestätigt.

TCP-Verbindung

Bei TCP hingegen [Abb. 1.4-9b] vereinbaren die zwei Rechner, wie der Verlauf des Datentransports zwischen den Sockets geregelt werden soll. Damit wird zwischen beiden Sockets eine *logische Verknüpfung* hergestellt, die eine *TCP-Verbindung* darstellt. Ein Socket bei TCP ist auch ein Endpunkt einer TCP-Verbindung. Eine TCP-Verbindung ist *vollduplex* und setzt sich aus zwei entgegen gerichteten, unidirektionalen Verbindungen zusammen. Eine TCP-Verbindung kann somit als 'zweispurige virtuelle Straße' über ein IP-Netz verstanden werden, über die ein gesicherter Datentransport erfolgt indem die empfangenen Daten quittiert werden [Abschnitt 3.3].

1.4.5 Multiplexmodell der Protokollfamilie TCP/IP

Nach der Beschreibung der einzelnen Schichten im Schichtenmodell für TCP/IP soll jetzt die Adressierung in IP-Netzen näher dargestellt werden. Abb. 1.4-10 zeigt ein Multiplexmodell der Protokollfamilie TCP/IP, falls ein IP-Netz auf LAN-Basis, z.B. auf Ethernet-Basis, aufgebaut wird.

Hier soll u.a. gezeigt werden, dass alle Schichten von `1` bis `n-1` einen Übermittlungsdienst der Schicht `n` zur Verfügung stellen. Die Schicht `1` stellt einen Dienst für die Übermittlung der Bitströme zur Verfügung. Der Zugang zu diesem Dienst erfolgt über physikalische Interfaces.

Interpretation der MAC-Adresse

Ein Rechner am LAN enthält normalerweise eine LAN-Adapterkarte, die zusammen mit einem Treiber u.a. die Funktion eines Multiplexers realisiert [Abb. 1.4-10]. Die Ports in diesem *LAN-Multiplexer* repräsentieren die Nummern der Protokolle von Schicht 3 [Abschnitt 1.3]. Die Nummer von IP ist beispielsweise `2048` (dezimal), bzw. `0x800`

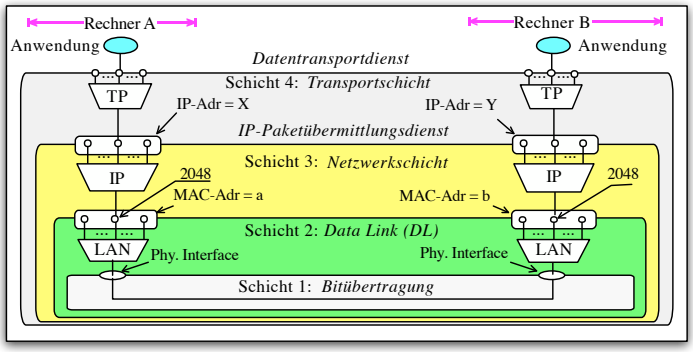


Abb. 1.4-10: Multiplexmodell der Protokollfamilie TCP/IP beim IP-Netz auf LAN-Basis
TP: UDP bzw. TCP

hexademizmal[2] Jeder Rechner am LAN ist unter einer *MAC-Adresse* erreichbar. Sie ist an der Grenze zwischen Schicht 2 und 3 anzusiedeln und kann auch als Zugangspunkt zum Dienst der Schicht 2 interpretiert werden. Über eine MAC-Adresse können daher verschiedene Protokolle der Schicht 3 auf diesen Dienst – also auf den LAN-Dienst – zugreifen.

IP-Multiplexer

Logisch gesehen wird die IP-Protokollinstanz aus der Schicht 3, die als *IP-Multiplexer* interpretiert werden kann (vgl. Abb. 1.4-3 und Abb. 1.4-4), an den Port 2048 im LAN-Multiplexer angebunden. Ein Port im IP-Multiplexer repräsentiert die Nummer eines Protokolls der Transportschicht. Schicht 3 stellt einen Dienst für die Übermittlung der IP-Pakete zwischen entfernten Rechnern. Eine IP-Adresse kann als Zugangspunkt zu diesem Dienst betrachtet werden, und über sie können mehrere Protokolle der Transportschicht diesen Dienst nutzen. Die Instanzen der Transportprotokolle TCP bzw. UDP realisieren ebenfalls die Multiplexfunktion [Abb. 1.4-7 und Abb. 1.4-8]. Daher können mehrere TCP- bzw. UDP-Anwendungen über eine IP-Adresse auf die Dienste für die Übermittlung der IP-Pakete zugreifen.

1.5 Komponenten der Protokollfamilie TCP/IP

Nach der Darstellung der Kommunikationsprinzipien bei TCP/IP anhand des Schichtenmodells soll nun gezeigt werden, welche Protokolle den einzelnen Schichten zuzuordnen sind und wie sie kooperieren. Abb. 1.5-1 zeigt die Protokollfamilie TCP/IP beim klassischen Protokoll IP, d.h. IP in Version 4 (IPv4); wobei sich eine vergleichbare Darstellung für IPv6 in Abb. 9.1-1 findet.

Wie hier gezeigt wurde, besteht die Protokollfamilie TCP/IP nicht nur aus den Protokollen TCP und IP, sondern enthält eine Reihe weiterer Protokolle, die den Schichten *Netzwerkschicht*, *Transportschicht*, *Supportschicht* und *Anwendungsschicht* im erweiterten Schichtenmodell für TCP/IP [Abb. 1.3-5] zugeordnet werden können.

[2] Bei Ethernet-Frames erfolgt die Angabe dieses *EtherType* [`http://www.iana.org/assignments/ieee-802-numbers/ieee-802-numbers.xhtml`] im MAC-Header unmittelbar vor dem eigentlichen Payload.

3 Transportprotokolle TCP, UDP und SCTP

Notwendigkeit der Transportprotokolle

Das Protokoll IP garantiert nicht die zuverlässige Übermittlung von Daten zwischen den Endsystemen. Deshalb sind Funktionen notwendig, um Verluste bzw. Verfälschungen der in Form von IP-Paketen übertragenen Daten zu entdecken und eventuell zu veranlassen, sodass der Quellrechner deren Übermittlung wiederholt. Diese Funktionen gehören der Transportschicht an, die Protokolle dieser Schicht werden *Transportprotokolle* genannt. Die TCP/IP-Protokollfamilie beinhaltet zwei klassische Transportprotokolle: UDP (*User Datagram Protocol*) und TCP (*Transmission Control Protocol*). Seit Oktober 2000 steht SCTP (*Stream Control Transmission Protocol*) als zusätzliches Transportprotokoll zu Verfügung, in dem versucht wurde, die Vorteile von UDP und TCP in Bezug auf die Übermittlung von Daten und Nachrichtenströmen zu vereinen.

Unterschiede zwischen UDP, TCP und SCTP

UDP stellt einen verbindungslosen und unzuverlässigen Dienst dar, mit dem voneinander unabhängige Nachrichten bzw. digitalisierte Echtzeitmedien wie Sprache, Audio und Video zwischen der Datenquelle (dem sendenden Rechner) und dem Datenziel (dem Empfänger) übermittelt werden. Zwischen den Kommunikationspartnern wird beim UDP keinerlei Vereinbarung hinsichtlich des Verlaufs der Kommunikation getroffen. Dagegen sind TCP und SCTP verbindungsorientierte Transportprotokolle. Der *Ablauf* der Kommunikation bei TCP bzw. bei SCTP ist durch das Protokoll geregelt, sodass sich die kommunizierenden Rechner gegenseitig laufend über den *Zustand* der Kommunikation und des Datenaustauschs verständigen. Hierdurch wird zwischen den Kommunikationspartnern eine *virtuelle Verbindung* etabliert und gepflegt.

Überblick über das Kapitel

Nach einer kurzen Erläuterung der Aufgaben der Transportschicht in Abschnitt 3.1 beschreibt Abschnitt 3.2 das Konzept und den Einsatz von UDP. Die an die Anforderungen der Echtzeitkommunikation angepasste und neue Version von UDP wird als *UDP-Lite* bezeichnet, worauf wir ebenfalls eingehen. Die Funktionen von TCP, insbesondere die Fehlerkontrolle und Flusskontrolle, erläutert Abschnitt 3.3 und die Implementierungsaspekte von TCP präsentiert Abschnitt 3.4. Die Überlastkontrolle bei TCP nach dem Prinzip ECN (*Explicit Congestion Notification*) stellt Abschnitt 3.5 dar. Auf das Konzept und den Einsatz von SCTP geht Abschnitt 3.6 ein.

Ziel dieses Kapitels

In diesem Kapitel werden u.a. folgende Fragen beantwortet:

- Welche Aufgaben haben die Transportprotokolle UDP, TCP und SCTP?
- Warum wurde UDP-Lite entwickelt und wann wird es eingesetzt?
- Wie werden die TCP-Pakete aufgebaut und welche Steuerungsangaben enthalten sie?
- Wie wird eine TCP-Verbindung auf- und abgebaut sowie wie verläuft die Fehler- und Flusskontrolle nach TCP?
- Warum ist eine Überlastkontrolle bei TCP notwendig, worin besteht deren Idee und wie kann sie realisiert werden?
- Welche Funktionen stellt das Transportprotokoll SCTP zur Verfügung?

3.1 Grundlagen der Transportprotokolle

Wozu Transportschicht?

Wir greifen die bereits in Abschnitt 1.4.2 gezeigte Darstellung des Transports von IP-Paketen auf und betrachten die ersten drei Schichten in einem IP-Netz als *IP-Übermittlungsdienst*. Da dieser Dienst über keine Mechanismen verfügt, um u.a. Verluste der übertragenen Daten zu entdecken und zu veranlassen, dass der Quellrechner die Übermittlung wiederholt, ist die Transportschicht nötig. Damit können bestimmte Mechanismen für die Garantie der zuverlässigen Datenübermittlung zur Verfügung gestellt werden. Abb. 3.1-1 illustriert die Bedeutung der Transportschicht in IP-Netzen und die Aufgabe ihrer Protokolle.

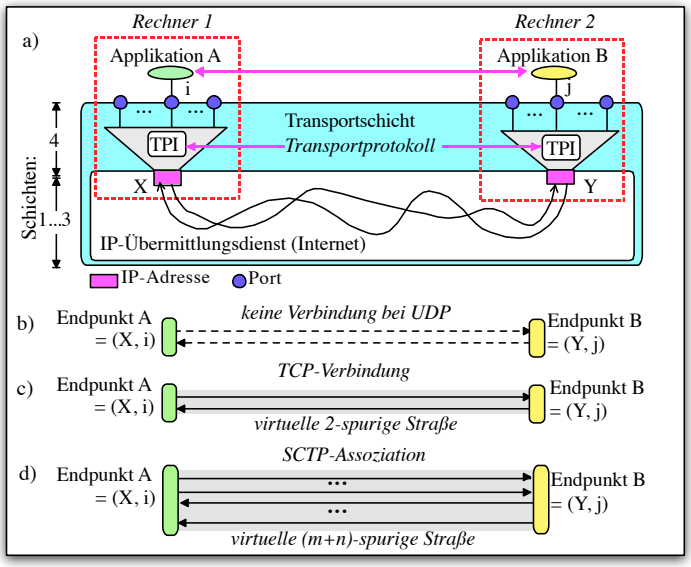


Abb. 3.1-1: Bedeutung der Transportschicht in IP-Netzen und ihre Protokolle: a) Logische Interpretation der Transportschicht, b) Kommunikation mit UDP, c) Interpretation einer TCP-Verbindung, d) Interpretation einer SCTP-Assoziation
TPI: Transportprotokollinstanz; X, Y: IP-Adressen; i, j: Ports

Sockets

Wie bereits in Abschnitt 1.4.4 [Abb. 1.4-9] gezeigt wurde, werden die Endpunkte einer Kommunikationsbeziehung als Paar (*IP-Adresse, Portnummer*) dargestellt und *Socket* genannt. Sockets können sowohl die Quelle als auch die Senke von Daten repräsentieren. Falls eine Applikationen die Daten sendet bzw. empfängt, wird sie an einen Port angebunden. Daher kann ein Socket als *Lokationsangabe* einer Applikation – d.h. auf welchem Rechner sie 'läuft' und welchen Port sie nutzt – angesehen werden.

Da mehrere Ports über eine IP-Adresse an das IP-Übermittlungsnetz 'angebunden' werden können, kann eine *Transportprotokollinstanz* als *logischer Multiplexer* angesehen werden [Abb. 1.4-7 und Abb. 1.4-8].

Transport-protokolle

Die wichtigsten Transportprotokolle in IP-Netzen sind:

- **UDP** (*User Datagram Protocol*),
- **TCP** (*Transmission Control Protocol*) und
- **SCTP** (*Stream Control Transmission Protocol*).

UDP

UDP ist ein verbindungsloses und unzuverlässiges Transportprotokoll, nach dem hauptsächlich voneinander unabhängige Nachrichten zwischen den kommunizierenden Rechnern ausgetauscht werden, ohne dass eine explizite Vereinbarung hinsichtlich des Ablaufs der Übertragung vorgenommen wird. Es wird daher keine virtuelle (logische) Verbindung zwischen den Kommunikationsinstanzen aufgebaut [Abb. 3.1-1b]. UDP wird in Abschnitt 3.2 näher dargestellt.

TCP

TCP ist ein verbindungsorientiertes und zuverlässiges Transportprotokoll. Vor dem eigentlichen Datenaustausch werden Vereinbarungen hinsichtlich des Verlaufs der Kommunikation zwischen den Rechnern getroffen. Zentraler Bestandteil der Vereinbarung sind der Aufbau, die Überwachung der Korrektheit der übertragenen Daten (Byte) und der Abbau einer virtuellen Verbindung, d.h. einer *TCP-Verbindung* [Abb. 3.1-1c], die unabhängig (d.h. vollduplex) für beide Kommunikationsrichtungen erfolgt.

SCTP

SCTP ist ein neues verbindungsorientiertes Transportprotokoll. Im Gegensatz zu TCP kann sich eine *SCTP-Verbindung*, die auch *SCTP-Assoziation* genannt wird, aus einer Vielzahl von entgegen gerichteten Kommunikationspfaden zusammensetzen [Abb. 3.1-1d]. SCTP wird in Abschnitt 3.5 detaillierter dargestellt.

Zu berücksichtigen ist hierbei, dass die Protokolle UDP, TCP und SCTP beispielsweise unabhängige Implementierungen der Transportschicht sind, sodass es hier prinzipiell keine Überschneidungen gibt.

Mehrere IP-Adressen

Ferner ist zu bedenken, dass alle aktuellen TCP/IP-Implementierungen mehrere IP-Adressen bereitstellen, und zwar die IPv4-Adressen `0.0.0.0` für die IP-Instanz (d.h. den Rechner) selbst [vgl. Tab. 2.3-2], `127.0.0.1` für das erste Netzwerk-Interface und die jeweils zugewiesenen privaten oder offiziellen IP-Adressen. Unterstützt der Rechner gleichzeitig IPv6, kommen noch die entsprechenden IPv6-Adressen hinzu.

Da die Portnummer als Angabe im TCP-, im UDP- und im SCTP-Header 16 Bit lang ist [Abb. 3.2-1, Abb. 3.3-2 und Abb. 3.6-3], kann ein Rechner pro verfügbare IP-Adresse gleichzeitig bis zu 65535 Ports organisieren. Die Ports mit den Nummern 0 bis 1023 können in der Regel nur von privilegierten, systemnahen Anwendungen benutzt werden.

Well-known Ports

Standardapplikationen (Standarddienste) wie z.B. FTP oder HTTP nutzen *Well-known Ports*, die typischerweise in der Datei `etc/services` deklariert sind. Hierdurch kann ein Client Dienste nicht nur über deren Portnummern, sondern auch über ihre zugeordneten Namen ansprechen.

Freie Ports

Die Ports mit den Nummern im Bereich von 49152 bis 65535 sind frei. Sie können den Applikationen in Rechnern zugeteilt werden und stehen insbesondere für Client-Anwendungen zur Verfügung.

Registered Ports und IANA

Die Vergabe von Portnummern und deren Zuordnung zu Applikationen wird durch die IANA koordiniert. Viele Portnummern aus dem Bereich zwischen 1024 und 49151 sind als *Registered Ports* vergeben und spezifischen Applikationen zugewiesen.

Eine aktuelle Auflistung von belegten Portnummern mit der Angabe zu jedem Port, welches Transportprotokoll der Port nutzt (UDP, TCP oder SCTP), findet man bei der IANA. Für weitere Details ist zu verweisen auf die Web-Adresse `http://www.iana.org/assignments/port-numbers`.

3.2 Konzept und Einsatz von UDP

Besonderheiten von UDP

Mit UDP wird ein einfacher verbindungsloser, unzuverlässiger Dienst zur Verfügung gestellt. UDP wurde bereits 1980 von der IETF in RFC 768 spezifiziert. Mittels UDP können Anwendungen ihre Daten als selbstständige *UDP-Pakete* senden und empfangen. UDP bietet – ähnlich wie IP – keine zuverlässige Übertragung und keine Flusskontrolle. Ergänzend kann UDP mittels einer Prüfsumme prüfen, ob die Übertragung eines UDP-Pakets fehlerfrei erfolgt ist. Enthält ein UDP-Paket einen Übertragungsfehler, wird es beim Empfänger verworfen, allerdings ohne dass der Absender des Pakets darüber informiert wird. UDP garantiert daher keine zuverlässige Kommunikation. Eine derartige Kommunikation liegt vielen Applikationen zugrunde. Hierzu gehören besonders jene, bei denen einzelne Nachrichten zwischen Rechnern ausgetauscht werden.

Einsatz von UDP

Der wichtigste Einsatz von UDP liegt in der Übermittlung von Netzwerkkonfigurationsnachrichten und unterstützt das

- DHCP (*Dynamic Host Configuration Protocol*) und
- DNS (*Domain Name System*).

Ein weiteres Einsatzgebiet von UDP besteht in der Unterstützung der Echtzeit- und Multimedia-Kommunikation (z.B. Skype):

- UDP wird vom RTP (*Real-time Transport Protocol*) für die Audio- und Video-Übermittlung und vom Signalisierungsprotokoll SIP (*Session Initiation Protocol*) verwendet.
- Bei *Voice over IP* dient UDP als primäres Transportprotokoll [Bad10].

Neben dem Protokoll RADIUS [Abschnitt 14.2] nutzen ferner die Anwendungen TFTP (*Trivial File Transfer Protocol*) und RPC/NFS (*Remote Procedure Call* beim *Network File System*) UDP als Transportprotokoll.

3.2.1 Aufbau von UDP-Paketen

Den Aufbau von UDP-Paketen zeigt Abb. 3.2-1. Der UDP-Header enthält folgende Angaben:

- `Source Port` (Quellport)
  Angabe der Nummer des Ports als Kommunikationspuffer der Applikation im Quellrechner [Abb. 3.1-1a].
- `Destination Port` (Zielport)
  Die Nummer des Ports als Identifikation der Applikation im Zielrechner.
- `UDP Length` (UDP-Paketlänge)
  Hier wird die Länge des UDP-Pakets angegeben.
- `Checksum` (Prüfsumme)
  Diese Prüfsumme ermöglicht, sowohl im UDP-Paket als auch in einigen Angaben im IP-Header, die den *IP-Pseudo-Header* bilden, Übertragungsfehler zu entdecken [Abb. 3.2-2].

Mit der Angabe `Source Port` und `Destination Port` im UDP-Header und der Angabe von `Source IP Address` und `Destination IP Address` im IP-Header werden die

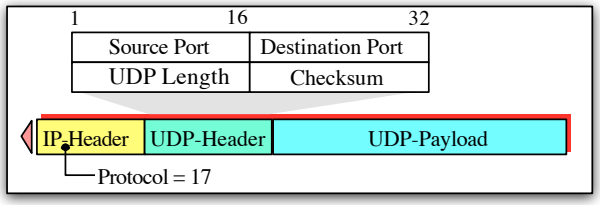


Abb. 3.2-1: Aufbau von UDP-Paketen

beiden Endpunkte (d.h. die Sockets) festgelegt, zwischen denen der Datenaustausch stattfindet [Abb. 3.1-1b].

UDP-Prüfsumme

Die Berechnung der Prüfsumme wird in RFC 1071, RFC 1141 und RFC 1624 detailliert spezifiziert, dessen Nutzung in der UDP-Spezifikation RFC 768 jedoch nicht gefordert.

IP-Pseudo-Header

Die Prüfsumme im UDP-Header deckt auch einige Angaben im IP-Header ab, die den *IP-Pseudo-Header* bilden. Abb. 3.2-2 zeigt dies.

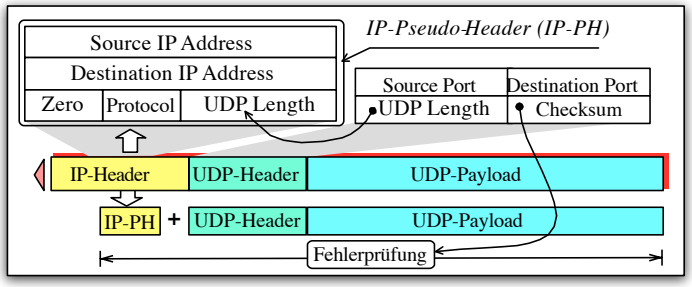


Abb. 3.2-2: Prüfsumme im UDP-Header und Angaben im IP-Pseudo-Header (IP-PH)

Die Angaben im IP-Pseudo-Header sind:

- `Source IP Address` (*IP-Quelladresse*), d.h. Quelladresse aus dem IP-Header,
- `Destination IP Address` (*IP-Zieladresse*), d.h. Zieladresse aus dem IP-Header,
- `Zero`: Der Wert `0` wird hier eingetragen.
- `Protocol`: Es wird die Protokollnummer 17 von UDP angegeben [Tab. 2.1-1].
- `UDP Length` (UDP-Paketlänge): Die Länge des UDP-Pakets wird mitgeteilt. Dieser Wert ist aus dem Feld `UDP Length` im UDP-Header übernommen.

Die Konstruktion des IP-Pseudo-Header kann nur als historisch betrachtet werden. Einerseits verstößt sie elementar gegen das Schichtenprinzip für den Kommunikationsablauf, andererseits kann sie keines der angestrebten Ziele wie beispielsweise Identifikation von 'fremden' UDP-Paketen in einer Kommunikationsbeziehung garantieren. Daher wird dieses Konstrukt bereits bei UDP-Lite aufgegeben.

UDP-Paketgröße 512 Byte

Eine zentrale Einschränkung bei der Kommunikation über UDP besteht in der Notwendigkeit, die maximale Größe der UDP-Pakete auf 512 Byte zu beschränken. Diese Anforderung entstammt nicht der Definition von UDP, sondern hat einen praktischen Hintergrund: RFC 1122 verlangt bei der IP-Kommunikation als minimale Größe des Datenpuffers 576 Byte. Zieht man den (minimal) 20 Byte großen IP-Header ab, bleiben für das UDP-Paket

556 Byte und abzüglich des UDP-Header (8 Byte) noch maximal 548 Byte für Nutzdaten. Diese historisch festgelegte Größe, die Unabhängigkeit einzelner UDP-Pakete voneinander sowie das Fehlen einer Flusskontrolle (wie bei TCP) führen dazu, dass UDP-Pakete in der Regel auf eine Größe von 512 Byte beschränkt sind.

3.2.2 Protokoll UDP-Lite

UDP verwendet man für die Übermittlung von Echtzeitdaten wie z.B. Audio oder Video über das Internet bzw. über andere IP-Netze. Im Gegensatz zur Datenkommunikation haben Bitfehler bei Audio- und Videokommunikation oft nur eine geringe negative Auswirkung auf die Qualität der Kommunikation. Ein Bitfehler bei Voice over IP ist sogar für das menschliche Ohr direkt nicht bemerkbar.

Wozu UDP-Lite?

Um die empfangenen UDP-Pakete mit einzelnen Bitfehlern in Audio- bzw. in Videodaten nicht verwerfen zu müssen und damit die Häufigkeit der Verluste von UDP-Paketen mit Audio- und Videodaten zu reduzieren, war eine Modifikation von UDP nötig. Die UDP-Fehlerkontrolle wurde so verändert, dass nur die Bereiche in UDP-Paketen überprüft werden, in denen Übertragungsbitfehler eine große Auswirkung auf die Qualität der Kommunikation haben. Dies hat zur Entstehung von UDP-Lite geführt [RFC 3828].

UDP-Lite stellt daher eine Modifikation von UDP dar. Abb. 3.2-3 zeigt den Aufbau von UDP-Lite-Paketen.

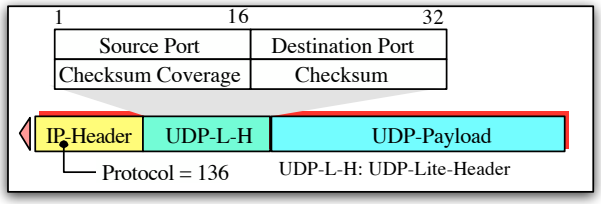


Abb. 3.2-3: Aufbau von UDP-Lite-Paketen

Der Header von UDP-Lite hat fast die gleiche Struktur wie der von UDP [Abb. 3.2-1]. Im Unterschied zu UDP enthält der Header von UDP-Lite `Checksum Coverage` anstelle der Angabe `UDP Length`. Die Nummern der *Well-known Ports* sind bei UDP-Lite die gleichen wie bei UDP. Damit können sowohl UDP als auch UDP-Lite die gleichen Applikationen nutzen.

Idee von UDP-Lite

Die Idee von UDP-Lite besteht darin, dass im Header mit `Checksum Coverage` angegeben werden kann, welcher Teil der UDP-Payload bei der Fehlerprüfung berücksichtigt wird. Abb. 3.2-4 illustriert die Idee von UDP-Lite.

Mit `Checksum Coverage` informiert der Sender den Empfänger darüber, welcher Teil von UDP-Payload durch die Prüfsumme (Checksum) abgedeckt wird. `Checksum Coverage` stellt daher die *Reichweite der Prüfsumme* dar.

Falls als UDP-Payload ein RTP-Paket übermittelt wird, kann mit `Checksum Coverage` angegeben werden, dass die Fehlerprüfung beispielsweise nur im RTP-Header stattfinden

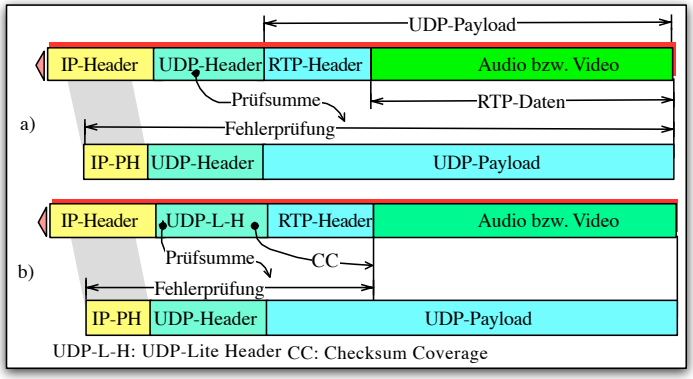


Abb. 3.2-4: Überprüfung von Übertragungsbitfehlern: a) bei UDP, b) bei UDP-Lite
IP-PH: IP-Pseudo-Header, RTP: Real-time Transport Protocol

soll. Daher deckt die Prüfsumme in diesem Fall nur den IP-Pseudo-Header, den UDP-Header und den RTP-Header ab.

Der Wert von `Checksum Coverage` wird von der Applikation auf der Sendeseite bestimmt. Diese Angabe besagt, wie viele Byte, beginnend vom ersten Byte im UDP-Lite-Header, die Prüfsumme abdeckt. Der UDP-Lite-Header muss immer mit der Prüfsumme abgedeckt werden. Ist nur der UDP-Lite-Header mit der Prüfsumme abgedeckt, wird als `Checksum Coverage` der Wert `0` eingetragen. Daher ist `Checksum Coverage` entweder gleich `0` oder größer als `8` (d.h. größer als die Anzahl von Byte im UDP-Header). Ein IP-Paket mit `Checksum Coverage` von 1 bis 7 wird einfach verworfen.

Checksum Coverage

Vergleicht man Abb.3.2-2 und Abb.3.2-4, so ist ersichtlich, dass die Angabe `UDP Length` im UDP-Header durch `Checksum Coverage` bei UDP-Lite ersetzt wurde. Dies ist möglich, da die Angabe `UDP Length` im UDP-Header redundant ist und die Länge des UDP-Pakets aus den Angaben im IP-Header (d.h. `UDP Length = Total Length minus Header Length`) berechnet werden kann.

Das Feld `Checksum` in UDP-Paketen weist lediglich eine Größe von 16 Bit auf. Dies bedeutet Einschränkungen im Hinblick auf den Umfang der erkannten Bitfehler und der möglichen Korrekturen. Daher ist es sinnvoll, nur diejenigen Daten per `Checksum` zu sichern, die von besonderer Bedeutung sind.

3.3 Funktion des Protokolls TCP

TCP ist ein verbindungsorientiertes, zuverlässiges Transportprotokoll, das bereits in RFC 793 spezifiziert wurde und seither ständig weiterentwickelt wird. Seine Bedeutung kommt in Abb. 3.3-1 zum Ausdruck. Überdies wurde hier auch verdeutlicht, dass mittels TCP mehrere Applikationen auf die Dienste von IP gleichzeitig zugreifen können. Daher kann TCP auch als *logischer Multiplexer* von Applikationen angesehen werden.

TCP als Kontrollprotokoll

Im Gegensatz zum UDP-Paket – das als reiner Datencontainer aufgefasst werden kann – beinhalten TCP-Pakete umfangreiche Steuerungsinformationen, die zum geregelten Ablauf der Verbindung eingesetzt werden, aber teilweise auch für die auf TCP aufbauenden

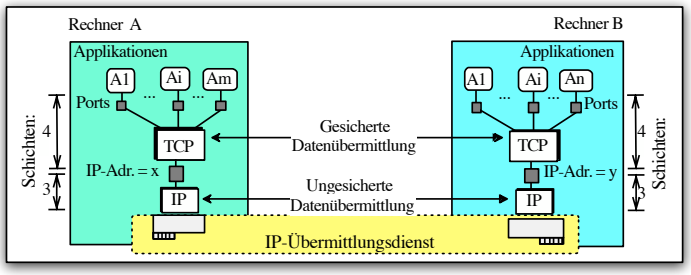


Abb. 3.3-1: TCP als Sicherungsprotokoll zwischen zwei entfernten Rechnern

Applikationen genutzt werden können. Die Steuerungsinformationen werden bei Bedarf im TCP-Header im Optionsfeld (`Options`) untergebracht. TCP hat sich im Laufe seiner Entwicklung durch den Einsatz neuer sowie die Umdefinition bestehender Optionen wie ein Chamäleon an veränderte Gegebenheiten angepasst.

TCP-Verbindungen

Die Zuverlässigkeit von TCP wird dadurch gewährleistet, indem sich die TCP-Instanzen gegenseitig über ihren jeweiligen Zustand (*Status*) informieren, also eine *virtuelle Verbindung* unterhalten. Um die Datenübermittlung nach TCP zu gewährleisten, muss daher zunächst eine *TCP-Verbindung* aufgebaut werden. Wenn der Nutzdatenfluss ausschließlich von *A* nach *B* geht, existiert immer auch eine Kontrollverbindung von *B* nach *A*. Typischerweise hat aber auch die Gegenseite etwas mitzuteilen, wodurch auch Nutzdaten von *B* nach *A* übertragen werden. In diesem Fall werden die Kontrollinformationen in den (beidseitigen) Nutzpaketen untergebracht. TCP realisiert eine *Vollduplex-Verbindung* zwischen den Kommunikationspartnern mit einem ausgeklügelten und effizienten *In-Band-Steuerungsverfahren* [Abb. 3.1-1]. Im Vergleich hierzu stellt beispielsweise ICMP für IPv4 [Abschnitt 2.7] einen Out-of-Band-Kontroll- und Fehlermechanismus bereit; bei UDP existiert hingegen gar keiner.

3.3.1 Aufbau von TCP-Paketen

Arten von TCP-Paketen

Über eine TCP-Verbindung werden die Daten in Form von festgelegten Datenblöcken – von nun an *TCP-Pakete* bzw. *TCP-PDUs* genannt – ausgetauscht. Typischerweise treten bei einer TCP-Verbindung zwei Arten von Paketen auf:

- TCP-Pakete mit Nutzdaten – als *TCP-Nutzlast* – und mit Steuerungsinformationen im TCP-Header [Abb. 1.4-2]. Zusätzlich zur eigentlichen Übermittlung von Nutzdaten sind Angaben zur Realisierung der Flusskontrolle nach dem *Sliding-Window-Prinzip* [Abb. 3.4-2] im TCP-Header dieser Pakete enthalten.
- TCP-Pakete ausschließlich mit Steuerungsinformationen. Deren Aufgabe ist das Regeln des Auf- und Abbaus von TCP-Verbindungen sowie die Überprüfung ihrer Fortdauer in Form von *Keep-Alives*, falls längere Zeit keine Datenübermittlung erfolgte.

Nummerierung von Byte

Ist eine TCP-Verbindung aufgebaut, besteht die zentrale Aufgabe von TCP darin, die zu übertragenden Daten bzw. Nachrichten vom Quellrechner *byteweise zu nummerieren* und diese in Form von *Datensegmenten* zu übermitteln. Die Nummer des ersten Nutzbyte im TCP-Datencontainer ist die laufende *Sequenznummer* und bezeichnet somit die An-

zahl aller gesendeten Nutzdatenbyte der vorigen Segmente (ohne Wiederholungen); eine Methode, die als *Byte-stream* bezeichnet wird [Abb. 3.3-8].

Flusskontrolle

Die Partnerinstanz quittiert die erhaltenen Daten, indem sie als *Quittung* (*Acknowledgement*) den letzten Wert der *Sequenznummer* (`SEQ`) plus 1 – also `SEQ + 1` – zurückgibt [Abb. 3.3-8]. Damit dieser Mechanismus funktioniert, wird vor Beginn einer Übermittlung zwischen den TCP-Instanzen im Quell- und im Zielrechner entsprechend der anliegenden Datenmenge die maximale Segmentgröße vereinbart werden (z.B. 1000 Byte). Diese und weitere Angaben werden im *TCP-Header* eingetragen.

Garantie der Reihenfolge

Die TCP-Instanz im Zielrechner setzt die empfangenen Datensegmente mittels der übertragenen Sequenznummern in der richtigen Reihenfolge in die ursprünglichen Daten zurück. Erreicht ein TCP-Paket den Zielrechner nicht, wird die Wiederholung der Übertragung der fehlenden Datensegmente durch die TCP-Partnerinstanz veranlasst.

TCP-Header

Abb. 3.3-2 zeigt den Aufbau des TCP-Header.

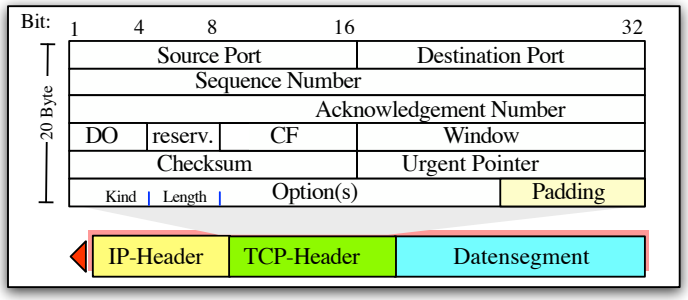


Abb. 3.3-2: Aufbau des TCP-Header
CF: Control Flags, DO: Data Offset

Die einzelnen Angaben im TCP-Header haben folgende Bedeutung:

- `Source Port` (Quellport)
  Hier wird die Nummer des Ports der Applikation im Quellrechner angegeben, die die TCP-Verbindung initialisiert hat.
- `Destination Port` (Zielport)
  Hier wird die Nummer des Ports dieser Applikation im Zielrechner angegeben, an die die Daten adressiert sind.
- `Sequence Number` (Sequenznummer)

  Bedeutung von ISN

  Die Sequenznummer bezieht sich auf den Sender und dient diesem zur Byte-weisen Nummerierung der gesendeten Daten. Beim Aufbau der TCP-Verbindung generiert jede TCP-Instanz eine Anfangssequenznummer `ISN` (*Initial Sequence Number*). Diese Nummern werden ausgetauscht und gegenseitig bestätigt [Abb. 3.3-5]. Um die gesendeten TCP-Pakete eindeutig am Zielrechner zu identifizieren, muss der Fall ausgeschlossen werden, dass sich zu einem Zeitpunkt im Netz mehrere Segmente mit der gleichen Sequenznummer befinden. Aus diesem Grund darf sich die Sequenznummer innerhalb der festgelegten *Lebenszeit* (*Time to Live*) für die IP-Pakete nicht wiederholen [Abb. 2.2-1]. Im Quellrechner wird die Sequenznummer immer jeweils um die Anzahl bereits gesendeter Byte erhöht.
  Die `ISN` wird in der Regel nicht von 0 hochgezählt, sondern bei aktuellen TCP-Implementierungen zufällig erzeugt. Hierdurch kann in gewissem Umfang gewährleistet werden, dass ein Kommunikationsendpunkt zum Zeitpunkt der Initialisierung über eine weit-

gehend einmalige ISN verfügt, die somit neben dem Socket (*IP-Adresse, Portnummer*) ein zusätzliches Verbindungsmerkmal darstellt.

- `Acknowledgement Number` (Quittungsnummer)
  Die Quittungsnummer wird vom Empfänger vergeben und dient zur Bestätigung der empfangenen Daten, indem dem Quellrechner mitgeteilt wird, bis zu welchem Byte (*Sequenznummer*) die Daten korrekt empfangen wurden.
- `Data Offset` (Datenabstand)
  Das vier Bit große Feld gibt die Länge des TCP-Header in 32-Bit-Worten an und damit die Stelle, ab der die Daten beginnen. Dieses Feld kann auch als `Header Length` interpretiert werden.
- Reserved (Reserviert)
  Diese drei Bit sind reserviert und üblicherweise auf `0` gesetzt.
- `Control Flags` (Kontrollflags)
  Die Kontrollflags legen fest, welche Felder im Header gültig sind und dienen zur Verbindungssteuerung. Zurzeit sind 9 Bit vorgesehen. Ist das entsprechende Bit gesetzt, gilt Folgendes:
  - `NS` (ECN-nonce Concealment Protection) [RFC 3540]: Unterstützung des ECN-Verfahrens [Abschnitt 3.5] mit einem Nonce-Bit.
  - `CWR` (*Congestion Window Reduced*): Einsatz bei der Überlastkontrolle nach `ECN` (*Explicit Congestion Notification*) [RFC 3168].
  - `ECE` (*ECN-Echo*): Einsatz bei der Überlastkontrolle nach ECN [RFC 3168].
  - `URG`: Der *Urgent Pointer* (Zeiger im Urgent-Feld) ist vorhanden und gültig.
  - `ACK`: Die *Quittungsnummer* ist vorhanden und gültig.
  - `PSH` (*Push-Funktion)*: Die Daten sollen sofort an die nächsthöhere Schicht weitergegeben werden, ohne nächste Segmente abzuwarten. UNIX-Implementierungen senden `PSH` immer dann, wenn sie mit dem aktuellen Segment gleichzeitig alle Daten im Sendepuffer übergeben.
  - `RST` (*Reset*): Die TCP-Verbindung soll zurückgesetzt werden.
  - `SYN`: Aufbau einer TCP-Verbindung und ist mit einem `ACK` zu quittieren [Abb. 3.3-4].
  - `FIN`: Einseitiger Abbau einer TCP-Verbindung und das Ende des Datenstroms aus dieser Richtung, das mit einem `ACK` quittiert werden muss [Abb. 3.3-5].

Window für Flusskontrolle

- `Window` (Fenstergröße)
  Diese Angabe dient der Flusskontrolle nach dem Fenstermechanismus [Abb. 3.3-6]. Das Feld `Window` gibt die Fenstergröße an, d.h. wie viele Byte – beginnend ab der Quittungsnummer – der Zielrechner in seinem Aufnahmepuffer noch aufnehmen kann. Empfängt der Quellrechner eine TCP-PDU mit der Fenstergröße gleich `0`, muss der Sendevorgang gestoppt werden. Wie die Fenstergröße die Effizienz der Übermittlung beeinflußen kann, ist aus Abb. 3.4-1bersichtlich (Senderblockade).
- `Checksum` (Prüfsumme)
  Diese Prüfsumme erlaubt es, den TCP-Header, die Daten und einen Auszug aus dem IP-Header (u.a. Quell- und IP-Zieladresse), der an die TCP-Instanz zusammen mit den Daten übergeben wird, auf das Vorhandensein von Bitfehlern zu überprüfen. Bei Berechnung der Prüfsumme wird dieses Feld selbst als null angenommen.
- `Urgent Pointer` (Urgent-Zeiger)
  TCP ermöglicht es, wichtige (dringliche) und meist kurze Nachrichten (z.B. Interrupts) den gesendeten normalen Daten hinzuzufügen und an Kommunikationspartner direkt zu übertragen. Damit können außergewöhnliche Zustände signalisiert werden. Derartige Daten

werden als *Urgent-Daten* bezeichnet. Ist der Urgent-Zeiger gültig, d.h. `URG = 1`, so zeigt er auf das Ende von Urgent-Daten im Segment. Diese werden immer direkt im Anschluss an den TCP-Header übertragen. Erst danach folgen normale Daten.

- `Options`
  TCP erlaubt es, `Service-Optionen` anzugeben. Das erste Byte in jeder `Option` legt den *Optionstyp* (*Kind*) fest, das zweite Byte die Länge (*Length*) der entsprechenden Option. Hierdurch können mehrere Optionen zugleich angegeben werden. Der experimentelle RFC 6994 geht noch einen Schritt weiter und erlaubt in den nächsten 2 Byte die Angabe einer *Experimental-ID* (`ExID`).

  Das `Options`-Feld ist entsprechend Tab. 3.3-1 zu interpretieren. In den RFCs 1323 und 2018 wurde die Bedeutung des `Options`-Feldes folgendermaßen festgelegt:

Options-Type	**Option-Feldlänge** [Byte]	**Bedeutung**
0	nicht vorgesehen	Ende der Optionsliste
1	nicht vorgesehen	No-Operation
2	4	Maximum Segment Size (MSS)
3	3	Window Scale (WSopt)
4	2	SACK erlaubt
5	variabel	SACK
8	10	TimeStamp (TSOpt)
11*)	6	Connection Count CC (bei T/TCP)
12*)	6	CC.NEW (T/TCP)
13*)	6	CC.ECHO (T/TCP)
29	≥ 4	TCP-Authentication

Tab. 3.3-1: Verwendung möglicher Optionen im `Options`-Feld des TCP-Headers `http://www.iana.org/assignments/tcp-parameters`; *) sind *historische* Optionen, die nicht mehr genutzt werden

  - `Maximum Segment Size` (`MSS`): Diese Option wird beim Aufbau einer TCP-Verbindung genutzt. Der Client teilt dem Kommunikationspartner im `<SYN>`-Paket und der Server dies `<SYN,ACK>`-Paket mit. Die gewählte, maximale Segmentgröße hängt von der `MTU` des Links ab, und es gilt:

    ```
    MSS = MTU - IP-Header-Länge - TCP-Header-Länge
    ```

  - `Window Scale` (`WSopt`): Mit `WSopt` können die Kommunikationspartner während des Aufbaus einer TCP-Verbindung (also im `SYN`-Paket) festlegen, ob die Größe des 16 Bit `Window` um einen konstanten Skalenfaktor multipliziert wird. Dieser Wert kann unabhängig für den Empfang und das Versenden von Daten ausgehandelt werden. Als Folge dessen wird nun die Fenstergröße von der TCP-Instanz nicht mehr als 16 Bit-, sondern als 32 Bit-Wert aufgefasst. Der maximale Wert für den Skalenfaktor von `WSopt` beträgt 14, was einer neuen oberen Grenze für `Window` von 1 GByte entspricht.
  - `Timestamps Option` (`TSopt`):
    Dieses Feld besteht aus den Teilen `Timestamp Wert` (`TSval`) und `Timestamp Echo Reply` (`TSecr`), die jeweils eine Länge von 4 Byte aufweisen. Eingetragen wird hier der *Tickmark*, ein interner Zähler, der pro Millisekunde um eins erhöht wird. `Timestamps Option` kann nur im ersten `SYN`-Paket angezeigt werden, das Feld ist nur bei `ACK`-PDUs erlaubt. Genutzt wird `TSopt` zur besseren Abschätzung der `RTT` (*Round Trip Time*) und zur Realisierung von `PAWS` (*Protect Against Wrapped Sequences*) [RFC 1323].
  - Mit RFC 2018 wurde das Verfahren *Selective Acknowledgement* (`SACK`) eingeführt, das sich wesentlich auf das Optionsfeld `SACK` stützt und es zulässt, dieses Feld variabel zu erweitern. Auf dieses Verfahren wird später näher eingegangen.

  - Ergänzt werden die Optionen um `Connection Count` (`CC`) sowie `CC.NEW` und `CC.ECHO`, die bei der Implementierung von *T/TCP* anzutreffen sind [Abschnitt 3.4.4].
  - Bei Verwendung von *Multipath TCP* [Abschnitt 6.5] werden noch weitere TCP-`Options` genutzt [Abb. 6.5-6].
- `Padding` (Füllzeichen)
  Die Füllzeichen ergänzen die Optionsangaben auf die Länge von 32 Bit.

TCP Timeouts

TCP verhindert den gleichzeitigen Aufbau einer TCP-Verbindung seitens der beiden Instanzen, d.h. nur eine Instanz kann den Aufbau initiieren. Des Weiteren ist es nicht möglich, einen mehrfachen Aufbau einer TCP-Verbindung durch den Sender aufgrund eines *Timeout* des ersten Verbindungsaufbauwunsches zu generieren. Der Datenaustausch zwischen zwei Stationen erfolgt erst nach dem Verbindungsaufbau. Registriert der Sender, dass der Empfänger nach Ablauf eines *Timeout* die übertragenen Daten nicht bestätigt hat, wird eine Wiederholung der nicht-quittierten Segmente gestartet. Aufgrund der Sequenznummer ist es prinzipiell möglich, $2^{32} - 1$ Datenbyte (4 Gigabyte) pro bestehende TCP-Verbindung zu übertragen, innerhalb dessen doppelt übertragene Daten erkannt werden. Bei den meisten Implementierungen ist die Sequenznummer aber als *signed Integer* deklariert, sodass nur die Hälfte, d.h. 2 Gigabyte möglich sind.

TCP-Window

Die Flusskontrolle nach dem Fenstermechanismus (*Window*) erlaubt es einem Empfänger, dem Sender mitzuteilen, wie viel Pufferplatz zum Empfang von Daten zur Verfügung steht. Ist der Empfänger zu einem bestimmten Zeitpunkt der Übertragung einer höheren Belastung ausgesetzt, signalisiert er dies dem Sender über das `Window`-Feld, sodass dieser die Senderate reduzieren kann.

3.3.2 Konzept der TCP-Verbindungen

Three-Way Handshake

Eine TCP-Verbindung wird mit dem Ziel aufgebaut, einen zuverlässigen Datenaustausch zwischen den kommunizierenden Anwendungsprozessen in entfernten Rechnern zu gewährleisten. Die TCP-Verbindungen sind vollduplex. Man kann eine TCP-Verbindung als ein Paar von gegenseitig gerichteten unidirektionalen Verbindungen zwischen zwei Sockets interpretieren [Abb. 3.1-1c]. Der Aufbau einer TCP-Verbindung erfolgt immer mittels des *Three-Way Handshake* (*3WHS*)-Verfahrens, das für eine Synchronisation der Kommunikationspartner sorgt und gewährleistet, dass die TCP-Verbindung in jede Richtung korrekt initialisiert wird. An dieser Stelle ist hervorzuheben, dass die Applikation im Quellrechner mit TCP über einen wahlfreien Port kommuniziert, der dynamisch zugewiesen wird.

TCP-Zustandsmodell

Das TCP-Modell geht von einer *Zustandsmaschine* aus. Eine TCP-Instanz befindet sich immer in einem wohldefinierten Zustand. Die Hauptzustände sind `Listen` und (Verbindung-) `Established`. Zwischen diesen stabilen Zuständen gibt es gemäß Abb. 3.3-3 eine Vielzahl zeitlich befristeter (Zwischen-)Zustände. Mittels der Kontroll-Flags `ACK`, `FIN`, `SYN` und ggf. auch `RST` wird zwischen den Kommunikationspartnern der Wechsel zu bzw. der Verbleib in einem Zustand signalisiert.

Fenstermechanismus

Um die Menge der auf einer TCP-Verbindung übertragenen Daten zwischen den kommunizierenden Rechnern entsprechend abzustimmen, was man als Flusskontrolle bezeichnet, kommt der *Fenstermechanismus* (Window) zum Einsatz. Zur effizienten Nutzung des Fenstermechanismus stehen zwei Parameter zur Verfügung, die zwischen den TCP-

Instanzen im Verlauf der Kommunikation dynamisch angepasst werden. Es handelt sich hierbei um: `Windowsize` und `Maximum Segment Size`.

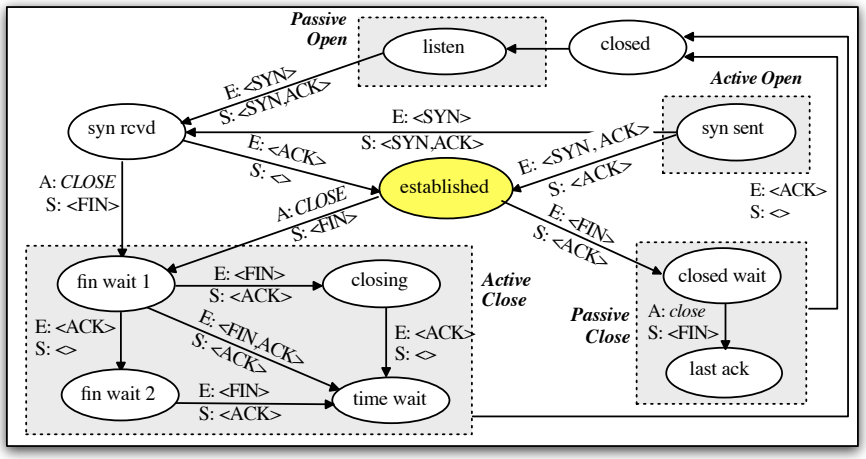


Abb. 3.3-3: TCP-Zustandsdiagramm und gestattete Übergänge zwischen den Zuständen
E: Empfänger, S: Sender, A: Applikation

Interpretation von Windowsize

`Windowsize` (`WSIZE`) ist als die Größe des TCP-Empfangspuffers in Byte zu interpretieren. Aufgrund des maximal 16 Bit großen Feldes im TCP-Header kann dieses maximal einen Wert von $2^{16} - 1 = 65535$ Byte (d.h. rund 64 KByte) aufweisen. Moderne TCP-Implementierungen nutzen allerdings die in den TCP-Header vorgesehene Option des *WSopt*, sodass nun Werte bis 2^{30}, also rund 1 GByte, möglich sind.

`Windowsize` als Konfigurationsparameter der TCP-Implementierung ist üblicherweise auf einen Wert von 4, 8, 16 oder 32 KByte initialisiert. Beim Verbindungsaufbau teilt die TCP-Empfängerinstanz dem Sender ihre *Windowsize* mit, was als *advertised Windowsize* (*advWind*) bezeichnet wird.

Interpretation von Maximum Segment Size

`Maximum Segment Size` (`MSS`) stellt das Gegenstück zu `WSIZE` dar, ist also die Größe des TCP-Sendepuffers für die zu übertragenden Daten. Für übliche TCP-Implementierungen gilt die Ungleichung `MSS < WSIZE`. Die dynamische Aushandlung dieser Parameter zusammen mit der Methode der Bestimmung der *Round Trip Time* (`RTT`) begründet ursächlich das gute Übertragungsverhalten von TCP in sehr unterschiedlichen Übermittlungsnetzen (siehe Abschnitt 3.4.2).

3.3.3 Auf- und Abbau von TCP-Verbindungen

Den Aufbau einer TCP-Verbindung zeigt Abb. 3.3-4. Hier soll insbesondere zum Ausdruck gebracht werden, dass eine TCP-Verbindung vollduplex ist und als ein Paar von zwei unidirektionalen logischen Verbindungen gesehen werden kann [Abb. 3.3-1c]. Die Kommunikationspartner befinden sich zum Anfang der Übertragung immer in folgenden Zuständen [Abb. 3.3-3]:

- *Passives Öffnen* (`Listen`): Eine Verbindung tritt in den Empfangsstatus ein, wenn eine Anwendungsinstanz TCP mitteilt, dass sie Verbindungen für eine bestimmte Portnummer annehmen möchte.
- *Aktives Öffnen* (`SYN sent`): Eine Applikation teilt dem TCP mit, dass sie eine Verbindung mit einer bestimmten IP-Adresse und Portnummer eingehen möchte, was mit einer bereits abhörenden Applikation korrespondiert.

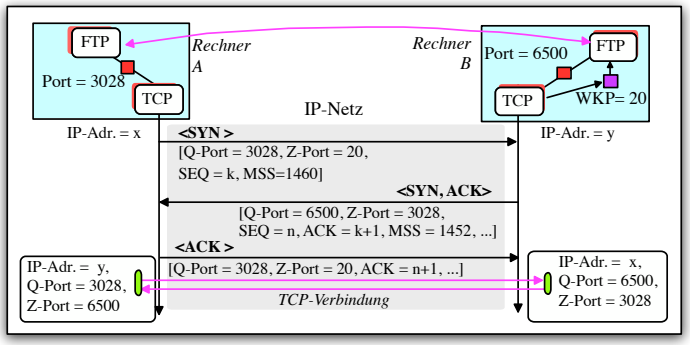


Abb. 3.3-4: Beispiel für den Aufbau einer TCP-Verbindung
Q: Quell, Z: Ziel, WKP: Well-known Port

Beispiel: FTP

In Abb. 3.3-4 wird die TCP-Verbindung im Rechner *A* mit der IP-Adresse `x` durch die Applikation FTP initiiert. Hierbei wird ihr für die Zwecke der Kommunikation beispielsweise die Portnummer 3028 zugewiesen. Die TCP-Instanz im Rechner *A* generiert ein TCP-Paket, in dem das Flag `SYN` gesetzt ist. Somit wird dieses Paket hier als `<SYN>`-Paket bezeichnet. Der Verbindungsaufbau beginnt damit, dass jeder der beiden Kommunikationspartner zunächst einen Anfangswert für die jeweilige Sequenznummer wählt. Dieser Anfangswert für eine Verbindung wird als *Initial Sequence Number* (`ISN`) bezeichnet.

<SYN>-Paket

Die TCP-Instanz im Rechner *A* sendet dazu an den Rechner *B* ein `<SYN>`-Paket, in dem u.a. folgende Informationen enthalten sind:

- `SYN`-Flag im TCP-Header wird gesetzt. Damit ist der Name `<SYN>`-Paket zu begründen.
- frei zugeteilte Nummer des Quellports; hierzu werden besonders die Ports[1] zwischen 5000 und 64000 benutzt.
- Zielport als *Well-known Port*, um die richtige Anwendung 'anzusprechen'. Ein *Well-known Port* ist ein Kontaktport (*Contact Port*) einer Applikation und kann als ihr Begrüßungsport angesehen werden.
- `SEQ`: *Sequenznummer* der Quell-TCP-Instanz (hier `SEQ = k`).

Das gesetzte `SYN`-Bit bedeutet, dass die Quell-TCP-Instanz eine Verbindung aufbauen (synchronisieren) möchte. Mit der Angabe des Zielports (als *Well-known Port*) wird die gewünschte Applikation im Rechner *B* gefordert.

Reaktion des Zielrechners auf <SYN>-Paket

Die Ziel-TCP-Instanz befindet sich im *Listenmodus*, sodass sie auf ankommende `<SYN>`-Pakete wartet. Nach dem Empfang eines `<SYN>`-Pakets leitet die Ziel-TCP-Instanz ihrerseits den Verbindungswunsch an die Ziel-Applikation (hier FTP) gemäß der empfangenen Nummer des Zielports weiter. Falls die Applikation im Zielrechner die ankommende TCP-

[1] Ports unter 1000 sind für Server-Anwendungen reserviert; die hohen Portnummern stehen für Client-Anwendungen zur Verfügung.

Verbindung akzeptiert, wird ihr ein Port als Puffer für zu sendende und zu empfangende Daten eingerichtet. Diesem Port wird eine große Nummer (z.B. 6500) zugeteilt. Danach wird eine eigene ISN in Richtung Rechner *A* generiert.

Im zweiten Schritt des Verbindungsaufbaus wird ein TCP-Paket im Rechner B mit folgendem Inhalt an den Rechner *A* zurückgeschickt:

<SYN,ACK>-Paket

- Die beiden Flags SYN und ACK im TCP-Header werden gesetzt, sodass man von einem <SYN,ACK>-Paket spricht.
- Die beiden Quell- und Zielportnummern werden angegeben. Als Quellport wird der neu im Rechner *B* eingerichtete Port 6500 angegeben. Als Zielport wird der Quellport im Rechner *A* eingetragen.
- Die Sequenznummer der Ziel-TCP-Instanz (hier SEQ = n) wird mitgeteilt.

Reaktion auf <SYN,ACK>-Paket

Das ACK-Bit signalisiert, dass die Quittungsnummer (hier kurz ACK) im <SYN,ACK>-Paket von Bedeutung ist. Die *Quittungsnummer* ACK enthält die nächste, von der TCP-Instanz im Rechner *B* erwartete Sequenznummer. Die TCP-Instanz im Rechner *A* bestätigt noch den Empfang des <SYN,ACK>-Pakets mit einem <ACK>-Paket, in dem das ACK-Flag gesetzt wird. Mit der *Quittungsnummer* ACK = n + 1 wird der TCP-Instanz im Rechner *B* bestätigt, dass die nächste Sequenznummer n + 1 erwartet wird.

MSS in <SYN>-Paketen

Zusätzlich teilt Rechner *A* die maximale Größe eines TCP-Segments Rechner *B* mit einem Wert von MSS = 1460 (Byte) mit, wohingegen Rechner *B* in der Gegenrichtung einen Wert von MSS = 1452 Byte vorschlägt, da *B* im TCP-Header noch (nicht gezeigte) Optionen übermittelt.

Aus Abb. 3.3-4 geht außerdem hervor, dass sich eine TCP-Verbindung aus zwei *unidirektionalen Verbindungen* zusammensetzt. Jede dieser gerichteten Verbindungen wird im Quellrechner durch die Angabe der IP-Zieladresse und der Quell- und Zielports eindeutig identifiziert.

Wurde eine TCP-Verbindung aufgebaut, so kann der Datenaustausch zwischen den kommunizierenden Applikationen erfolgen, genauer gesagt zwischen den mit der TCP-Verbindung logisch verbundenen Ports. Bevor wir aber auf die Besonderheiten der Datenübermittlung nach dem Protokoll TCP eingehen, soll zunächst der Abbau einer TCP-Verbindung kurz erläutert werden.

Abbau einer TCP-Verbindung

Den Abbau einer TCP-Verbindung illustriert Abb. 3.3-5. Im Normalfall kann der Abbau von einer der beiden kommunizierenden Applikationen initiiert werden. Da jede TCP-Verbindung sich aus zwei gerichteten Verbindungen zusammensetzt, werden diese gerichteten Verbindungen quasi nacheinander abgebaut. Jede TCP-Instanz koordiniert den Abbau ihrer gerichteten Verbindung zu ihrer Partner-TCP-Instanz und verhindert hierbei den Verlust von übertragenen, aber noch nicht quittierten Daten.

Maximum Segment Lifetime

Der Abbau wird von einer Seite mit einem TCP-Paket initiiert, in dem das FIN-Flag im Header gesetzt wird, sodass man dieses TCP-Paket als <FIN>-Paket bezeichnet. Dies wird von der Gegenseite durch ein <ACK>-Paket mit dem gesetzten ACK-Flag positiv bestätigt. Die positive Bestätigung erfolgt hier durch die Angabe der Quittungsnummer ACK = K + 1, d.h. der empfangenen Sequenznummer SEQ = K plus 1. Damit wird eine gerichtete Verbindung abgebaut. Der Verbindungsabbau in der Gegenrichtung wird mit dem TCP-Paket begonnen, in dem die beiden FIN- und ACK-Flags gesetzt sind, d.h. mit dem <FIN,ACK>-Paket. Nach der Bestätigung dieses <FIN,ACK>-Pakets durch die Gegenseite wird der Abbauprozess beendet.

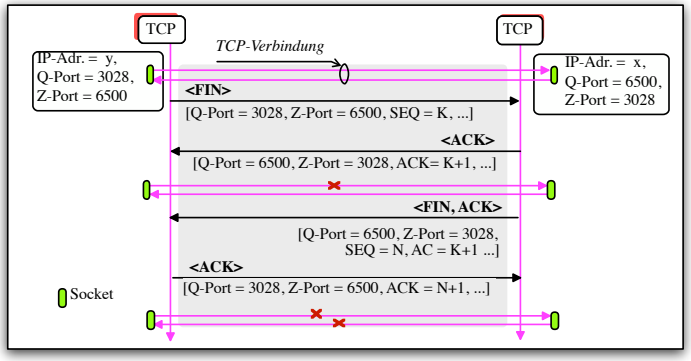


Abb. 3.3-5: Beispiel für den Abbau einer TCP-Verbindung

Beim Abbau einer Verbindung tritt u.U. ein zusätzlicher interner *Time-out-Mechanismus* in Kraft. Die TCP-Instanz geht in den Zustand *Active-Close*, versendet ein abschließendes `<ACK>`-Paket und befindet sich dann im Status *Time-Wait* [Abb. 3.3-3]. Dessen Zeitdauer beträgt `2 * MSL` (*Maximum Segment Lifetime*), bevor die TCP-Verbindung letztlich geschlossen wird. TCP-Pakete, die länger als die MSL-Dauer im Netz unterwegs sind, werden verworfen. Der Wert von *MSL* beträgt bei heutigen TCP-Implementierungen in der Regel 120 Sekunden. Anschließend wird der Port freigegeben und steht (mit einer neuen `ISN`) für spätere Verbindungen wieder zur Verfügung.

3.3.4 Flusskontrolle bei TCP

Sliding-Window-Prinzip

Bei der Datenkommunikation muss die Menge der übertragenen Daten an die Aufnahmefähigkeit des Empfängers angepasst werden. Sie sollte nicht größer sein als die Datenmenge, die der Empfänger aufnehmen kann. Daher muss die Menge der übertragenen Daten zwischen den kommunizierenden Rechnern entsprechend abgestimmt werden. Diese Abstimmung bezeichnet man oft als *Datenflusskontrolle* (*Flow Control*). Sie erfolgt beim TCP nach dem Prinzip *Sliding Window*. In Abschnitt 1.2.2 wurde bereits das Window-Prinzip (*Fensterprinzip*) bei der Nummerierung nach dem Modulo-8-Verfahren kurz erläutert. Bevor auf die Besonderheiten der Flusskontrolle beim TCP eingegangen wird, soll das allgemeine *Sliding-Window-Prinzip* näher veranschaulicht werden.

Für die Zwecke der Flusskontrolle nach dem Sliding-Window-Prinzip dienen folgende Angaben im TCP-Header:

- *Sequence Number* (Sequenznummer),
- *Acknowledgement Number* (Quittungs- bzw. Bestätigungsnummer),
- *Window* (Fenstergröße).

Sequenznummer

Mit der Sequenznummer `SEQ` werden die zu sendenden Daten fortlaufend byteweise nummeriert. Sie besagt, mit welcher Nummer die Nummerierung der im TCP-Paket gesendeten Byte beginnen soll [Abb. 3.3-8].

Quittungsnummer und `Window`

Mit der Quittungsnummer teilt der Empfänger dem Sender mit, welche Sequenznummer als nächste bei ihm erwartet wird. Hierbei wird die Angabe `Window` wie folgt interpretiert:

- *Seitens des Senders* stellt die Window-Größe (Fenstergröße) bei TCP die maximale Anzahl von Daten in Byte dar, die der Sender absenden darf, ohne auf eine Quittung vom Empfänger warten zu müssen.
- *Seitens des Empfängers* ist `Window` als Anzahl der Daten in Byte zu verstehen, die von diesem auf jeden Fall immer aufgenommen werden können.

Wird die maximale Länge von Datensegmenten in TCP-Paketen festgelegt, so kann die Menge von Daten unterwegs mit den erwähnten drei Parametern (Sequenz- und Quittungsnummer sowie Window) jederzeit kontrolliert werden.

Flusskontrolle

Abb. 3.3-6 veranschaulicht die Flusskontrolle nach dem Sliding-Window-Prinzip mit der `Window-Größe = 4`. Wie hier ersichtlich ist, lässt sich das Window als Sendefenster interpretieren.

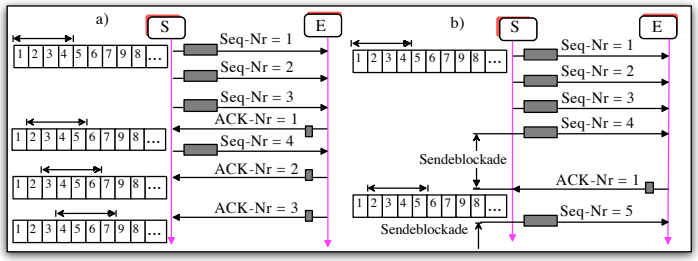


Abb. 3.3-6: Sliding-Window-Prinzip bei Window-Größe = 4:
a) fehlerfreie Übertragung, b) Bedeutung der Sende-Blockade
ACK: Quittungsnummer (Acknowledgement Number), Seq-Nr.: Sequenznummern;
E: Empfänger, S: Sender

Beispiel: Betrachten wir zunächst das Beispiel in Abb. 3.3-6a. Da die Window-Größe 4 beträgt, darf der Sender nur 4 Byte absenden, ohne auf eine Quittung warten zu müssen. Dies bedeutet, dass er die Byte mit den Nummern 1, 2, 3 und 4 absenden darf. Nach dem Absenden der ersten drei Byte ist eine Quittung eingetroffen, mit der das erste Byte quittiert wird. Dadurch verschiebt sich das Fenster (*Window*) mit den zulässigen Sequenznummern um eine Position nach rechts. Da maximal 4 Byte unterwegs sein dürfen, kann der Sender nun die nächsten Byte mit den Nummern 4 und 5 senden. Nach dem Absenden des Byte mit Sequenznummer 5 wird das Byte mit der Sequenznummer 2 durch den Empfänger positiv quittiert. Dadurch verschiebt sich das Fenster nach rechts um eine Position weiter.

Sendeblockade

Abb. 3.3-6b zeigt die Situation, in der der Sendeprozess blockiert werden muss (*Sendeblockade*). Sie kommt dann oft vor, wenn einerseits die Verzögerungszeit im Netz groß und andererseits die Window-Größe zu klein ist. Mit großen Verzögerungszeiten ist immer zu rechnen, wenn eine Satellitenstrecke als ein Übertragungsabschnitt eingesetzt wird. Wie hier ersichtlich ist, muss der Sender nach dem Absenden von Byte mit den Sequenznummern 1, 2, 3 und 4 auf eine Quittung warten. Hier wurden die Daten aus dem Sendefenster abgesendet, und deren Empfang wurde noch nicht bestätigt. Bevor einige Byte quittiert werden, darf der Sender keine weiteren Byte senden. Nach dem Eintreffen der Quittung für das Byte mit Sequenznummer 1 verschiebt sich das Sendefenster um eine Position nach rechts. Das Byte mit der Sequenznummer 5 darf nun gesendet werden. Anschließend muss der Sendevorgang wiederum bis zum Eintreffen der nächsten Quittung blockiert werden.

3.3.5 TCP Sliding-Window-Prinzip

Bei TCP erfolgt die Flusskontrolle nach dem *Sliding-Window-Prinzip*. Hierbei legt die Window-Größe die maximale Anzahl von Byte fest, die der Quellrechner absenden darf, ohne auf eine Quittung vom Zielrechner warten zu müssen. Abb. 3.3-7 illustriert die Interpretation von Window bei TCP.

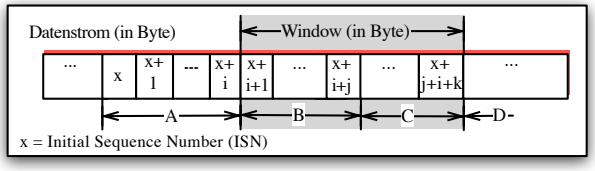


Abb. 3.3-7: Interpretation von Window bei TCP

Byte in Flight

Mit dem Parameter *Window* wird bei TCP ein Bereich von Nummern markiert, die den zu sendenden Datenbyte zuzuordnen sind. Dieser Bereich kann als *Sendefenster* angesehen werden. Im Strom der Datenbyte sind vier Bereiche zu unterscheiden:

- *A*: `i` Datenbyte, die abgesendet und bereits positiv quittiert wurden,
- *B*: `j` Datenbyte, die abgesendet und noch nicht quittiert wurden ('*Data in flight*'),
- *C*: `k` Datenbyte, die noch abgesendet werden dürfen, ohne auf eine Quittung warten zu müssen,
- *D*: Datenbyte außerhalb des Sendefensters. Diese Datenbyte dürfen erst dann abgesendet werden, wenn der Empfang von einigen vorher abgeschickten Datenbyte bestätigt wird.

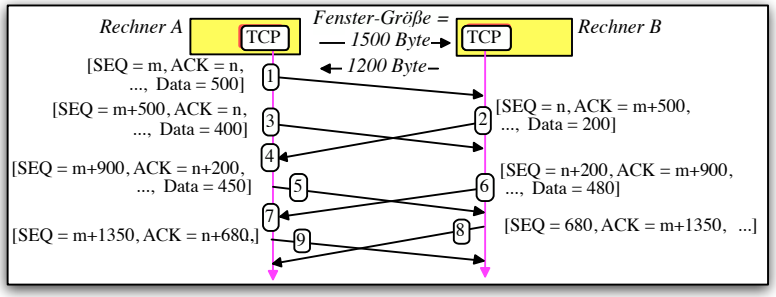


Abb. 3.3-8: Beispiel für den TCP-Ablauf bei der fehlerfreien Datenübermittlung
SEQ: Sequenznummer, ACK: Quittungsnummer

Fehlerfreie Datenübermittlung

Beispiel: Im Folgenden wird der Datenaustausch nach TCP verdeutlicht und damit auch das Sliding-Window-Prinzip näher erläutert. Abb. 3.3-8 zeigt ein Beispiel für eine fehlerfreie Datenübermittlung. Die hier dargestellten einzelnen Ereignisse sind wie folgt zu interpretieren:

1. Das erste TCP-Paket von Rechner *A* zu Rechner *B* mit der Sequenznummer `SEQ = m`. Dieses Paket enthält die ersten `500` Datenbyte. `SEQ = m` verweist darauf, dass den einzelnen übertragenen Datenbyte die Nummern `m, m+1, ..., m+499` zugeordnet sind.

2. Von Rechner *B* wird mit einem TCP-Paket, in dem 200 Datenbyte enthalten sind und bei dem das ACK-Flag gesetzt wurde, bestätigt, dass das erste TCP-Paket von Rechner *A* fehlerfrei aufgenommen wurde und das nächste Datenbyte mit der Nummer m+500 erwartet wird. SEQ = n besagt, dass die Nummern n, n+1, ..., n+199 den hier übertragenen Datenbyte zuzuordnen sind.
3. Das zweite TCP-Paket von Rechner *A* zu Rechner *B* mit den nächsten 400 Datenbyte und mit SEQ = m+500. Somit enthält dieses TCP-Paket die Datenbyte mit den Nummern m+500, m+501, ..., m+899. Damit wurden bereits 900 Datenbyte von Rechner *A* abgeschickt und vom Zielrechner *B* noch nicht quittiert. Da die Window-Größe in Richtung zum Rechner *B* 1500 Byte beträgt, können vorerst nur noch 1500 - 900 = 600 Datenbyte abgesendet werden.
4. Nach dem Empfang dieses TCP-Pakets werden 500 Datenbyte vom Zielrechner *B* positiv quittiert. Somit verschiebt sich das Sendefenster im Rechner *A* um 500. Da der Empfang von 400 Byte (das zweite TCP-Paket) noch nicht bestätigt wurde, können noch 1500-400 = 1100 Datenbyte gesendet werden.
5. Das dritte TCP-Paket von Rechner *A* zu Rechner *B* mit SEQ = m+900 und mit 450 Datenbyte. Dieses Paket enthält die Datenbyte mit den Nummern von m+900 bis m+1349. Somit sind bereits 850 Datenbyte abgeschickt, die noch nicht quittiert wurden. Darüber hinaus können nur noch weitere 650 (d.h. 1500 - 850) Datenbyte abgesendet werden. Mit diesem TCP-Paket wird dem Rechner *B* auch mitgeteilt, dass die nächsten Datenbyte ab Nummer n+200 erwartet werden.
6. Rechner *B* quittiert mit einem TCP-Paket, in dem das ACK-Flag gesetzt wird, das dritte TCP-Paket von Rechner A und sendet zu Rechner *A* die nächsten 480 Datenbyte. Diesen Datenbyte sind die Nummern n+200, ..., n+679 zuzuordnen.
7. Nach dem Empfang dieses TCP-Pakets werden die an Rechner *A* abgeschickten Datenbyte mit den Nummern m+900-1 positiv quittiert. Damit verschiebt sich in Rechner *A* das Sendefenster entsprechend.
8. Rechner *B* bestätigt mit einem TCP-Paket, in dem das ACK-Flag gesetzt wird, die Datenbyte einschließlich bis zur Nummer m+1350-1. Auf diese Weise wurden alle zu Rechner *B* abgeschickten Daten quittiert. Rechner *A* kann nun an Rechner *B* die durch die Window-Größe festgelegte Datenmenge (d.h. 1500 Byte) unmittelbar weitersenden, ohne vorher auf eine positive Quittung von Rechner *B* warten zu müssen.
9. Rechner *A* quittiert mit ACK = n+680 positiv Rechner *B*, alle Datenbyte bis zur Nummer n+680-1 empfangen zu haben. Zugleich wird mit SEQ = m+1350 signalisiert, dass bereits m+1350 Datenbyte geschickt wurden.

Sendeblockade bei der Datenübermittlung

Den Ablauf des Protokolls TCP bei einer fehlerbehafteten Datenübermittlung illustriert Abb. 3.3-9.

Beispiel: Die einzelnen Ereignisse sind hier folgendermaßen zu interpretieren:

1. Das erste TCP-Paket von Rechner *A* zu Rechner *B* mit SEQ = m und die ersten 500 Datenbyte. Diesen Datenbyte sind daher die Nummern m, m+1, ..., m+499 zuzuordnen. Mit ACK = n wird dem Rechner *B* mitgeteilt, dass das nächste Datenbyte mit der Nummer *n* von ihm erwartet wird.
2. Das zweite TCP-Paket von Rechner *A* zu Rechner *B* mit den nächsten 500 Datenbyte und mit SEQ = m+500. Somit enthält dieses TCP-Paket die Datenbyte mit den Nummern m+500, ..., m+999. Mit dem Absenden dieser 200 Datenbyte wurden die Nummern zur Vergabe der zu sendenden Datenbyte 'verbraucht' (⇒ Window = 1000 Byte). Aus diesem Grund muss der Sendeprozess blockiert werden.

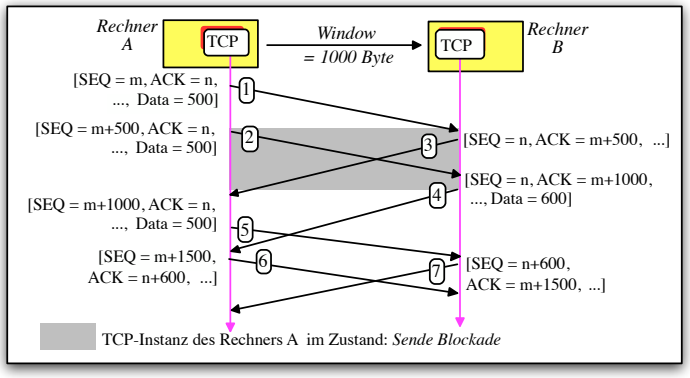


Abb. 3.3-9: Beispiel für das TCP-Verhalten bei einer fehlerbehafteten Datenübermittlung

3. Es werden 500 Datenbyte vom Zielrechner *B* positiv quittiert. Damit verschiebt sich das Sendefenster in Rechner *A* entsprechend, sodass weitere 500 Datenbyte gesendet werden dürfen.
4. Rechner *B* sendet 600 Datenbyte und quittiert dem Rechner *A* alle Datenbyte bis einschließlich Nummer `m+1000-1`.
5. Das dritte TCP-Paket von Rechner *A* zu Rechner *B* mit den nächsten 500 Datenbyte und mit der Sequenznummer `SEQ = m+1000`.
6. Rechner *A* quittiert Rechner *B* alle Datenbyte bis einschließlich Nummer `n+600-1`.
7. Rechner *B* quittiert Rechner *A* alle Datenbyte bis einschließlich Nummer `m+1500-1`.

Fehlerbehaftete Datenübermittlung

Da IP zu den unzuverlässigen Protokollen gehört, muss TCP über einen Mechanismen verfügen, der es in die Lage versetzt, mögliche Fehler auf den unteren Protokollschichten (z.B. Verlust von IP-Paketen, Verfälschung der Reihenfolge usw.) zu erkennen und zu beheben. Der von TCP verwendete Mechanismus ist bemerkenswert einfach:

> Ist die Wartezeit für *ein* abgesendetes TCP-Segment überschritten, innerhalb derer eine Bestätigung erfolgen sollte (*Maximum Segment Lifetime*), wird die Übertragung *aller* Datenbyte wiederholt, für die bis dahin noch keine Quittungen vorliegen.

Im Unterschied zu anderen Methoden der Fehlerkontrolle kann hier der Empfänger zu keinem Zeitpunkt eine wiederholte Übertragung erzwingen. Dies liegt zum Teil daran, dass kein Verfahren vorhanden ist, um negativ zu quittieren, sodass keine wiederholte Übertragung einzelner TCP-Pakete direkt veranlasst werden kann. Der Empfänger muss einfach abwarten, bis das von vornherein festgelegte Zeitlimit *MSL* (*Maximum Segment Lifetime*) auf der Sendeseite abgelaufen ist und infolgedessen bestimmte Daten nochmals übertragen werden.

MSL Timer

Das Funktionsweise des MLS-Timer bei TCP zeigt Abb. 3.3-10. Um die Darstellung zu vereinfachen, werden hier nur jene Angaben gezeigt, die nötig sind, um dieses Prinzip zu erläutern.

Beispiel: Wie in Abb. 3.3-10 dargestellt, wird der MSL-Timer nach dem Absenden jedes TCP-Segments neu gestartet. Mit diesem *Timer* wird eine maximale Wartezeit (*Timeout*) auf die Quittung angegeben. Kommt innerhalb dieser festgelegten maximalen Wartezeit

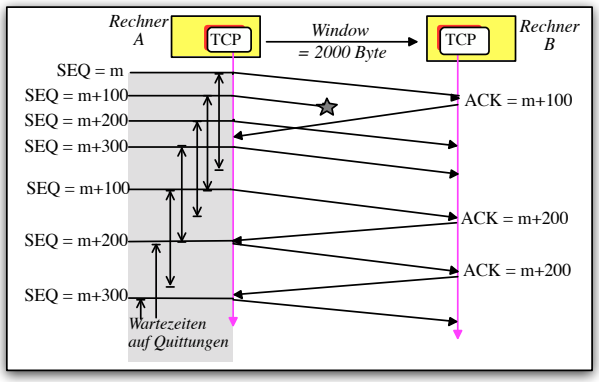


Abb. 3.3-10: Gesteuerte Segmentwiederholung über den TCP MLS Timer
ACK: Quittungsnummer, SEQ: Sequenznummer

keine Quittung an, wird die Übertragung des betreffenden TCP-Pakets wiederholt. Darin besteht das eigentliche Prinzip der Fehlerkontrolle bei TCP.

Das TCP-Paket mit der Sequenznummer `m+100` hat den Empfänger nicht erreicht, obwohl die später abgeschickten TCP-Pakete (mit den Sequenznummern `m+200` und `m+300`) dort ankamen. Die TCP-Instanz im Rechner *B* sendet keine Bestätigung für den Empfang des TCP-Pakets mit der Sequenznummer `m+200`, da die Datenbyte mit den Nummern `m+100, ..., m+199` noch nicht empfangen wurden. Das nächste TCP-Paket mit `SEQ = 300` wird ebenfalls nicht bestätigt. Dies hat zur Folge, dass das Zeitlimit für die Übertragung des TCP-Pakets mit der Sequenznummer `x+100` abläuft und dieses Paket infolgedessen erneut übertragen wird.

Bestätigung

Der Empfang der TCP-Pakete wird nur dann bestätigt, wenn ihre Reihenfolge vollständig ist. Somit kann die Situation eintreten, dass eine Reihe von TCP-Paketen (soweit die Window-Größe dies zulässt) sogar dann wiederholt übertragen werden muss, wenn sie bereits fehlerfrei beim Zielrechner ankamen. Wie dem Beispiel in Abb. 3.3-10 zu entnehmen ist, betrifft dies in unserem Fall die TCP-Pakete mit den Sequenznummern `m+200` und `m+300`, sie müssen nochmals übertragen werden.

Round Trip Time

Die maximale Wartezeit auf die Quittung ist ein zentraler Parameter von TCP. Er hängt von der zu erwartenden Verzögerung im Netz ab. Die Verzögerung im Netz kann durch die Messung der Zeit, die bei der Hin- und Rückübertragung zwischen dem Quell- und dem Zielrechner auftritt, festgelegt werden. In der Literatur wird diese Zeit als *Round Trip Time* (`RTT`) bezeichnet. In Weitverkehrsnetzen, in denen auch Satellitenverbindungen eingesetzt werden, kann es einige Sekunden dauern, bis eine Bestätigung ankommt.

Im Laufe einer Verbindung kann die `RTT` aufgrund der Netzbelastung schwanken. Daher ist es nicht möglich, einen festen Wert für die maximale Wartezeit auf die Quittung einzustellen. Wird ein zu kleiner Wert gewählt, läuft die Wartezeit ab, bevor eine Quittung eingehen kann und das Segment unnötig erneut gesendet. Wird ein zu hoher Wert gewählt, hat dies lange Verzögerungspausen zur Folge, da die gesetzte Zeitspanne abgewartet werden muss, bevor eine wiederholte Übertragung stattfinden kann. Durch den Verlust eines Segments kann der Datendurchsatz erheblich sinken.

7 Das Protokoll IPv6

Notwendigkeit von IPv6

Das massive Wachstum des Internet, die hierbei entstehenden Probleme und neuen Anforderungen haben in den 90er Jahren des letzten Jahrhunderts die Entwicklung eines neuen Internetprotokolls stark vorangetrieben. Insbesondere stößt der Adressraum des klassischen Internetprotokolls IP, das auch als IPv4 (v4 steht für Version 4) bezeichnet wird, aufgrund der Adresslänge von nur 32 Bit an seine Grenzen. Als IPv4 konzipiert wurde, konnte man sich nicht vorstellen, dass das Internet zu einem weltumgreifenden Kommunikationsnetz heranwachsen würde, dessen 'Hosts' nicht nur fast alle tragbaren, an das Internet über Mobilfunknetze angebundenen Endgeräte sein werden, sondern als 'Host' am *Internet of Things* auch diverse Komponenten zur Ansteuerung allgegenwärtiger technischer 'Dinge' kommen können. Zudem besteht für IP-Netze der Bedarf an verbesserter Sicherheit sowie der Unterstützung von Multimedia- und Echtzeitanwendungen.

Diese Ziele sollten mit dem neuen Internetprotokoll IPv6 (d.h. in der Version 6) erreicht werden. Zum Lösung dieser Probleme wurde schon Anfang der 90er Jahre ein Entwurf als IPnG (next Generation) gemacht, der im Anschluss in das Protokoll IPv6 mündete. Weil die Versionsnummer 5 bereits vorher für *Stream Protocol* (ST) als eine IP-Variante vergeben wurde, ist somit IPv6 der dedizierte Nachfolger von IPv4.

IPv6 als mächtiges Protokoll

Mit IPv6 steht ein mächtiges Protokoll zur Verfügung, mit dem sich die Konfiguration und Administration der Netzwerke deutlich vereinfachen lässt, aber ein grundlegendes Verständnis der IPv6-Adressstruktur und der Mechanismen der dynamischen IPv6-Adressvergabe verlangt. Bei IPv6 wurde nicht nur die Länge der Adresse auf 128 Bit erweitert, sondern zugleich auch eine Vielzahl wichtiger Erweiterungen eingeführt. Diese reichen von Sicherheitsfunktionen über mehr Flexibilität bis hin zur Unterstützung von Plug&Play und neuartigen Anwendungen.

Überblick über das Kapitel

Nach einer Vorstellung der Neuerungen bei IPv6 in Abschnitt 7.1 gehen die Abschnitte 7.2 und 7.3 auf IPv6-Header und die sog. Erweiterungs-Header ein. Die Nutzung von Options-Headern erläutert Abschnitt 7.4, den Einsatz von IPv6-Jumbogrammen zeigt Abschnitt 7.5. Source Routing illustriert Abschnitt 7.6, und die Fragmentierung langer Pakete präsentiert Abschnitt 7.7. Eine Beschreibung der Adressstrukturen bei IPv6 befindet sich in Abschnitt 7.8. Die Abschnitte 7.9 und 7.10 stellen Unicast-, Multicast- und Anycast-Adressen sowie deren Bedeutung und Einsatz dar. Der Zuweisung von IPv6-Adressen ist der Abschnitt 7.11 gewidmet und das Kapitel wird mit einem Ausblick in Abschnitt 7.12 abgerundet.

Ziele dieses Kapitels

In diesem Kapitel werden u.a. folgende Fragen beantwortet:

- Welche Ziele wurden bei der Entwicklung von IPv6 verfolgt?
- Welche neuen Funktionen bringt IPv6 mit sich?
- Wie sind IPv6-Pakete aufgebaut?
- Wie wird die Route für die Übermittlung der IPv6-Pakete bestimmt?
- Wie ist der IPv6-Adressraum strukturiert?
- Welche Arten von IPv6-Adressen gibt es und für welche Zwecke werden sie genutzt?
- Wie findet die automatische Vergabe von IPv6-Adressen statt?
- Wie können den Rechnern IPv6-Adressen zugewiesen werden?

7.1 Neuerungen bei IPv6 gegenüber IPv4

IPv6 wurde mit dem Ziel entwickelt, ein neues Internetprotokoll zur Verfügung zu stellen, das die Nachteile und Schwächen von IPv4 beheben soll. Welche Nachteile und Schwächen hat aber IPv4? Man hört oft nur: Die IPv4-Adressen sind knapp. Die Struktur des IPv4-Headers ist einerseits zu komplex und andererseits wurde noch nicht alles berücksichtigt (z.B. Sicherheitsaspekte). Zudem muss bei IPv4 bei jedem Hop im Router die Header Checksum neu berechnet werden, was bei heutigen Routern häufig nicht mehr vorgenommen wird.

Ziele von IPv6

Die wichtigsten Ziele bei der Entwicklung von IPv6 waren:

- *Vergrößerung und hierarchische Strukturierung des IP-Adressraums*
  Aus diesem Grund haben die IPv6-Adressen die Länge von 128 Bit (sind also 4-mal länger als IPv4-Adressen). wobei die ersten Bit den Gültigkeitsbereich (*Scope*) der IPv6-Adresse, die folgenden Bit zum Routing und die niedrigwertigsten 64 Bit zur Identifikation das Interface vorgesehen sind.
- *Verbesserung der Header-Struktur*
  Die Struktur des Headers der IPv6-Pakete wurde gegenüber dem Header der IPv4-Pakete wesentlich verbessert. Es wurde eine deutlichere Unterteilung zwischen notwendigen und optionalen Angaben vorgenommen. Die optionalen Angaben können nun bedarfsweise in speziellen Headern (sog. *Extension Headers*) übermittelt werden.
- *Gültigkeitsbereich von IPv6-Adressen*
  Jeder IPv6-Adresse wird ein Gültigkeitsbereich (*Scope*) zugesprochen. Den öffentlichen IPv4-Adressen entsprechen bei IPv6 solche mit globalen Scope, die globalen Unicast-Adressen. IPv6 *Unique Local Unicast Addresses* besitzen nur Gültigkeit im lokalen Linksegment und sind somit den privaten IPv4-Adressen [Abb. 7.9-7b] gleichgestellt. Die sog. *Link-Local Unicast Addresses* stellen 'technische' Adressen dar, die zum Ablauf des IPv6-Protokolls im lokalen Linksegment notwendig sind [Abb.7.9-7a]. Bei IPv6 werden auch spezielle Adresstypen definiert, um verschiedene Fälle zu unterstützen, in denen die beiden Protokolle IPv4 und IPv6 parallel eingesetzt werden [Abschnitt 7.9.5].
- *Autoconfiguration*
  IPv6-fähige Rechner richten ihre Netzwerkkonfiguration pro Interface nach dem Plug-and-Play-Prinzip eigenverantwortlich ein: *Autoconfiguration* [Abschnitt 7.2]. Damit dies unabhängig für jedes Interface und das angeschlossene Linksegment erfolgen kann, muss die IP-Instanz sich nun einen *Interface-Identifier* (Interface-ID) bzw. allgemein ein *Link-Token* merken, was zu Änderungen der Socket-Schnittstelle führt.
- *Verbesserung der Sicherheit*
  Hierfür wurden die beiden Extension Headers *Encapsulation Security Payload* und *Authentication Header* vorgesehen. Es hat sich aber herausgestellt, dass diese beiden Extension Headers auch beim IPv4 eingesetzt werden können. Dies hat zur Entstehung des ergänzenden Protokolls IPsec (*IP Security*) für IPv4 geführt [Abschnitt 5.4].
- *Default Minimum-MTU-Size von 1280 Byte*
  Ein IPv6-Netz verlangt, dass die Data-Link-Schicht in der Lage sein muss, IPv6-Pakete bis zu einer Grösse von 1280 Byte ohne Einschränkungen zu übermitteln [RFC 2460]. Für IPv4 lag dieser Wert bei lediglich 576 Byte. UDP-Nachrichten (wie z.B. bei DNS im Einsatz) können auf IPv6-Netzen daher deutlich größer sein.

Sowohl bei IP4 als auch bei IP6 finden zusätzlich weitere Hilfsprotokolle und Routing-Protokolle Anwendung, sodass man von einer IPv4- bzw. IPv6-Protokollfamilie sprechen kann. Abb. 7.1-1 zeigt die wesentlichen Neuerungen bei IPv6 bei einer Gegenüberstellung der beiden Protokollfamilien auf. Bei der Protokollfamilie von IPv6 wird hier auch angegeben, in welchem IETF-Dokument das entsprechende Protokoll spezifiziert wird.

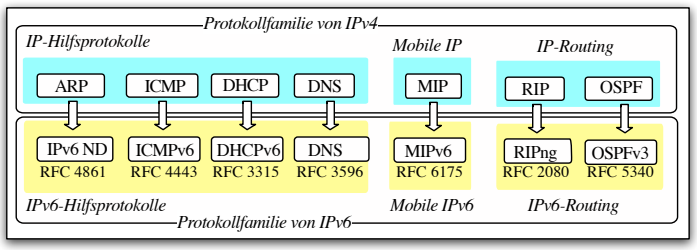


Abb. 7.1-1: Gegenüberstellung der Protokollfamilien von IPv4 und von IPv6

Wie hier ersichtlich, lassen sich die Neuerungen bei IPv6 gegenüber IPv4 wie folgt zusammenfassen:

- ARP (*Address Resolution Protocol*) gibt es bei IPv6 nicht mehr. Die Funktionen von ARP hat bei IPv6 das Protokoll NDP (*Neighbor Discovery Protocol*) übernommen. Die wichtigste Aufgabe von NDP ist somit die Ermittlung von MAC-Adressen aufgrund einer vorliegenden IPv6-Adresse [Abschnitt 8.2].
- ICMP (*Internet Control Message Protocol*) wurde für IPv6 modifiziert und trägt die Bezeichnung ICMPv6 [Abschnitt 8.1]. Die ICMPv6-Nachrichten werden u.a. in den Protokollen NDP und MIPv6 verwendet.
- Die dynamische Vergabe von Adressen und die Zuweisung ergänzender Angaben für die Anwendungen erfolgt bei IPv6 nach dem Protokoll DHCPv6 [Abschnitt 8.4], das eine Modifikation von DHCP (*Dynamic Host Configuration Protocol*) darstellt.
- Das DNS (*Domain Name System*) wurde für die Unterstützung von IPv6 entsprechend erweitert, was wir bereits in Abschnitt 4.2 besprochen haben.
- Die Mobilität von Rechnern wird bei IPv4 durch das Protokoll MIP (*Mobile IP*) unterstützt. Die für IPv6 modifizierte Version von MIP trägt die Bezeichnung MIPv6 [Abschnitt 15.4].
- In Netzen mit IPv4 werden die Routing-Protokolle RIP (*Routing Information Protocol*) und OSPF (*Open Shortest Path First*) eingesetzt. Für den Einsatz in Netzen mit IPv6 wurden diese Protokolle entsprechend modifiziert. Man bezeichnet diese Modifikationen als RIPng und OSPFv6 [Kapitel 10].

7.2 Header-Struktur bei IPv6

Zu den wichtigsten Zielen bei der Entwicklung von IPv6 zählte einerseits, die recht umfangreiche und nur aufwendig zu bearbeitende Struktur des IPv4-Headers zu vereinfachen und andererseits Flexibilität und Möglichkeiten für zukünftige Erweiterungen zuzulassen.

Aufbau des IPv6-Headers

Abb. 7.2-1 zeigt die Struktur des Header von IPv6. Man beachte, dass einige sog. Erweiterungs-Header (Extension Headers) direkt nach dem IPv6-Header ins Paket ein-

gebettet werden können [Abb. 7.3-1]. Abb. 7.2-1 illustriert den Fall, in dem keine Erweiterungs-Header im IPv6-Datenpaket enthalten sind.

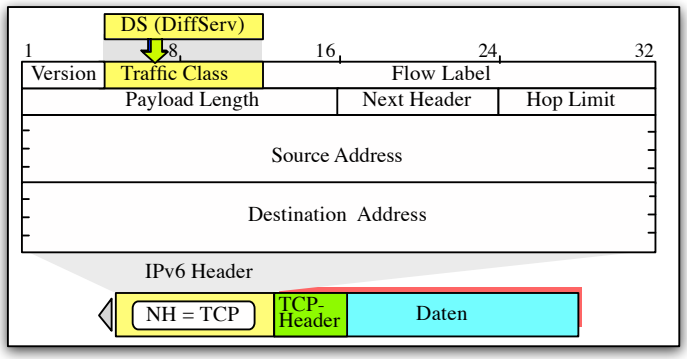


Abb. 7.2-1: Struktur des Header von IPv6 (nach RFC 2460)
ohne ergänzende Erweiterungs-Header
DS: Differentiated Services, NH: Next Header

Der IPv6-Header stellt eine deutliche Vereinfachung und Erweiterung gegenüber dem IPv4-Header dar. Wie bereits in Abschnitt 2.2 gezeigt, enthält der IPv4-Header zehn Felder, zwei jeweils 32 Bit lange IPv4-Adressen und eventuell ein Feld mit Optionen, das immer bis zur nächsten 32-Bit-Grenze aufgefüllt wird. Diese Konstruktion führt beim IPv4-Header zu einer Länge von 20 Byte ohne Optionen. Die minimale Größe des IPv4-Header ist daher 20 Byte.

128-Bit-IP-Adressen

Im Gegensatz dazu verfügt der IPv6-Header nur über sechs Felder, zwei jeweils 128 Bit lange IPv6-Adressen und keine Optionen. Die Anzahl der Felder wurde auf das Minimum beschränkt, um den Paket-Overhead zu verringern und damit die Effizienz der Übertragung zu verbessern. Trotz viermal so langer Quell- und Ziel-IPv6-Adressen, die allein 24 Byte belegen, ist der IPv6-Header mit 40 Byte nur doppelt so lang wie der IPv4-Header.

Die einzelnen Angaben im IPv6-Header haben folgende Bedeutung:

- `Version` (4 Bit)
  Hier wird die Version des IP-Protokolls angegeben. Für IPv6 enthält dieses Feld die Zahl 6.
- `Traffic Class` (8 Bit)
  Um *Quality of Service* in IPv6-Netzen zu unterstützen, ermöglicht dieses Feld dem Sender bzw. Router von Paketen diesen eine Priorität zu vergeben. Daher kann eine höhere Priorität den IPv6-Paketen mit zeitkritischen Daten (z.B. bei VoIP-Anwendungen) im Vergleich zu den IPv6-Paketen mit 'normalen' Daten zugeordnet werden. Für die Garantie der Kompatibilität mit IPv4 wird hier das Feld `Differentiated Services` nach RFC 2474 eingebettet [Abschnitt 2.2.1].
- `Flow Label` (20 Bit)
  Dieses Feld stellt die zufällig gewählte Identifikationsnummer einer unidirektionalen, virtuellen Ende-zu-Ende-Verbindung (z.B. für eine TCP-Verbindung) dar. Diese Angabe kann dazu genutzt werden, jene Pakete zu kennzeichnen, die eine besondere Behandlung im Übermittlungsnetz benötigen. Die Router unterwegs können alle zu

einer Ende-zu-Ende-Verbindung gehörenden Pakete anhand ihres Flow Label direkt weiterleiten, ohne den Rest des IPv6-Header auswerten zu müssen. Eine derartige Weiterleitung der Pakete könnte beispielsweise ermöglichen, isochrone Bitströme bei der Multimedia-Kommunikation über IPv6-Netze zu übermitteln. `Flow Label` wird näher in RFC 3697 bzw. 6437 spezifiziert. Durch den Einsatz von MPLS [Abschnitt 11.2] ist aber `Flow Label` fast überflüssig geworden.

- `Payload Length` (16 Bit)
  Hier wird angegeben, wie viele Byte (Oktette) nach dem IPv6-Header als Nutzlast (*Payload*) noch folgen. Somit kann diese Angabe als Nutzlastlänge angesehen werden. Da dieses Feld 16 Bit enthält, lassen sich theoretisch maximal 2^{16} = 65536 Byte als Nutzlast (weitere Steuerungsangaben und Daten) in einem IPv6-Paket transportieren. Eine Nutzlastlänge von 0 verweist auf ein sog. *Jumbo-Paket* [Abschnitt 7.5].
- `Next Header` (8 Bit)
  In diesem Feld wird der Header-Typ angezeigt, der unmittelbar nach dem IPv6-Header folgt. Es handelt sich hierbei
  - entweder um den Header eines nächsthöheren Protokolls – d.h. aus der Schicht 4 (Transportschicht) – wie z.B. TCP bzw. UDP
  - oder um einen *Erweiterungs-Header* (*Extension Header*), der eine Erweiterung des IPv6-Header ermöglicht, um bestimmte zusätzliche Steuerungsangaben über das Netz zu übermitteln [Abb. 7.3-2].

  Das Feld `Next Header` entspricht der Funktion nach dem Feld `Protocol` im IPv4-Header [Abb. 2.2-1].
- `Hop Limit` (8 Bit)
  Dieses Feld gibt die maximale Anzahl von Routern an, die ein Paket durchlaufen darf, bevor es automatisch gelöscht wird. Dies entspricht dem Feld `Time To Live` bei IPv4. Der hier eingetragene Wert wird in jedem durchlaufenen Router um 1 reduziert. Der Router, der den Wert auf 0 setzt, verwirft das betreffende Paket und signalisiert dies der Quelle mit der ICMPv6-Nachricht `Time Exceeded` [Abschnitt 8.1].
- `Source Address` (128 Bit)
  In diesem Feld steht eine IP-Adresse des Quellrechners.
- `Destination Address` (128 Bit)
  Hier wird die Adresse des Empfängers angegeben. Falls `Routing Header` als eine Erweiterung des IPv6-Header existiert, kann hier auch die Adresse einer 'Zwischenstation' (z.B. ein geforderter Router) angegeben werden.

Eine wichtige Besonderheit von IPv6 besteht darin, dass einige zusätzliche Steuerungsangaben in Form von festgelegten *Erweiterungs-Headern* (*Extension Headers*) zwischen dem IPv6-Header und dem TCP/UDP-Header eingebettet werden können.

7.3 Erweiterungs-Header

Das Feld `Next Header` im IPv6-Header nimmt eine zentrale Rolle bei der Strukturierung der IPv6-Pakete ein, und mit dessen Hilfe können die Verweise auf die Erweiterungen des IPv6-Header gemacht werden. `Next Header` weist darauf hin, was direkt nach dem IPv6-Header folgt. Es sind hierbei zwei Fälle zu unterscheiden: Entweder folgt

- direkt ein TCP- bzw. UDP-Header mit den dazugehörigen Daten [Abb. 7.2-1] oder
- zuerst ein weiterer *Erweiterungs-Header*, der wiederum ein Feld `Next Header` enthält [Abb.7.3-1]. Nach den Erweiterungs-Headern folgen dann ein TCP- bzw. UDP-Header und anschließend die Daten.

Angabe: Next Header

In einem IPv6-Paket lassen sich beliebig viele Erweiterungs-Header aneinander reihen, bis schließlich der Header des entsprechenden Transportprotokolls (z.B. TCP) beginnt. Das Prinzip einer derartigen Erweiterung des IPv6-Header illustriert Abb. 7.3-1. Ein IPv6-Paket kann keinen, einen oder mehrere Erweiterungs-Header enthalten. Da die einzelnen Erweiterungs-Header beliebige Längen aufweisen können, ist es notwendig, gemäß RFC 6564 die Länge des Headers durch einen 8-Bit-Wert anzugeben [Abb. 7.3-2].

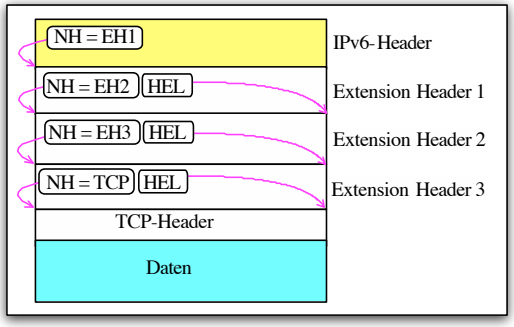


Abb. 7.3-1: Prinzip der Erweiterung des IPv6-Header
EH: Extension Header, NH: Next Header, HEL: Hdr Ext Len = Header Extension Length

Routing Header

Beispiele: In Abb. 7.3-2a folgt ein Routing Header dem IPv6-Header, in dem wiederum ein Verweis `NH` (`Next Header`) enthalten ist, der auf den TCP-Header verweist. Abb. 7.3-2b zeigt die mehrfache Verschachtelung von Erweiterungs-Headern. Der IPv6-Header verweist zuerst auf den `Routing Header`, dieser wiederum verweist auf den `Fragment Header` und dieser schließlich auf den TCP-Header mit den folgenden Daten.

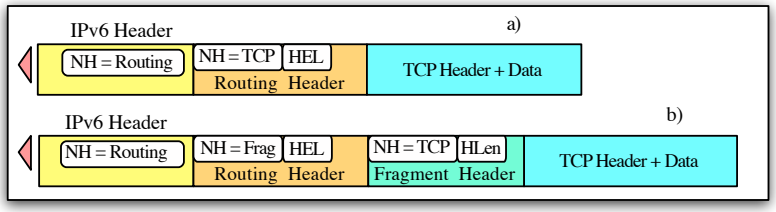


Abb. 7.3-2: Beispiel für die Erweiterung des IPv6-Header: a) mit nur einem Erweiterungs-Header, b) mit mehreren verschachtelten Erweiterungs-Headern NH: Next Header, HEL: Header Extension Length

Nach dem IPv6-Header kann im Allgemeinen entweder

- ein Erweiterungs-Header oder
- der Header von TCP bzw. von UDP, eines Routing-Protokolls bzw. eines sonstigen Protokolls (z.B. von IPv4 bei *IPv4 over IPv6* [Abb. 9.7-1]) folgen.

Im Protokoll IPv6 sind folgende Erweiterungs-Header[1] vorgesehen:

- `Hop-by-Hop Options Header`,
- `Routing Header`,
- `Fragment Header`,
- `Destination Options Header`,
- `Authentication Header`,
- `Encapsulation Security Payload`,
- `Mobility Header`.

Jeder Erweiterungs-Header sollte in einem Paket nur einmal enthalten sein. Nur der `Destination Options Header` kann maximal zweimal vorkommen. Werden mehrere Erweiterungs-Header in einem IPv6-Paket eingesetzt, so wird eine festgelegte Reihenfolge vorgeschlagen. Abb. 7.3-3 zeigt sie. Hier wurde ein Sonderfall angenommen, in dem ein IPv6-Paket sämtliche Erweiterungs-Header enthält.

Hop-by-Hop Options Header

Der `Hop-by-Hop Options Header` enthält sog. *Type-Length-Value*-Angaben (kurz *TLV-Angaben*), die als *Optionen* bezeichnet werden. Da diese TLV-Angaben in jedem Router (Zwischensystem) unterwegs interpretiert werden, muss dieser Header direkt nach dem IPv6-Header folgen. Dadurch lässt sich die für Paketvermittlung notwendige Zeit in Routern reduzieren.

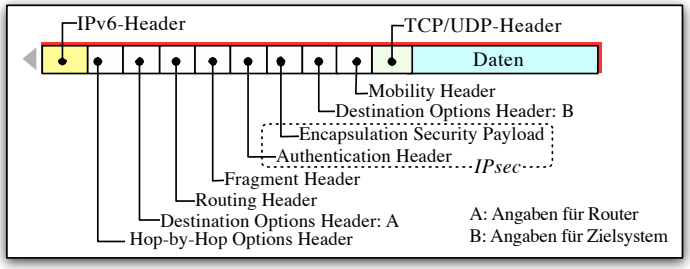


Abb. 7.3-3: IPv6-Paket mit allen Erweiterungs-Headern in der vorgeschriebenen Reihenfolge Destination Options Header

Destination Options Header

Der `Destination Options Header` kann in einem IPv6-Paket zweimal vorkommen. Er enthält die TLV-Angaben sowohl für die Router als auch für das Zielsystem. Enthält ein *Destination Options Header* die TLV-Angaben für Router, so folgt dieser Header direkt nach dem `Hop-by-Hop Options Header`. Die TLV-Angaben für das Zielsystem werden in einem anderen `Destination Options Header` transportiert, der möglichst am Ende in der Header-Reihenfolge positioniert werden soll.

Routing Header

Im *Routing Header* (RH) wird eine Liste von Routern bzw. von anderen Zwischensystemen angegeben, die das zu übermittelnde Paket unterwegs 'besuchen' muss [Abb. 7.6-1]. Es gibt bereits mehrere RH-Typen. Mit `RH Type 0` wurde in RFC 2460 zunächst ein IPv6-basiertes Source-Routing vorgeschlagen, aufgrund von Sicherheitsüberlegungen aber [RFC 5095] wieder verworfen. Wir wollen auf das Konzept dennoch in Abschnitt 7.6 eingehen. `RH Type 2` wird bei Mobile IPv6 verwendet [Abschnitt 15.4]. Die RH-Typen 1, 3 und 4 wurden nur in Internet Drafts vorgeschlagen.

[1] Eine Liste kompletter Angaben findet sich unter `http://www.iana.org/assignments/ipv6-parameters/ipv6-parameters.xhtml`.

Fragment Header

Mittels des `Fragment Header` ist es der Quelle von IPv6-Paketen möglich, ein langes Paket (länger als die zulässige *Path-MTU*) auf eine Reihe von Teilpaketen (Fragmenten) aufzuteilen. *Fragment Header* enthält auch die benötigten Steuerungsangaben, um eine Folge von Fragmenten am Zielsystem wieder zu einem langen Paket zusammenzusetzen.

Authentication Header

`Authentication Header` (AH) kann eingesetzt werden, um folgende Sicherheitsdienste zu realisieren[2]:

- *Authentisierung der Datenquelle*, um feststellen zu können, ob die Daten vom wahren (gültigen) Quellrechner stammen.
- *Überprüfung der Datenintegrität*, um eine mögliche Verfälschung der Frames auf dem Übertragungsweg festzustellen.

Encapsulation Security Payload

Unter *Encapsulation Security Payload* (`ESP`) ist eigentlich ein Frame zu verstehen, das sich aus einem Header und einem Trailer zusammensetzt. Mit `ESP` können die Sicherheitsdienste *Vertraulichkeit*, *Authentisierung* und *Überprüfung der Datenintegrität* realisiert werden. Im Vergleich zu `AH` wird mit `ESP` zusätzlich die Vertraulichkeit durch eine Verschlüsselung gewährleistet. `ESP` kann eigenständig oder kombiniert mit AH eingesetzt werden

Header-Typen

Eine Zusammenstellung der wichtigsten Header-Typen, die dem IPv6-Header folgen können, enthält Tab. 7.3-1.

Header-Typ	Name	EH oder PH
0	Hop-by-Hop Options Header	EH
4	IPv4-Header (IP in IP Encapsulation)	PH
6	TCP- Header	PH
17	UDP-Header	PH
43	Routing Header	EH
44	Fragment Header	EH
47	Generic Encapsulation Header [RFC 2743, 2784]	PH
50	Encapsulation Security Payload	EH
51	Authentication Header	EH
58	ICMP für IPv6	PH
59	kein nächster Header	
60	Destination Options Header	EH
89	OSPF-Header	PH
132	SCTP-Header	PH
135	Mobility Header (Mobile IPv6)	EH
136	UDP-Lite-Header	PH
xx	*Protokoll-Header wie bei IPv4*	PH

Tab. 7.3-1: Mögliche Header-Typen nach dem IPv6-Header
EH: Erweiterungs-Header , PH: Protokoll-Header (wie nach dem IPv4-Header), xx: Protokollnummer; gleiche Angabe wie im Feld Protocol des IPv4-Header, [`http://www.iana.org/assignments/ipv6-parameters`] und [`http://www.iana.org/assignments/protocol-numbers`]

Mobility Header

Abschließend sei erwähnt, dass der `Mobility Header` in RFC 3775 zur Übertragung der *Mobility Options* bei *Mobile IPv6* hinzugefügt wurde. Der `Mobility Header` wird als letzter Extension Header im IPv6-Paket übermittelt.

[2] Es hat sich gezeigt, dass die beiden Erweiterungs-Header `AH` und `ESP` auch bei IPv4 eingesetzt werden können. Dies hat zur Entstehung des Protokolls IPsec für IPv4 geführt [Abschnitt 5.4].

7.4 IPv6-Flexibilität mit Options-Headern

Die grundsätzliche Idee bei der Entwicklung von IPv6 bestand darin, dass der IPv6-Header bei Bedarf mit optionalen Headern, den *Options-Headern*, ergänzt werden kann. Sie ermöglichen, zusätzliche Steuerungsangaben für Zwischen- und Zielsysteme zu übermitteln und damit u.a. das Routing bzw. bestimmte Sicherheitsfunktionen zu unterstützen.

Arten von Options-Header

Es sind zwei Arten von Options-Headern zu unterscheiden:

- `Destination Options Header`
  Dieser Header dient zur Angabe zusätzlicher Steuerungsangaben für den Empfänger des Pakets. Mit Optionen vom Typ 1 werden zusätzliche Steuerinformationen dem Empfänger (z.B. einem Router) des Pakets auf der ersten Zwischenetappe angezeigt. Dagegen sind die Optionen vom `Typ 2` für das endgültige Zielsystem gedacht.
- `Hop-by-Hop Options Header`
  Ein Paket kann auf seinem Weg zum Ziel zusätzliche Informationen für die Zwischenstationen (Hops) enthalten. `Hop-by-Hop Options Header` dient daher zur Übermittlung von Angaben, die von jedem zu passierenden Router bzw. von möglichen Zwischensystemen auf dem Weg zum Datenziel zu beachten sind.

7.4.1 Aufbau von Options-Headern

Die beiden Options-Header, d.h. sowohl `Hop-by-Hop Options Header` als auch `Destination Options Header`, weisen die gleiche in Abb. 7.4-1 gezeigte Struktur auf. Sie besitzen eine variable Länge und lassen sich daher flexibel verwenden.

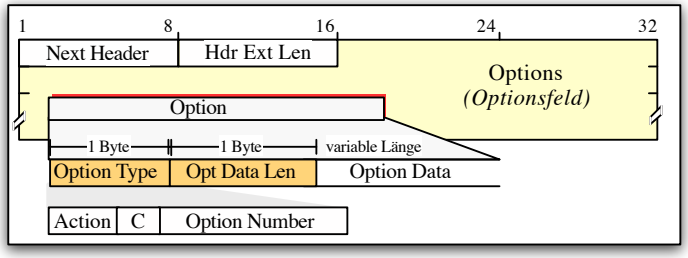


Abb. 7.4-1: Struktur von Options-Headern

Angaben im Options-Header

Die einzelnen Felder im Options-Header haben folgende Bedeutung:

- `Next Header`: Hier wird ein Verweis auf den nächsten Header gemacht [Abb. 7.3-1].
- `Hdr Ext Len` (*Header Extension Length*): Dieses 8-Bit-Feld gibt die Länge des Options-Headers an, die immer ein Vielfaches von 8 Byte betragen muss.
- `Options`: Dieses Feld enthält eine Optionsliste mit folgenden Angaben:
  - `Option Type` (Optionstyp),
  - `Opt Data Len` (*Option Data Length*, Länge von Optionsdaten) und
  - `Option Data` (Optionsdaten).

Der Optionstyp ist ein 8-Bit-Wert, dessen zwei höchstwertige Bit `Action` (Aktion) definieren, die ein System ausführen muss, falls es die betreffende Option nicht kennt. Es sind folgende Aktionen vorgesehen:

- `Action = 00` sorgt dafür, dass diese Option übersprungen wird und das betreffende System zur Bearbeitung der nächsten Option übergeht.
- `Action = 01` bestimmt, dass das gesamte Paket verworfen und keine Fehlermeldung an den Absender des Pakets gesendet wird.
- `Action = 10` veranlasst, dass das gesamte Paket verworfen und eine Fehlermeldung (als ICMPv6-Nachricht `Destination Unreachable`) an den Absender gesendet wird, falls es sich bei der Zieladresse um eine Multicast-Adresse handelt.
- `Action = 11` legt fest, dass das gesamte Paket verworfen und eine Fehlermeldung an den Absender gesendet wird, falls es sich bei der Zieladresse um keine Multicast-Adresse handelt.

Das Bit `C` zeigt an, ob die nachfolgenden Optionsdaten auf dem Weg zum Ziel verändert werden dürfen oder nicht:

- `C = 0`: verbietet eine Veränderung von Optionsdaten (z.B. in einem Router).
- `C = 1`: erlaubt die Modifikation von Optionsdaten.

Mit der Festlegung von `Action` kann eine wichtige Angabe in Bezug auf die Übertragung sicherheitsrelevanter Daten erfolgen. Der Absender solcher Daten kann festlegen, ob z.B. im Falle einer technisch bedingten oder bösartig herbeigeführten Unterbrechung einer Route Datenpakete verworfen werden sollen oder nicht.

7.4.2 Belegung des Option-Feldes

Das `Option`-Feld [Abb. 7.4-1] innerhalb des `Hop-by-Hop Options Header` bzw. des `Destination Options Header` kann durch die verschiedenen Optionen natürlich auch mit unterschiedlichen Längen belegt werden. Hierbei sind einige Prinzipien der Belegung des Option-Feldes zu beachten: Ein `x`-Byte-Datenfeld sollte vom Header-Anfang um ein Vielfaches von `x` Byte platziert werden. Somit kann die Entfernung der Option von Beginn des `Hop-by-Hop Options Header` bzw. des `Destination Options Header` nach einer der folgenden Belegungsregeln vorgenommen werden: (n*4 + 2) Byte, (n*4 + 3) Byte bzw. (n*8 + 2) Byte (n = 0, 1, ...). Die Länge dieser Header sollte immer ein Vielfaches von 8 Byte betragen.

Um die erwähnten Regeln realisieren zu können, werden zwei Fülloptionen (*Padding Options*) definiert:

- Option `Pad1` mit 1-Byte-Länge : Diese Option wird verwendet, um einen 1-Byte-Eintrag im Option-Feld mit 'künstlichen' Daten zu füllen.
- Option `PadN` mit `N`-Byte-Länge: Diese Option wird verwendet, um einen (`N-2`)-Byte-Eintrag im `Option-Feld` mit 0 Byte (`x'00'`) nach Bedarf zu füllen.

Die Belegung des Option-Feldes wird nun anhand von Beispielen näher dargestellt.

Beispiel 1: Das Option-Feld soll eine Option `X` mit folgenden zwei Datenfeldern enthalten: Das *erste Datenfeld ist 4 Byte* und das *zweite Datenfeld 8 Byte lang*. Die Belegung des `Option`-Feldes zeigt Abb. 7.4-2 (vgl. auch Abb. 7.4-1).

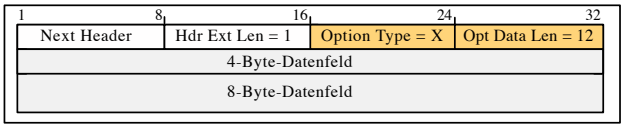


Abb. 7.4-2: Belegung des `Option`-Feldes mit einer 12 Byte langen Option [Abb. 7.4-1]

Hier gilt die Belegungsregel (`n*4 + 2`) Byte (`n = 0`), d.h. die Option beginnt in der Entfernung 2 Byte vom Anfang des Headers.

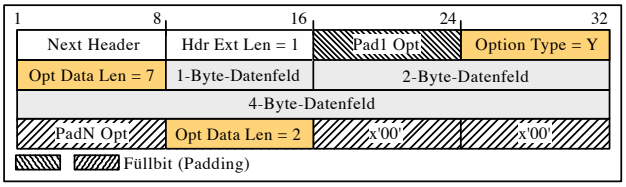


Abb. 7.4-3: Belegung des `Option`-Feldes mit 7-Byte langen Options-Angaben

Beispiel 2: Das `Option`-Feld soll eine Option `Y` mit folgenden drei Datenfeldern enthalten: Das *erste Datenfeld ist 1 Byte*, das *zweite 2 Byte und das dritte 4 Byte lang*. Die Belegung des `Option`-Feldes in diesem Fall zeigt Abb. 7.4-3. Hier gilt die Belegungsregel (`n*4 + 3`) Byte. Die Option beginnt nach 3 Byte ausgehend von Header-Beginn. Um dies zu erreichen, wird das dritte Byte mit einer Option `Pad1` gefüllt, die keine Steuerungsangaben enthält. Das 4-Byte-Datenfeld beginnt nach `n*4` Byte (`n = 2`) vom Anfang des Headers. Um die gesamte Länge des Option Headers auf ein Vielfaches von 8 Byte zu ergänzen, werden die letzten 4 Byte mit `PadN` gefüllt.

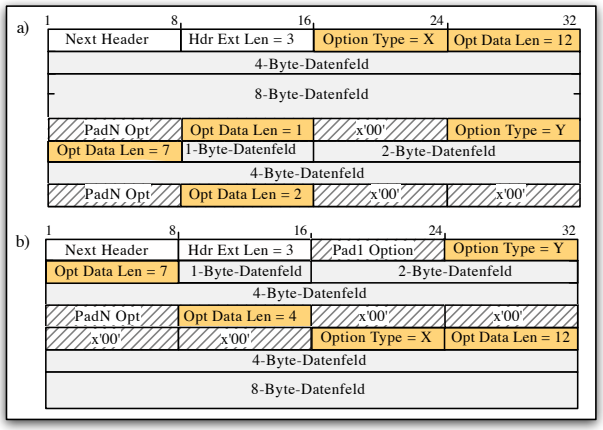


Abb. 7.4-4: `Option`-Feld mit mehreren Option-Typen folgender Belegung: a) zuerst Option `X` und dann Option `Y`, b) zuerst Option `Y` und dann Option `X`

Beispiel 3: `Hop-by-Hop Options Header` bzw. `Destination Options Header` soll die beiden Optionen `X` und `Y` aus den vorherigen Beispielen 1 und 2 enthalten. Abb. 7.4-4a illustriert den Fall, wenn zuerst Option `X` und dann Option `Y` im `Option`-Feld platziert werden. Es sei vermerkt, dass die letzten vier Byte mit `PadN` gefüllt werden, um die gesamte Gesamtlänge auf ein Vielfaches von 8 Byte zu ergänzen.

Abb. 7.4-4b zeigt die Situation, in der zuerst die Option `Y` und dann die Option `X` im `Option`-Feld platziert wird. Da ein `8`-Byte-Datenfeld um ein Vielfaches von `8` Byte von Header-Anfang platziert werden sollte, wird hier die 4. Zeile mit `PadN` gefüllt.

7.5 Einsatz von Jumbo Payload

Ebenso wie bei IPv4 stehen auch nur 16 Bit im Feld `Payload Length` des IPv6-Header zur Verfügung [Abb. 7.2-1], um die Länge von Nutzdaten anzugeben. Dadurch können die Nutzdaten nicht mehr als 65.535 Byte betragen. Falls größere Mengen von Nutzdaten in einem IPv6-Paket übertragen werden sollen, kann dies unter Einsatz der sog. *Jumbo Payload Option* im `Hop-by-Hop Options Header` markiert werden [RFC 2675]. Man spricht in diesem Zusammenhang von einem *IPv6-Jumbogram*. Abb. 7.5-1 illustriert die Struktur von `Hop-by-Hop Options Header` mit `Jumbo Payload Option`.

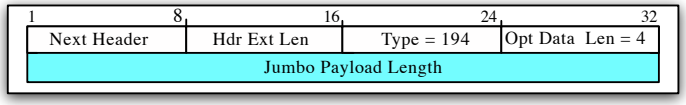


Abb. 7.5-1: `Hop-by-Hop Options Header` mit Angabe der Jumbo Payload Length (JPL)

Jumbograms

Vorausgesetzt, die MTU des unterliegenden Netzwerks (wie z.B. Gigabit-Ethernet) ist in der Lage, diese Datenmenge in einem Frame zu transportieren, kann `Hop-by-Hop Options Header` verwendet und die Paketlänge wird als `Jumbo Payload Length` angegeben. Dies veranschaulicht Abb. 7.5-2.

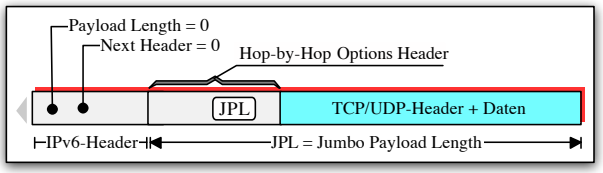


Abb. 7.5-2: IPv6-Paket mit Jumbo Payload

Wie hier ersichtlich ist, müssen die Angaben `Payload Length` und `Next Header` im IPv6-Header den Wert `0` enthalten.

7.6 Source Routing bei IPv6

Laut RFC 5095 ist der Einsatz von Source Routing in IPv6-Netzen untersagt. Genauer gesagt verbietet es Endsystemen, die Informationen die *Source Routing* Angaben zu beachten, und schlägt hingegen vor, solche IPv6-Pakete an der Firewall auszufiltern. Trotzdem wollen wir auf das Konzept des Source Routing eingehen, d.h. wie ein IPv6 seinen Weg durch das Netz von der Quelle (*Source*) zum Ziel (*Destination*) finden kann. Um ein derartiges Routing zu realisieren, kann der IPv6-Header mit einem `Routing Header` erweitert werden [Abb. 7.3-2]. Abb. 7.6-1a zeigt den Aufbau des `Routing Header`.

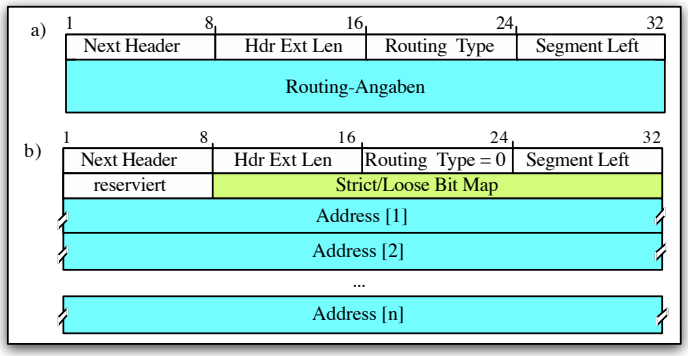


Abb. 7.6-1: Aufbau des Routing Header: a) allgemeine Struktur, b) Header mit Routing-Typ 0

Source Routing vom Typ 0

Mit der Angabe `Routing Type` sind unterschiedliche Source-Routing-Varianten möglich. Es wurde nur das `Source Routing` vom Typ 0 (`Routing Type = 0`) festgelegt; der *Routing Type 2* ist für Mobile IPv6 vorgesehen [Abschnitt 14.4]. Die Struktur des `Routing Header` bei diesem Routing-Typ zeigt Abb. 7.6-1b.

Die einzelnen Angaben im Routing Header beim Source Routing vom Typ 0 haben folgende Bedeutung:

- `Segment Left`: Hier wird die Anzahl der restlichen *Routing-Segmente*, die das betreffende Paket bis zum Ziel durchlaufen muss, eingetragen. Ein Routing-Segment ist als Hop bzw. Route-Abschnitt zu sehen.
- `Strict/Loose Bit Map`: Dies ist eine Bitfolge `b0, b1, ..., bi, ..., b23`, in der jedes Bit einem Routing-Segment entspricht, was eine Art *Route-Spezifikation* darstellt. Mittels dieses Felds entsteht die Möglichkeit, eine Teilstrecke einer Route fest vorzuschreiben bzw. deren Auswahl dem Router zu überlassen [Abb. 7.6-2 und Abb. 7.6-3]. Mit `bi = 1` wird darauf verwiesen, dass Router *i* ein direkter Nachbar von Router `i-1` ist. Dies bedeutet, dass Router *i-1* das Paket direkt an Router *i* adressieren muss. Der Fall `bi = 0` bedeutet nur, dass Router *i* kein direkter 'Nachbar'-Router ist, sondern Router *i* das nächste Ziel des Pakets darstellt. Router `i-1` leitet das Paket nach seiner Routing-Tabelle zum Router `i` als Ziel weiter.
- `Address [i]`: Angegeben wird die IPv6-Adresse des Routers `i` bzw. des Zielsystems.

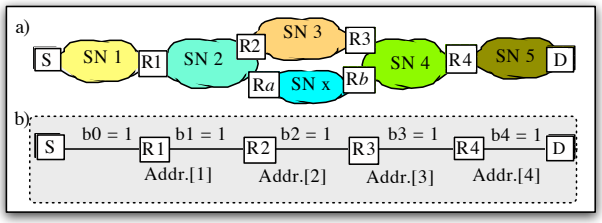


Abb. 7.6-2: Vollkommen festgelegte Route bei der Ende-zu-Ende-Kommunikation: a) Verbund von Subnetzen, b) Route mit festgelegten Routing-Abschnitten
S: Source, D: Destination, R: Router, SN: Subnetz

Strict Source Route

Beispiel: Abb. 7.6-2 illustriert, wie eine Ende-zu-Ende-Route mittels des Felds `Strict/Loose Bit Map` vollständig vorgeschrieben werden kann: *Strict Source Rou-*

te. In diesem Fall ist `bi = 1, i = 1, 2, 3, 4`. Das Feld `Address A[i]` wird vom Router `i` gelesen und enthält die IPv6-Adresse des nächsten Routers `i+1`.

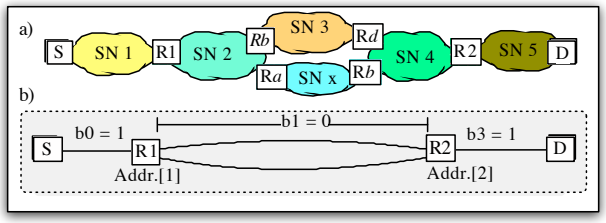


Abb. 7.6-3: Teilweise festgelegte Route bei der Ende-zu-Ende-Kommunikation:
a) physikalischer Verbund von Subnetzen,
b) Route mit zwei festgelegten und einem freien Routing-Abschnitt
SS: Source, D: Destination, R: Router, SN: Subnetz

Loose Source Route

Beispiel: Abb. 7.6-3 zeigt eine teilweise festgelegte Ende-zu-Ende-Route: *Loose Source Route*. Da `b1 = 0` ist, bedeutet dies, dass der Router *R2* kein direkter Nachbar-Router des Routers *R1* ist. Von *R1* zu *R2* wird das Paket geroutet, d.h. zwischen *R1* und *R2* existiert ein freier (nicht festgelegter) Routing-Abschnitt. Das Feld `Address A[1]` wird von Router `R1` gelesen und als die IPv6-Adresse des nächsten Ziels interpretiert.

7.7 Fragmentierung langer IPv6-Pakete

Durch die Verwendung eines `Fragment Headers` kann ein IPv6-Paket, dessen Länge den Wert der möglichen Path-MTU [Abb. 2.7-5] überschreitet, auf eine Reihe zusammenhängender Teile (sog. *Fragment-Pakete*) aufgeteilt werden. Die einzelnen Fragment-Pakete können selbstständig übermittelt werden. Diesen Vorgang bezeichnet man als *Fragmentierung*. Die Fragmentierung von IPv6-Paketen kann nur bei der Quelle dieser Pakete erfolgen. Im Gegensatz dazu kann die Fragmentierung langer IPv4-Pakete auch unterwegs in Routern stattfinden.

Fragment Header

Liegt ein derart langes IPv6-Paket bei der Quelle vor, dass der vereinbarte MTU-Wert überschritten wird, kann dieses Paket als Folge von mehreren und kleineren Fragment-Paketen übermittelt werden. Hierfür wird der Erweiterungs-Header `Fragment Header` verwendet. Dessen Struktur zeigt Abb. 7.7-1.

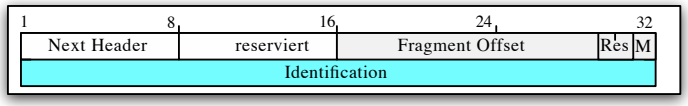


Abb. 7.7-1: Struktur des Fragment Header
Res: Reserviert

Die einzelnen Steuerungsangaben im Fragment Header haben folgende Bedeutung:

- `Fragment Offset`: Dieses 13-Bit-Feld gibt den Abstand (*Offset*) des Datensegments in Anzahl von je 8 Byte ab Datenbeginn an.

- `M-Flag`: Mit `M`-Flag wird markiert, ob es sich um das letzte Fragment-Paket handelt. Daher wird das letzte Fragment-Paket mit `M = 0` markiert. In anderen Paketen muss `M = 1` sein.
- `Identification`: Für jedes Paket, das aufgeteilt werden muss, wird eine Identifikation (`Fragment-ID`) generiert.

Fragment-Identifikation

Die Identifikation des Pakets ist in jedem Fragment-Paket enthalten, wodurch es am Ziel möglich ist, die empfangenen Fragment-Pakete zu sammeln und das 'Originalpaket' zu rekonstruieren. Die Fragmentierung eines langen IPv6-Pakets illustriert Abb. 7.7-2.

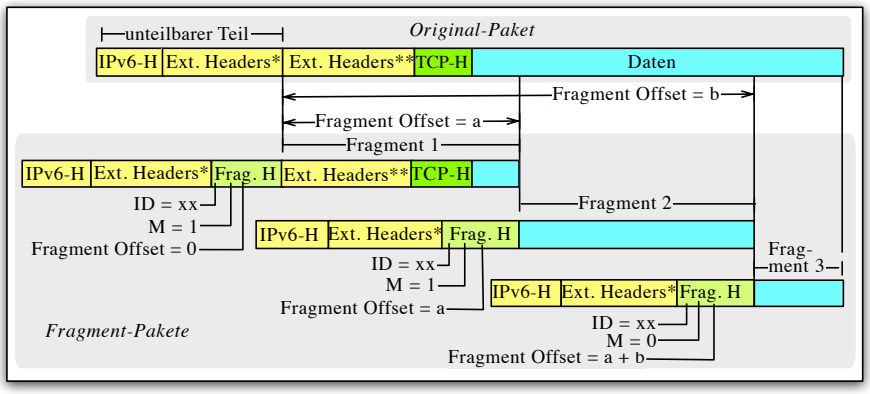


Abb. 7.7-2: Fragmentierung eines langen IPv6-Pakets
Ext. Headers*: Extension Headers, die in Routern interpretiert werden,
Ext. Headers**: Extension Headers, die nur im Endsystem interpretiert werden

Beispiel: Im Originalpaket wird hier ein Teil (als Extension Header* bezeichnet) besonders hervorgehoben, der von den Routern unterwegs interpretiert wird. Dieser Teil darf nicht aufgeteilt werden, er stellt den unteilbaren Teil (*Unfragmentable Part*) des Pakets dar. Sind im IPv6-Paket ein `Routing Header` oder ein `Hop-by-Hop Options Header` vorhanden, so gehören sie zum unteilbaren Teil des Pakets. Der unteilbare Teil muss in jedem Fragment-Paket vorkommen. Der restliche und teilbare Teil des Pakets kann wiederum auf eine Reihe von Fragmenten aufgeteilt werden. Wie aus Abb. 7.7-2 ersichtlich ist, wird jedem Fragment der unteilbare Teil des Originalpakets und anschließend der Fragment Header vorangestellt.

Fragment Offset

Für jedes Paket, das aufgeteilt werden muss, wird eine 32 Bit lange Identifikation (`ID`) generiert, die in jedem Fragment-Paket enthalten sein muss. Mit `M`-Flag wird das letzte Fragment-Paket markiert (`M = 0`). In jedem `Fragment Header` wird der Abstand des Fragments (d.h. *Fragment Offset*) in Anzahl von Byte zum unteilbaren Teil des Originalpakets angegeben. Beim ersten Fragment-Paket ist somit `Fragment Offset = 0`. Mittels der Angaben im `Fragment Header` kann das Originalpaket beim Zielsystem wieder zurückgewonnen werden.

7.8 Aufbau von IPv6-Adressen

Bereits Anfang der 90er-Jahre war festzustellen, dass der auf dem Protokoll IPv4 basierende Adressraum bei dem Weiteren rapiden Internet-Wachstum bald zu knapp sein würde.

Einer der Hauptgründe, ein neues Internetprotokoll zu entwickeln, war die Erweiterung der Adressierung. Die Adresslänge in IPv6 wird auf das Vierfache – jeweils 128 Bit für Quell- und Zieladresse – im Vergleich zu der Adresslänge 32 Bit bei IPv4 erweitert. Somit sind 2^{128} Adressen bei IPv6 verfügbar. Dies bedeutet die Vergrößerung des Adressraums um den Faktor 2^{96}.

Interface-ID

Ähnlich wie bei IPv4 identifiziert eine IPv6-Adresse nicht eine ganze Netzwerkkomponente bzw. Endsystem (Rechner, Router, Layer-3-Switch, ...), sondern lediglich *ein* Interface. Beispielsweise werden einem Router, mit dem mehrere Subnetze miteinander verbunden sind, mehrere IPv6-Adressen zugeteilt, und zwar jeweils eine IPv6-Adresse pro Interface zu einem Subnetz. Ein Rechner stellt daher ein *Multihoming-System* dar. Bei IPv6 unterscheidet man zwischen folgenden Kategorien von Adressen (die in Abschnitt 7.9.1 detaillierter dargestellt werden):

- Unicast-Adressen,
- Multicast-Adressen und
- Anycast-Adressen.

Unicast-Adresse

Unicast-Adressen verwendet man bei der Punkt-zu-Punkt-Kommunikation. Bei dieser häufigsten Adressierungsart sendet ein Quellsystem die Daten an ein direkt angegebenes Zielsystem. Eine Unicast-Adresse identifiziert ein Interface in einem System. Es gibt mehrere Typen von Unicast-Adressen [Tab. 7.8-2].

Multicast-Adresse

Eine *Multicast-Adresse* identifiziert eine Gruppe von Interfaces, die sich in der Regel in verschiedenen Rechnern befinden. Ein Paket mit einer Multicast-Adresse wird an alle Interfaces einer Multicast-Gruppe quasi parallel übermittelt. Daher kann die Quelladresse eines Pakets nie eine Multicast-Adresse sein.

Anycast-Adresse

Eine *Anycast-Adresse* identifiziert ebenfalls eine Gruppe von Interfaces, die sich in verschiedenen Rechnern befinden, wobei über den Anycast-Automatismus immer das nächstliegende Interface adressiert werden soll. Anycast-Adressen ermöglichen den Versand von Paketen über eine festgelegte Stelle an alle Interfaces aus einer Gruppe. Ein Paket mit einer Anycast-Adresse wird zuerst an ein Interface aus der Gruppe (z.B. einen speziellen – dedizierten – Router) übergeben, der das empfangene Paket im nächsten Schritt an ein weiteres Interface aus dieser Gruppe weiterleiten kann. Anycast-Adressen erlauben es, unterschiedliche Rechner zu einer funktionellen Gruppe zusammenzufassen. Eine Anycast-Adresse kann somit nie die Quelladresse eines Pakets sein.

7.8.1 Darstellung von IPv6-Adressen

IPv6-Adresssyntax

Eine IPv6-Adresse wird in 16-Bit-Blöcken dargestellt, wobei jeder 16-Bit-Block in eine aus vier Ziffern bestehende Hexadezimalzahl konvertiert wird und die einzelnen 16-Bit-Blöcke durch Doppelpunkte getrennt sind [Abb. 7.8-1b]:

```
x:x:x:x:x:x:x:x,
```

Eine derartige Darstellung ist als *Doppelpunkt-Hexadezimalnotation* bekannt. IPv6-Adressen weisen hierbei acht Blöcke à 16 Bit bzw. 2 Byte auf, die als 4 hexadezimale Zahlen dargestellt werden. Der Doppelpunkt ist das Trennzeichen für die Blöcke.

Abb.7.8-1 gibt ein Beispiel, wie ausgehend von einer IPv6-Adresse als Binärwert zunächst die Doppelpunkt-Hexadezimalform der Adresse gebildet und anschließend in die sog. *kompaktifizierte* Form überführt wird.

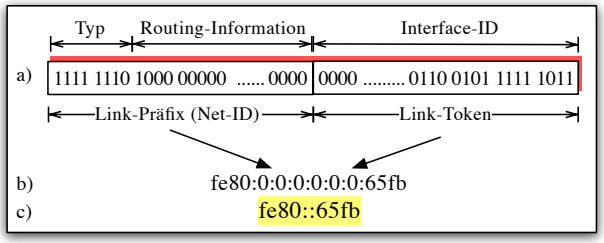


Abb. 7.8-1: Beispiel der Darstellung IPv6-LLU-Adressen a) als Binärwert, b) in Doppelpunkt-Hexadezimalnotation und c) als kompaktifizierte IPv6-Adresse

Durch ihre Bitwertigkeit wird eine IPv6-Adresse in unterschiedliche Bereiche eingeteilt:

- Die ersten Bit legen den *Typ* der Adresse fest. Die Anzahl der Bit ist variabel und kann 3 bis 104 Bit betragen [Tab. 7.8-2].
- Anschließend folgen die Bit für *Identifikation des Netzwerks* bzw. die Routing-Informationen; auch diese Angabe ist variabel, aber immer kleiner als 61 Bit.
- Der unterste Teil der IPv6-Adresse ist zur *Identifikation des Interface* vorgesehen. Ist die (Unicast-) IPv6-Adresse einem physikalischen Interface zugewiesen, beträgt dieser immer 64 Bit.

Wir folgen hier der Konvention von RFC 5952 und benutzen Kleinbuchstaben (im Gegensatz zu den früheren Beispielen entsprechend RFC 4291) für die Hexadezimalzeichen als Repräsentanten der Dezimalzahlen 10 bis 15. Hierdurch kann eine IPv6-Adresse von einer MAC-Adressen unterschieden werden, bei der häufig auch das ':' als Trennzeichen zwischen den Gruppen eingesetzt wird.

Eine IPv6-Adresse kann also folgendermaßen aussehen:

`adcf:0005:0000:0000:0000:0000:0600:fedc`

IPv6-Adressen Kompaktifizierung

Bei der maschinenbezogenen Nutzung und Darstellung wird von *kompaktifizierten IPv6-Adressen* Gebrauch gemacht, die entsprechend folgender Regeln gebildet werden:

- **Regel 1**: Führende Nullen können weggelassen werden:
  `0000` ⇒ `0` ; `0005` ⇒ `5` ; `0600` ⇒ `600`
  `adcf:0005:0000:0000:0000:0000:0600:fedc` ⇒
  `adcf:5:0:0:0:0:600:fedc`
- **Regel 2**: Mehrere aufeinander folgende 16 Bit-Null-Werte können 'unterdrückt' werden:
  `adcf:5:0:0:0:0:600:fedc` ⇒ `adcf:5:::::600:fedc`
- **Regel 3**: *Ein* Block konsekutiver Doppelpunkte ist durch einen Doppelpunkt zu ersetzen:
  `adcf:5:::::600:fedc` ⇒ `adcf:5::600:fedc`

  Hierbei gilt: (a) der längste konsekutive Block ist zu ersetzen, und
  (b) der erste von zwei identisch langen Blöcken wird herangezogen.

Beispiele für IPv6-Adressen

Beispiel 1:

Volle Darstellung	*Kompakte Darstellung*
`adcf:ba56:600:fedc:0:0:0:0`	`adcf:ba56:600:fedc::`
`0:0:0:0:adcf:ba56:600:fedc`	`::adcf:ba56:600:fedc`
`0:0:0:adcf:ba56:0:0:0`	`::adcf:ba6:0:0:0` *oder* `0:0:0:adcf:ba56::0`

Beispiel 2:

Volle Darstellung	*Kompakte Darstellung*
`1080:0:0:0:8:800:200c:417a`	`1080::8:800:200c:417a`
`ff01:0:0:0:0:0:0:101`	`ff01::101`
`0:0:0:0:0:0:0:1`	`::1`
`0:0:0:0:0:0:0:0`	`::`

Das Symbol `'::'` darf nur an einer Seite der Adresse verwendet werden. Die folgende Darstellung ist daher nicht eindeutig:

`::adcf:ba56::` als `0:0:0:adcf:ba56:0:0:0`

Bemerkung: Wird mit einer IPv6-Adresse zusätzlich der Port angegeben, hat sich gemäß RFC 4921 die Schreibweise `[2001:db8::1]:80`, also die IPv6-Adresse in rechteckigen Klammern bewährt.

IPv6-Adresspräfix

Bedeutung des Adresspräfixes

Die von IPv4 bekannte Unterteilung von IPv4-Adressen [Abb. 2.3-1] wird bei IPv6 aufgegriffen und erweitert, sodass sich verschiedene Typen von IPv6-Adressen definieren lassen. Um welchen Adresstyp es sich handelt, bestimmt das sog. *Adresspräfix*, dass eine variable Länge besitzt und durch die ersten (von links gelesenen) Bit bestimmt wird. Durch den Präfixeinsatz kann der ganze Adressraum flexibel aufgeteilt werden.

IPv6-Adresspräfixe

Wie bei der Bildung der Bereiche (Blöcke) von IPv4-Adressen bei der Nutzung der Präfixlängennotation CIDR [Abschnitt 2.5.3] legt man mit dem Adresspräfix bei IPv6 zugleich einen IPv6-Adressbereich (*Netz-ID*; vgl. Tab. 7.8-2) fest [RFC 4921]:

`IPv6-Adresse/Präfixlänge`

IPv6-Adresse/-Präfixlänge

Beispiel: Ein IPv6-Adressbereich mit dem 60 Bit langen Adresspräfix `20010cff0000cd3` (hexadezimal) kann dargestellt werden als

`2001:ceff:0:cd30::/60`

Das bedeutet, dass die ersten 60 Bit als Präfix fest sind und die nachfolgenden 68 Bit beliebig sein können. Bei der Präfixnotation wird die IPv6-Adresse durch die abschließende Angabe von `::` dargestellt.

Die Bildung von IPv6-Adressbereichen wird in Tab. 7.8-1 beispielhaft gezeigt.

IPv6 Adresspräfix	IPv6 Adressbereich/Adressblock
`2000::/3`	001y yyyy yyyy yyyyy ··· yyyy yyy yyyy
`2000::/4`	2xxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx
`2000::/16`	2000:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx
`::/128`	000:0000:0000:0000:0000:0000:0000:0000
`::/96`	000:0000:0000:0000:0000:0000:xxxx:xxxx
`::ffff/96`	000:0000:0000:0000:0000:ffff:xxxx:xxxx

Tab. 7.8-1: Bildung der IPv6-Adressbereiche mittels des Adresspräfixes
y=0,1 (Bit), x={0,f} (hexadezimale Ziffer)

7.8.2 IPv6-Adressensystematik und -Gültigkeitsbereiche

Bei IPv6 wird mittels des Adresspräfixes angegeben, um welchen Adresstyp es sich handelt. Dieser Zusammenhang wird besonders deutlich, wenn wir statt der üblichen, kompaktifizierten IPv6-Adressen deren Bitmuster betrachten, so wie sie für die Verarbeitung in den IPv6-Systemen relevant sind.

Hieraus ergibt sich die in Tab. 7.8-2 deutlich erkennbare Systematik, wobei die IPv6-Adressen gemäß der Wertigkeit der ersten Bit aufgelistet sind:

Typ	Netz-ID	Präfixlänge	Bitmuster
Multicast	`ff`	/8	1111 1111 0000 0000
Solicited-Node MC	`ff02::1:ff`	/104	1111 1111 0000 0010 ... 0001 1111 1111
All-Node MC	`ff02::1`	/128	1111 1111 0000 0010 0000 ... 0000 0001
All-Router MC	`ff01::2`	/1288	1111 1111 0000 0001 0000 ... 0000 0010
Site-local Unicast **SLU**	`fec0`	/10	1111 1110 1100
Link-Local Unicast **LLU**	`fe80`	/10	1111 1110 1000
Unique-local Unicast **ULA**	`fc00`	/7	1111 1100
Well-known Prefix [RFC 6052]	`64:ff9b`	/96	0110 0100 1111 1111 1001 1011 ...
Teredo Prefix *)	`3ffe:831f`	/32	0011 1111 1111 1000 0011 0001 1111 ...
6to4 Adresse (mit IPv4)	`2002:2`	/48	0010 0000 0000 0010 0000 0000 0010 ...
6to4 Adresse (ohne IPv4)	`2002`	/16	0010 0000 0000 0010
Teredo Prefix [RFC 4380]	`2001`	/12	0010 0000 0000 0001
Loopback	`::1`	/128	0000 0000 ... 0001
Unspecified	`::`	/128	0000 0000 ... 0000

Tab. 7.8-2: IPv6-Adressen-Systematik gemäß RFC 4291, außer falls explizit angegeben
*) inoffiziell von Microsoft genutzt

- **Multicast-Adressen**: IPv6-Adressen, bei denen die ersten 8 Bit gesetzt sind (`ff00::/8`), bilden den Bereich der Multicast-Adressen.
- **Lokal gültige Unicast-Adressen**: Mit den IPv6-Bereichen `fec0::/10`, `fe80::/10` und `fc00::/7` folgen Adressen, die innerhalb lokaler Linksegmente einzusetzen sind und nicht ins Internet geroutet werden.
- **Funktionale Adressen**: Speziell für die Zusammenführung von IPv4- und IPv6-Netzen sind einige IPv6-Adressbereiche reserviert, die für diese Zwecke genutzt werden können und routbar sind.
- **Global Unicast Adressen**: Der Bereich der `2000::/3` IPv6-Adressen bildet aktuell den Umfang der routbaren Adressen im IPv6-Internet.
- **Host-IPv6-Adressen**: Die IPv6-Loopback-Adresse `::1` und die unspezifizierte Adresse `::` bilden den Abschluss der IPv6-Adressenhierarchie.

Die Aufteilung des IPv6-Adressraums wird von IANA (*Internet Assigned Numbers Authority*) koordiniert und die immer aktuelle Aufteilung findet man unter `http://www.iana.org/assignments/ipv6-address-space/ipv6-address-space.xhtml`.

Scope von IPv6-Adressen

Der Gültigkeitsbereich, d.h. der *Scope* einer IPv6-Adresse, wird durch ihre Netz-ID festgelegt [Tab. 7.8-2] und kann entsprechend Abb. 7.8-2 in eine lokale Hierarchie eingeteilt werden. Die unspezifische IPv6-Adresse sowie alle LLU-Adressen sind nicht routbar und werden somit auch nicht in die Routing-Tabelle aufgenommen.

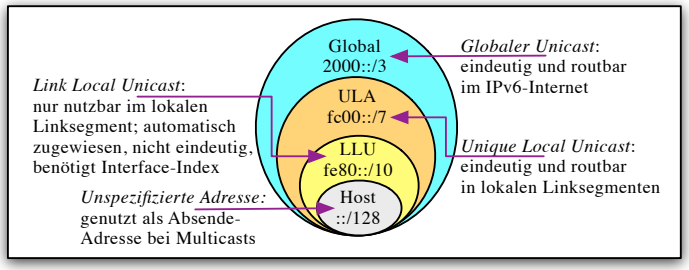


Abb. 7.8-2: Veranschaulichung von IPv6-Unicast-Adressen unterschiedlicher Scopes

Neben den Scopes für Unicast-Adressen wird auch für Multicast-Adressen ein erweiterter Scope definiert [Abb. 7.10-1], der deutlich granularer ist als das Unicast-Pendant.

7.8.3 Interface-ID in IPv6-Adressen

IPv6-Adresse aus MAC-Adresse

In IPv4-Netzen ist folgendes Problem zu lösen: Ein IP-Paket muss zu einem Rechner im LAN abgeschickt werden. Hierfür wird das IP-Paket in einen MAC-frame (*Media Access Control*) eingebettet. In diesem Frame muss die MAC-Adresse des Zielrechners gesetzt werden. Im IP-Paket ist aber nur die IP-Zieladresse enthalten. Um die richtige MAC-Adresse des Zielrechners (d.h. seine physikalische Adresse) zu ermitteln, wird ARP [Abschnitt 2.6.1] in Anspruch genommen.

ARP hat die Aufgabe, die Zuordnung: *IP-Zieladresse* ⇒ *Ziel-MAC-Adresse* zu bestimmen. Auf ARP könnte man verzichten, wenn eine IP-Adresse den Bezug zur entsprechenden physikalischen Adresse hätte. Dieser Ansatz wird bei IPv6-Adressen verfolgt. Als `Interface-ID` in einer IPv6-Adresse [Abb. 7.8-3] kann eine MAC-Adresse verwendet werden. Daher wird die Interface-ID einer IPv6-Adresse für einen Rechner aus dessen *Link-ID* abgeleitet [Abb. 7.8-1], die in der Regel seiner MAC-Adresse entspricht. Zur Ermittlung der MAC-Adresse für IPv6 wird die auf ICMPv6 aufbauende *Neighbor Discovery* [Abschnitt 8.2] genutzt, die die Funktion von ARP ersetzt.

Je nach Implementierung kann die `Interface-ID` einer IPv6-Adresse wie folgt gebildet werden:

- Aus einer *Link-Layer-Adresse* im IEEE-Format EUI-64 (*Extended Unique Identifier*),
- aus einem *zufälliges Bitmuster*, das sich im Laufe der Zeit ändert, um einen gewissen Grad an Anonymität zu bieten,
- als statische bzw. manuell zugewiesene Interface-ID.

Was ist EUI-64?

Bei EUI-64 handelt es sich um das bei IEEE genormte und 64 Bit lange Format für die Adressen von Netzwerkadapterkarten (Interfaces), d.h. für die Link-Layer-Adressen. Daher spricht man auch von *EUI-64-Adressen*. Eine 48 Bit lange MAC-Adresse, die eine physikalische LAN-Adresse darstellt und auch als *IEEE802-Adresse* bezeichnet wird, kann in das EUI-64-Format umgewandelt werden. Abb. 7.8-3 illustriert den Fall. Die

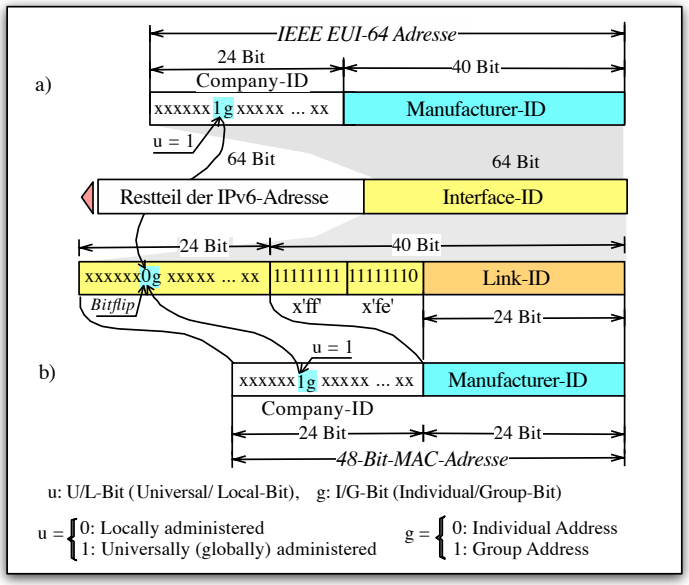


Abb. 7.8-3: Interface-ID in einer IPv6-Adresse als: a) EUI-64-Adresse, b) MAC-Adresse

Interface-ID in einer IPv6-Adresse ist entweder eine EUI-64-Adresse oder eine MAC-Adresse.

EUI-64-Adresse ⇒ Interface-ID

Eine EUI-64-Adresse besteht aus einer 24 Bit langen Company ID (Hersteller-ID), die jedem Hersteller von Adapterkarten bei IEEE eindeutig zugeordnet wurde, und aus einer 40 Bit langen, vom Hersteller festgelegten Identifikation der Adapterkarte (`Manufacturer-ID`). In `Company-ID` haben die Bit `u` und `g` eine besondere Bedeutung. Mit dem Bit `u` wird unterschieden, ob es sich um eine universelle (`u = 1`) oder um eine lokale Adresse (`u = 0`), d.h. eine lokal verwaltete (*locally administered*) Adresse handelt. Das Bit `g` verweist darauf, ob die Adresse eine individuelle Adresse oder eine Gruppenadresse (z.B. Multicast) ist. Wie Abb. 7.8-3a zeigt, werden alle 64 Bit der EUI-64-Adresse als `Interface-ID` übernommen.

Zur Vereinfachung der Administration lokaler IPv6-Adressen wurden entsprechend RFC 4291 beschlossen, das `u`-Bit zu invertieren; also auf `0` zu setzen. Zudem muss die `Interface-ID` in einer IPv6-Adresse nur innerhalb einer privaten Netzstruktur [Abb. 7.9-4] eindeutig sein, d.h. besitzt in der Tat nur eine lokale Bedeutung hat.

MAC-Adresse ⇒ Link-Token

Im Gegensatz zur EUI-64-Adresse ist die `Manufacturer-ID` in 48 Bit MAC-Adressen nur 24 Bit lang. Wie Abb. 7.8-3b zeigt, wird die MAC-Adresse zunächst als *Link-Token* aufbereitet: Der Teil `Company-ID` wird zuerst untergebracht und hierbei das Bit `u` auf `0` gesetzt, Danach folgen zwei Füllbyte `x'ff'` und `x'fe'`. Anschließend wird die `Manufacturer-ID` (als *Link-ID*) eingetragen.

Aus Abb.7.8-3b geht hervor, dass IPv6-Adressen, die direkt aus MAC-Adressen abgeleitet sind, immer die Füllbyte `x'ff'` und `x'fe'` besitzen und hierüber identifizierbar sind.

Link-ID vs. Link-Token

Der für die Bildung der IPv6-Adresse entscheidende Teil der MAC-Adresse ist aber der untere Teil, d.h. die ersten 24 Bit, die auf MAC-Level als `Manufacturer ID`, bei der

IPv6-Nutzung allgemein als *Link-ID* bezeichnet wird. Die Link-ID ist somit der eindeutige Teil des Link-Tokens der IPv6-Adresse, der die niedrigwertigsten 64 Bit umfasst (vgl. RFC 7136).

Beeinträchtigung der Sicherheit

Wie Abb. 7.8-3 zeigt, kann die IPv6-Adresse eines Rechners aus seiner MAC-Adresse eindeutig generiert werden. Daher hat der Rechner mit der gleichen MAC-Adresse, was beim Server in der Regel der Fall ist, immer die gleiche IPv6-Adresse. Da der Datenverkehr im Internet mitgeschnitten werden kann, führt die eben erwähnte Tatsache zur Beeinträchtigung der Sicherheit[3] während der Datenübertragung über öffentliche IP-Netze (wie dem Internet). Um diesen Missbrauch zu unterbinden, wurden zusätzliche Erweiterungen, die sog. *Privacy Extensions*, zunächst in RFC 3041 nachträglich eingeführt [Abschnitt 7.11].

7.8.4 Interface-Index bei Link-Local IPv6-Adressen

Scope-Id

Als Besonderheit bei *Link-Local Unicast-Adressen* ist zu verzeichnen, dass diese Adressen nicht eindeutig sein müssen: Auf unterschiedlichen Linksegmenten kann dieselbe IPv6-LLU-Adresse durchaus mehrfach verwendet werden. So bekommt das Loopback-Interface zusätzlich zur `::1/128` IPv6-Adresse auch noch die LLU-Adresse `fe80::1/10` automatisch zugewiesen und kann darüber angesprochen werden.

Interface-Index = Scope-Id

Zur Bindung einer IPv6-Adresse auf ein Interface wird dieses durch einen *Interface-Index* (*Interface-Idx*) beschrieben. Diese auch als *Scope-Id* bezeichnete Kennung wird normalerweise vom Betriebssystem einem Interface zugewiesen, sobald dieses aktiv wird – bei physikalischen Schnittstellen, sofern ein *Link* vorliegt.

Da (auf der gleichen Netzkomponente) auch ein physikalisches Interface die gleiche LLU aufweisen kann, ist es beim Versenden von IPv6-Paketen notwendig, den Interface-Index bzw. die *Scope-Id* beim Aufbau des Sockets mitzuteilen, was auch in der Regel über seinen symbolischen Namen wie `lo0` bzw. `eth0` erfolgen kann.

Beispiel: Ein Rechner unter Linux kann z.B. die IPv6-LLU-Adresse aus Abb.7.8-1 mittels des Hilfskommandos *ping* wie folgt ansprechen:

```
ping6 -I eth0 fe80::65fb bzw.
ping6 fe80::65fb%eth0
```

sofern das Ethernet-Interface die Bezeichnung `eth0` trägt. Bei MacOS wird das lokale Ethernet-Interface `en0` z.B. wie folgt mit einer SLU-Adresse versehen:

```
ifconfig en0
inet6 fe80::abc:8ff:fea1:65fb%en0 prefixlen 64 scopeid 0x4
```

Hierbei steht '`scopeid 0x4`' für das vierte Interface der Rechners; das Loopback-Interface '`lo`' besitzt '`scopeid 0x1`'.

Bei routbaren IPv6-Adressen ist dies nicht notwendig, da die Zuordnung des Interfaces automatisch über die Routing-Tabelle erfolgt.

[3] Dies gilt auch im Falle von IPsec, da die Authentisierung des Kommunikationspartners über dessen IPv6-Adresse erfolgt.

7.9 Unicast-Adressen bei IPv6

Die Unicast-Adressen werden für normale Punkt-zu-Punkt-Kommunikation verwendet. Unter den Unicast-Adressen sind wiederum folgende Adresstypen zu unterscheiden:

Arten von Unicast-Adressen

- **Globale Unicast-Adressen** (*Global Unicast Addresses*)
  Die Bezeichnung global besagt, dass diese Art von IPv6-Adressen weltweite Gültigkeit hat. Diese IPv6-Adressen haben das 3 Bit lange Präfix `001` [Tab. 7.8-2], sodass man hier auch vom Adressbereich `2000::/3` spricht. Die globalen Unicast-Adressen sind mit öffentlichen IPv4-Adressen vergleichbar. Auf diese IPv6-Adressen geht Abschnitt 7.9.1 detailliert ein.
- **Lokale Unicast-Adressen**
  Es handelt sich hierbei um Unicast-Adressen von nur lokaler Bedeutung. Zu dieser Klasse gehören:
  - *Link-Local Unicast Addresses* LLU sowie
  - *Unique Local Unicast Addresses* ULA und
  - *Site-Local Unicast Addresses* SLU.

  SLU-Adressen sind aber laut RFC 3879 nicht mehr einzusetzen, und daher gehen wir im Weiteren hierauf auch nicht näher ein.
- **IPv4-Kompatibilitätsadressen**
  Diese Unicast-Adressen wurden eingeführt, um die Migration zum IPv6-Einsatz und insbesondere den Fall zu unterstützen, wenn die beiden Protokolle IPv4 und IPv6 in einem Netz implementiert werden sollen. Die IPv4-Kompatibilitätsadressen werden in Abschnitt 7.9.4 besprochen.
- **Spezielle IPv6-Adressen**
  - *Loopback-IPv6-Adresse*
    Diese Adresse ist `0:0:0:0:0:0:0:1` bzw. kurz `::1/128`. Sie entspricht der Loopback-Adresse `127.0.0.1` von IPv4 und wird in IPv6-Paketen genutzt, die zwischen den Programmen innerhalb eines Rechners (z.B. beim Testen) ausgetauscht werden. Mittels dieser Adresse kann daher ein Rechner die IPv6-Pakete an sich selbst senden. An die Loopback-Adresse gerichtete Pakete werden nie nach außen weitergeleitet. Somit kann diese Adresse weder Quell- noch Zieladresse in IPv6-Paketen sein, die einen Rechner bzw. einen Router verlassen.
  - *Nicht spezifizierte Adresse* (*Unspecified Address*)
    Diese Adresse ist `0:0:0:0:0:0:0:0` bzw. kurz `::/128` und entspricht der nicht spezifizierten Adresse `0.0.0.0` von IPv4. Sie zeigt an, dass keine Adresse vorhanden ist. Die nicht spezifizierte Adresse wird nie einem Interface zugewiesen bzw. als Zieladresse verwendet. Sie wird aber in den Fällen als Quelladresse für IPv6-Pakete verwendet, wo versucht wird, die Eindeutigkeit einer vorläufigen IP-Adresse zu bestätigen.

> *Link-Local-Adressen* (LLU) dienen IPv6 zum korrekten Protokollablauf. Sie sind nicht für allgemeine Kommunikationszwecke zu benutzen. In privaten IPv6-Netzen sind hierfür die *Unique-Local-Adressen* (ULA) vorgesehen.

7.9.1 Globale Unicast-Adressen

Die globalen Unicast-Adressen (*Global Unicast Addresses*) von IPv6 sind mit öffentlichen IPv4-Adressen vergleichbar und im gesamten IPv6-Internet eindeutig. Den Aufbau von globalen Unicast-Adressen zeigt Abb. 7.9-1. Abb. 7.9-1a veranschaulicht die allgemeine Struktur einer globalen Unicast-Adresse, wie sie in RFC 3587 definiert ist. Abb. 7.9-1b zeigt die Struktur von globalen Unicast-Adressen, die zurzeit von der IANA zugewiesen sind. Diese IPv6-Adressen haben das Präfix `001` (binär) bzw. `2000::/3` (hexadezimal), und man bezeichnet sie auch als `2000::/3`-Adressen. Das Adresspräfix für momentan zugewiesene globale IPv6-Adressen lautet daher `2000::/3`.

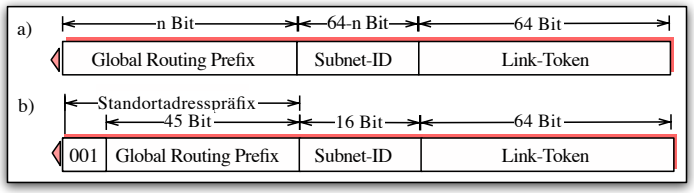


Abb. 7.9-1: Aufbau von globalen Unicast-Adressen: a) allgemeine Struktur (nach RFC 3587), b) Adressen mit Präfix 2000::/3

Die weiteren Angaben in globalen Unicast-Adressen sind:

- `Global Routing Prefix` (GRP)
  GRP kann hierarchisch strukturiert werden und wird verwendet, um die Route zu einer bestimmten Organisation anzugeben. Auf die Bedeutung von GRP wird im Weiteren näher eingegangen [Abb. 7.9-3, Abb. 7.9-4].
- `Subnet-ID`
  Als Subnet-ID wird die Identifikation eines Subnetzes innerhalb einer Organisation angegeben. `Subnet-ID` kann weiter strukturiert werden, um eine Subnetz-Hierarchie innerhalb eines physikalisch großen Netzes adressieren zu können, sodass man daher von privater Struktur sprechen kann.
- `Link-Token` bzw. `Interface-ID`
  Der Link-Token dient zur Zuordnung des physikalischen oder logischen Interfaces zur IPv6-Adresse (im Linksegment); kann also quasi als 'logische Netzadresse' eines Ports interpretiert werden [Abb. 7.9-4]. Wurde der Link-Token aus der MAC-Adresse abgeleitet, spricht man auch von einer *Interface-ID*.

Reservierte globale IPv6-Adressen

Der Adressraum für globale IPv6-Adressen wird von der IANA vorgegeben[4] und umfasst neben den eigentlichen routbaren IPv6-Adressen auch solche, die für Testzwecke reserviert sind. Insbesondere ist das Netz `2001:db8::/32` für Dokumentationszwecke, z.B. innerhalb dieses Buches vorgesehen und somit nicht vergeben bzw. allgemein nutzbar.

Nachteil von IPv4-Adressen

Exkurs: Routing-Probleme bei IPv4-Adressen
Eines der größten Probleme bei IPv4 ist der Umfang von Routing-Tabellen in großen Netzen und die damit verbundenen Leistungseinbußen. Dies ergibt sich aus der klassenbasierten Aufteilung der IPv4-Adressen, was jedoch durch die Einführung von *Classless*

[4] siehe: `http://www.iana.org/assignments/ipv6-unicast-address-assignments/ipv6-unicast-address-assignments.xhtml` und `http://www.iana.org/assignments/ipv6-address-space/ipv6-address-space.xhtml`

Inter-Domain Routing (CIDR) abgemildert werden konnte [Abschnitt 2.5.3]. Die Struktur von IPv4-Adressen wurde bereits in Abb. 2.3-1 dargestellt.

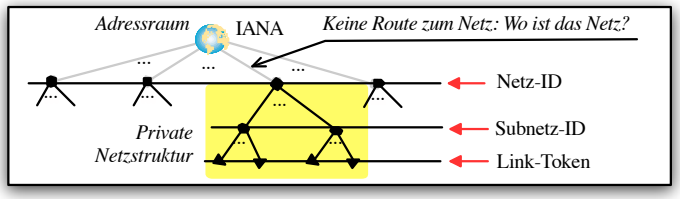


Abb. 7.9-2: Aufteilung des Adressraums bei IPv4-Adressen

Bei IPv4-Adressen wird der ganze Adressraum direkt auf die einzelnen Netze aufgeteilt. Wie Abb. 7.9-2 zeigt, haben IPv4-Adressen den Nachteil, dass sie keine Hierarchie in dem Sinne bilden, dass es möglich wäre, hieraus auf die geographische Lokation eines Netzes zu schließen bzw. unmittelbar aus der IP-Adresse eine Route abzuleiten. Dies hat eine negative Auswirkung auf das Routing und führt zum überproportionalen Anwachsen der Routing-Tabellen im Internet.

Global Routing Prefix GRP bei IPv6

Global Unicast Addresses mit dem Präfix `2000::/3` entsprechen den sog. *Aggregatable Global Unicast Addresses* (*AGU-Adressen*), die vorher in RFC 2374 spezifiziert wurden.

RFC 3587 ersetzt RFC 2374

In AGU-Adressen wurde das Präfix, das dem GRP entspricht, vorher auf TLA (*Top Level Aggregator*) und NLA (`Next Level Aggregator`) aufgeteilt. Dies hat sich aber im Laufe der Zeit als zu starr und zu wenig flexibel erwiesen. Nach RFC 3587 wurde die hierarchische TLA/NLA-Struktur von AGU-Adressen durch GRP ersetzt. Daher hat die TLA/NLA-Struktur heute nur eine historische Bedeutung. GRP kann aber hierarchisch flexibler strukturiert werden. Um die Bedeutung von GRP zu verdeutlichen, soll nun der Nachteil von IPv4-Adressen näher erläutert werden.

Bedeutung des Routing-Präfixes bei IPv6

Durch eine mehrstufige Strukturierung von globalen Unicast-Adressen lässt sich das Anwachsen der Routing-Tabellen in noch akzeptablen Grenzen halten und damit auch die Verzögerung der Pakete im Internet. Dem eben geschilderten Problem versucht man bei IPv6 durch eine hierarchische Strukturierung des globalen Routing-Präfixes (*GRP*) zu begegnen. Abb. 7.9-3 bringt dies zum Ausdruck.

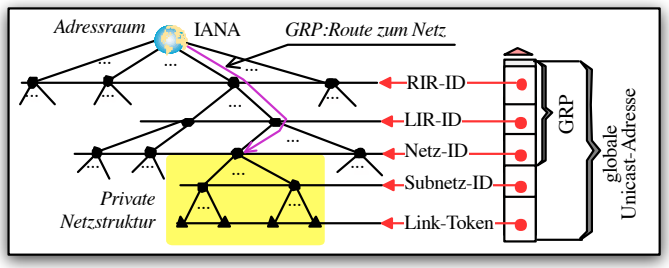


Abb. 7.9-3: Bedeutung von GRP (*Global Routing Prefix*) in IPv6-Adressen
RIR: Regional Internet Registry, LIR: Local Internet Registry

Mittels der hierarchischen Struktur von GRP, d.h. durch die Aufteilung auf `RIR-ID`, `LIR-ID` und `Netz-ID` lässt sich die öffentliche Internetstruktur recht gut gliedern. `GRP` stellt also eine Route dar, die einer Organisation zugewiesen ist. Durch die weitere Strukturierung von globalen Unicast-Adressen, d.h. durch die Angabe von *Subnetz-ID* lässt sich die private Netzstruktur einer Organisation hierarchisch strukturieren.

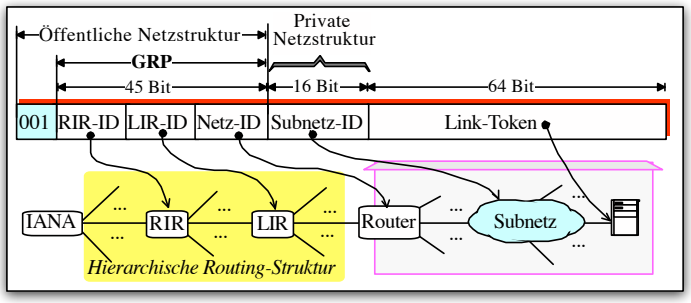


Abb. 7.9-4: Prinzip der Strukturierung von GRP (*Global Routing Prefix*)
Abkürzungen wie in Abb. 7.9-3

Strukturierung von GRP

Abb. 7.9-4 zeigt, wie GRP eine hierarchische Routing-Struktur schafft. Mittels GRP wird somit eine bestimmten Organisation festgelegt.

Die Kombination aus dem festen Präfix `001` und dem aus 45 Bit bestehenden GRP kann als *Standortadresspräfix* angesehen werden und verweist auf den Standort einer Organisation. Nach GRP leiten die Router im IPv6-Internet den Datenverkehr an den Router am Eingang zu einer Organisation weiter. Aufgrund der so strukturierten globalen Unicast-Adressen lässt sich Routing im IPv6-Internet sehr vereinfachen. Wie in Abb. 7.9-4 ersichtlich ist, verweist die Subnetz-ID auf das entsprechende Subnetz am Standort einer Organisation. Mit dem Link-Token wird eine Schnittstelle (ein Port) im Subnetz gekennzeichnet.

Aggregation von Routen

GRP ermöglicht auch die Aggregation von Routen. Das in Abb.7.9-5 dargestellte Beispiel soll dies näher erläutern.

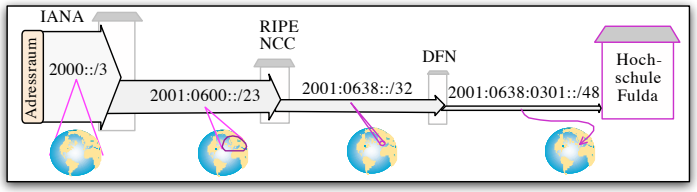


Abb. 7.9-5: Beispiel für die Aggregation von Routen mittels GRP
DFN: Deutsches Forschungsnetz

Das IPv6-Adresspräfix kann hier auch als aggregierte Route wie folgt interpretiert werden:

`2001:0600::/23`	aggregierte Route für den Bereich des RIPE NCC
`2001:0638::/32`	aggregierte Route zum DFN
`2001:0638:0504::/48`	aggregierte Route zur Hochschule Fulda.

9 Migration zum IPv6-Einsatz

Koexistenz von IPv4 und IPv6

Die Umstellung aller Rechner, in denen das herkömmliche Internetprotokoll IPv4 verwendet wird, auf das neue Protokoll IPv6 kann nicht auf einen Schlag geschehen. Dazu sind weltweit viel zu viele Rechner mit IPv4 installiert. Der Schlüssel zur Einführung von IPv6 liegt in der langfristigen und kostengünstigen Migration. Es muss mit einer Übergangszeit gerechnet werden, während der IPv4 und IPv6 parallel eingesetzt werden. Daher benötigt man bestimmte Ansätze und Systemlösungen, um die Integration von IPv4- und IPv6-Netzen zu ermöglichen. Nahezu alle aktuellen IT-Systeme unterstützten sowohl IPv4 als auch IPv6, weshalb man an dieser Stelle von *Dual-Stack-Systemen* spricht. Die Umstellung von IPv4 auf IPv6 ist also nicht mehr in erster Linie ein technisches Problem der Systeme wie Rechner, Server und Router, sondern mehr ein organisatorisches, d.h. ob und wie ein *IPv4-Netz* durch ein *IPv6-Netz* ergänzt oder abgelöst werden soll.

Überblick über das Kapitel

Dieses Kapitel gibt einen Überblick über verschiedene Ansätze und Systemlösungen, um die Koexistenz von IPv4 und IPv6 in verschiedenen Netzstrukturen zu ermöglichen. Nach einem Überblick über die unterschiedlichen Strategien zur Koexistenz von IPv4- und IPv6-Netzen in Abschnitt 9.1 widmet sich Abschnitt 9.2 der Darstellung verschiedener Dual-Stack-Technologien, über die IPv4 und IPv6 gemeinsam genutzt werden können. Die Möglichkeiten der Übermittlung von IPv6-Paketen speziell über IPv4-Netze, d.h. das *Tunneling*, werden in Abschnitt 9.3 besprochen. Auf das Konzept von *6to4* geht Abschnitt 9.4 ein. Abschnitt 9.5 erläutert die IPv6-Kommunikation über IPv4-Netze mit ISATAP. Die Übermittlung von IPv6-Paketen in IPv4-Netzen mit NAT nach dem als *Teredo* bezeichneten Verfahren erläutert Abschnitt 9.6. Die Integration der IPv4- und IPv6-Netze mithilfe der Translation *IPv4* $\Leftrightarrow$ *IPv6* wird in Abschnitt 9.7 dargestellt. Schließlich runden wir das Thema mit der Darstellung der aktuellen NAT64 und DNS64 Vorschläge in Abschnitt 9.8 ab, um die prinzipiellen Lösungen und Probleme nochmals kurz in den Schlussbemerkungen aufzugreifen.

Ziel dieses Kapitels

In diesem Kapitel werden u.a. folgende Fragen beantwortet:

- Wie kann man sich die beiden Protokolle IPv4 und IPv6 in einem Rechner bzw. in einem Router vorstellen?
- Welche Bedeutung hat der *Dual-Stack-Betrieb* von IPv4 und IPv6 für die bestehenden Netze?
- Welche Möglichkeiten der Koexistenz von IPv4 und IPv6 gibt es?
- Wie kann die Kommunikation nach IPv6 über IPv4-Netze realisiert werden?
- Wie kann der Zugang zum IPv6-Internet bereits heute erfolgen?
- Wie lassen sich die sog. *IPv6-Sites* über IPv4-Netze vernetzen?
- Wie können die Rechner in IPv6-Netzen auf das bestehende IPv4-Internet zugreifen?
- Welche Möglichkeiten bringt der Einsatz von IPv6 in Netzwerken mit privaten IPv4-Adressen und mit NAT?
- Wie erfolgt die Translation *IPv4* $\Leftrightarrow$ *IPv6* und was ermöglicht sie?
- Welche Lösungen für die Koexistenz von IPv4 und IPv6 liefert NAT64 mit seinem *stateful Address Mapping* und warum wird hierzu DNS64 benötigt?

9.1 Arten der Koexistenz von IPv6 und IPv4

Sieht man von einer kleinen Minderheit alter Rechner und Router ab, die nur das Protokoll IPv4 unterstützen, so sind alle aktuellen Systeme heute mit IPv4 und IPv6 ausgestattet: Sie sind also *Dual-Stack*-Systeme.

Um eine Vorstellung über die Protokollarchitektur eines Dual-Stack-Rechners und eines Dual-Stack-Routers zu vermitteln, zeigt Abb.9.1-1 die beiden Protokolle IPv4 und IPv6 im Schichtenmodell. Die hier dargestellte Struktur kann als allgemeine Protokollarchitektur eines Dual-Stack-Rechners angesehen werden. Man findet sie z.B. in einem Server unter Linux mit Kernel 3.2 vor, der als Dual-Stack-Rechner bzw. als Dual-Stack-Router dienen kann.

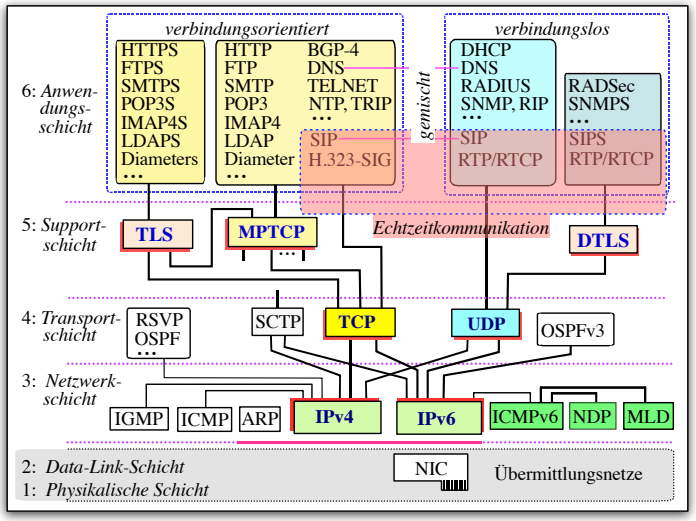


Abb. 9.1-1: Die Protokollfamilien IPv4 und IPv6 im Schichtenmodell
NIC: Network Interface Controller (Adapterkarte).
Für Erläuterung von Abkürzungen sei auf das Abkürzungsverzeichnis verwiesen.

Es wurde hier angenommen, dass die Anbindung an die untersten Schichten (d.h. die physikalische Schicht und die Data-Link-Schicht) mithilfe Netzwerk-Adapterkarte realisiert werden, z.B. in einem Rechner am Ethernet also mir einer Ethernet-Adapterkarte.

Netzwerkschicht

Innerhalb der Netzwerkschicht werden die beiden Internetprotokolle IPv4 und IPv6 mit ihren Hilfsprotokollen angesiedelt. Wie hier ersichtlich ist, wird ARP (*Address Resolution Protocol*), das von IPv4 verwendet wird, bei IPv6 durch NDP (*Neighbor Discovery Protocol*) ersetzt. IGMP (*Internet Group Management Protocol*) wird bei IPv6 durch MLD-Protokoll (*Multicast Listener Discovery*) ersetzt. ICMP (*Internet Control Message Protocol*) wird bei IPv6 zu ICMPv6 erweitert.

Transportschicht

Die Transportschicht in Abb.9.1-1 enthält die gleichen Transportprotokolle wie die Transportschicht in Abb. 1.5-1. Das Routing-Protokoll OSPF (*Open Shortest Path First*) ist der Transportschicht zuzuordnen. Die Version OSPFv3 bringt Unterstützung für IPv6 mit und wird daher auch OSPFv6 genannt.

Applikationsschicht

Die Applikationsschicht in Abb. 9.1-1 enthält im Vergleich zur Darstellung in Abb. 1.5-1 auch einige Supportprotokolle von IPv6 wie z.B. DHCPv6 (*Dynamic Host Configuration Protocol*), DNS (*Domain Name System*) und IPv6-Routing-Protokolle wie BGPv6 (*Border Gateway Protocol*) sowie RIPng (*Routing Information Protocol next generation*).

Unterstützt ein System (z.B. Rechner, Router) IPv4 und IPv6, muss ihm sowohl eine IPv4- als auch eine IPv6-Adresse zugeteilt werden.

Hauptmerkmale der Koexistenz von IPv4 und IPv6

Die Koexistenz von IPv4 und IPv6 hat unterschiedliche Aspekte, deren Hauptmerkmale die folgenden sind:

1. Kommunikation von IPv4- und IPv6-Rechnern untereinander; hierfür wird eine Protokollumsetzung (*Protocol Translation* , PT) benötigt. — IPv4 $\Leftrightarrow$ IPv6
2. IPv6-Kommunikation über ein IPv4-Transitnetz, zum Beispiel durch Kopplung von IPv6-Sites über das IPv4-Internet. IPv6 bietet durch den Protokoll-Type 41 (`Proto 41`, vgl. Tab. 2.7-1) hierfür eine Möglichkeit, was generell als *6in4* bezeichnet wird. — IPv6 $\rightleftharpoons$ IPv4 $\rightleftharpoons$ IPv6
3. IPv4-Kommunikation über IPv6-Netze, was dann Relevanz bekommt, sollte das IPv4-Internet 'abgeschaltet' werden. — IPv4 $\rightleftharpoons$ IPv6 $\rightleftharpoons$ IPv4

Im Hinblick auf den unterschiedlichen Charakter von IPv6- und IPv4-Adressen ergeben sich folgende Anforderungen:

- Öffentliche IPv4-Adressen lassen sich auf globale IPv6-Adressen statisch abbilden; wir sprechen in diesem Zusammenhang von einem *Stateful Mapping*. — Stateful Mapping
- Private IPv4-Adressen mit ständig wechselnden IPv4-Adressen, bedingt durch den NAT-Einsatz, benötigen ein dynamisches Verfahren, also ein *Stateless Mapping*. Häufig wird dies durch algorithmische Verfahren realisiert. Hierbei wird ausgenutzt, dass sich 32 Bit lange IPv4-Adressen in IPv6-Adressen zusammen mit einem 'Verwendungszweck' einbetten lassen. Hierbei sprechen wir dann auch von einem *Automatischen Mapping*. — Stateless Mapping

In den vergangenen Jahren wurden viele verschiedenartige Ansätze diskutiert (vgl. RFC 7059), vorgeschlagen und wieder verworfen. Nach längerer Zeit haben sich aber einige Konzepte herauskristallisiert, deren Zusammenhang Abb.9.1-2 illustriert. Die Koexistenz von IPv6 und IPv6 verlangt aber Lösungen folgender Probleme:

- Zuordnung von IPv4- auf IPv6-Adressen und damit eine gegenseitige logische Adressierbarkeit der Rechner untereinander.
- Einbeziehung privater IPv4-Adressen, die in der Regel per NAT/PAT auf öffentliche IPv4-Adressen abgebildet werden.
- 'Tunneling' von IPv4/IPv6-Paketen über das jeweils andere Transitnetz und unter Einbeziehung des Einflusses von NAT-Gateways.
- Unterstützung nicht nur für die 'Core'-Protokolle IPv4 und IPv6, sondern auch der Hilfsprotokolle wie ICMP und ggf. IGMP sowie MLD.

Migrationstechnologien

Für die Koexistenz von IPv4 und IPv6 kommen folgende Lösungsansätze in Betracht:

1. **Dual-Stack-Betrieb**: Einsatz von Systemkomponenten, z.B. Rechner oder Router im Netzwerk, die in der Lage sind, beide Protokolle zu nutzen.
2. **Tunneling**: In der Regel wird hier IPv6 über ein anderes, verbindungsloses Protokoll 'getunnelt'. Konkret haben es hier mit der Verkapselung (*Encapsulation*) von IPv6-

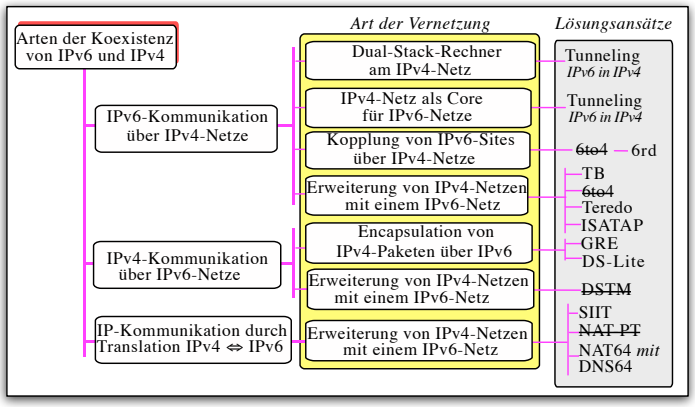


Abb. 9.1-2: Zusammenstellung der Ansätze für die Koexistenz von IPv6 und IPv4
6to4: 6to4 Transition Mechanism, DSTM: Dual-Stack Transiton Mechanism, ISATAP: Intra-Site Automatic Tunnel Addressing Protocol, NAT-PT: Network Address Translation - Protocol Translation, SIIT: Stateless IP/ICMP Translation Algorithm, GRE: Generic Routing Encapsulation, 6rd: IPv6 Rapid Deployment; TB: Tunnel-Broker, Teredo: Tunneling IPv6 over UDP through NATs; *die durchgestrichen Verfahren sind nicht mehr aktuell*

Paketen in andere, z.B. IPv4- oder auch UDP-Pakete zu tun. Das Tunneling-Verfahren kann allerdings auch für IPv4-Pakete als Payload in IPv6-Paketen realisiert werden.

GRE

In Ergänzung kann für beide Einsatzgebiete des Tunneling alternativ das *Generic Routing Encapsulation* (GRE) [RFC 2784] genutzt werden.

3. **Protokoll-Translation**: Wir nutzen die 'Gleichartigkeit' der Protokolle IPv4 und IPv6 und tauschen 'nur' den IP-Header samt Adressen aus, wobei die Hilfsprotokolle wie ICMP speziell zu behandeln sind.
4. **Middleboxen**: Unter einer Middleboxen werden spezielle Router verstanden, die beim Anwender im LAN angesiedelt sind und den Zugang zum ISP realisieren. Daher werde sie auch als *Customer Premises Equipment*(CPE) bezeichnet. Diese nehmen an diesem zentralen Punkt
   - eine *6to4-Translation* vor, was Gegenstand des *6to4 Rapid Deloyment* 6rd ist, wie in RFC 5569/5969 spezifiziert. Hierbei haben wir es mit einer *4in6-Encapsulation* zu tun und zugleich mit einem NAT, das nun nicht beim Nutzer, sondern beim ISP stattfindet, wofür auch die Bezeichnung *Carrier-Grade NAT* (CGN) verwendet wird. Alternativ kann
   - ein *4in6-Tunneling* vorgenommen werden, so wie dies *Dual-Stack Lite* (DS-Lite) [RFC 6333] vorsieht,
5. **Application-Level-Gateways (ALG)**: Für einige Protokolle wie z.B. HTTP bietet sich der Einsatz von ALGs an, um älteren Systemen Zugang zum IPv6-Netz zu verschaffen. Ein Dual-Stack HTTP-Proxy kann z.B. einem Rechner, der nur per IPv4 angeschlossen ist, auch Zugang zu IPv6-Web-Quellen bieten. ALGs implementieren daher ein *Dual-Stack-Gateway*.

Arten der Koexistenz

Beschränken wir uns für die Koexistenz von IPv4 und IPv6 auf eine Kopplung auf den Schichten 3 und 4, so kommen folgende Arten der Netzanbindung in Frage:

- *IPv6-Kommunikation über IPv4-Netze*: Es handelt sich hier um die Kopplung von Rechnern mit IPv6 über IPv4-Netze bzw. um die Erweiterung der IPv4-Netze mit IPv6-Netzen. In diesem Fall unterscheidet man zwischen den folgenden Vernetzungsarten:
  - Einsatz von Dual-Stack-Rechnern an einem IPv4-Netz: Dies ist durch das IPv6-in-IPv4-Tunneling möglich [Abb. 9.1-3b].
  - *IPv4-Netz als Core-Netzwerk bzw. als Transitnetz für IPv6-Netze*: Dies ist ebenfalls durch das IPv6-in-IPv4-Tunneling möglich [Abb. 9.3-4].
  - *Kopplung von IPv6-Sites über IPv4-Netze*: In einem IPv4-Netz können einige 'Inseln' mit nur IPv6-Systemkomponenten eingerichtet werden. Solche IPv6-Inseln werden als *IPv6-Sites* bezeichnet. Die Kopplung von IPv6-Sites über IPv4-Netze ermöglicht das Konzept *6to4*, das mittlerweile 'historisch'[1] ist [RFC 3964] und durch das Protokoll *6rd* [RFC 5569] abgelöst wurde.
  - *Erweiterung eines IPv4-Netzes mit einem IPv6-Netz*: Ein IPv4-Netz kann 'räumlich' mit einem IPv6-Netz erweitert werden. Um dies ist zu erreichen, stehen die Konzepte *Tunnel-Broker* [Abschnitt 9.3.3], *ISATAP* [Abschnitt 9.5] und *Teredo* [Abschnitt 9.6] zur Verfügung.
- *IPv4-Kommunikation über IPv6-Netze*: Es handelt sich hier um den Einsatz von Dual-Stack-Rechnern in einem IPv4-Netz bzw. um eine räumliche Erweiterung eines IPv4-Netzes mit einem IPv6-Netz. Für diese Art der Kommunikation wurde der Vorschlag 'Dual Stack IPv6 Dominant Transition Mechanism (DSTM)'[2] gemacht, der aber nicht mehr aktuell ist. Wir werden daher *DSTM* nicht weiter berücksichtigen. Statt dessen wird vom *Generic Routing Encapsulation* (GRE) RFC 2784 Gebrauch gemacht, was häufig auch als '`Proto 47`'-Encapsulation bezeichnet wird.
- *IP-Kommunikation durch Translation*: Zwischen einem IPv4-Netz und einem IPv6-Netz kann ein Router eingesetzt werden, in dem der IPv4-Header auf den IPv6-Header und umgekehrt umgesetzt werden kann. Es handelt sich daher um eine *Translation IPv4 ⇔ IPv6* [Abschnitt 9.7]. Man kann in diesem Fall von einer *IP-Kommunikation* zwischen IPv4-Rechner und IPv6-Rechner sprechen. Für die Unterstützung dieser Art der Kommunikation stehen *SIIT* [Abschnitt 9.8] und früher auch *NAT-PT* zur Verfügung. NAT-PT wurde aber aufgrund vieler Interoperabilitätsprobleme nicht mehr empfohlen [RFC 4966] und wir gehen daher nicht weiter darauf ein.
  Der Nachfolger von NAT-PT ist *NAT64* [RFC 6144, 6146], das mit *DNS64* [RFC 6147] sinnvoll ins DNS integriert ist.

Wir wollen nun zunächst die prinzipiellen Aspekte der Technologien zur Sicherstellung der Koexistenz beider Netzprotokolle vorstellen, bevor wir diese in den nächsten Abschnitten im Einzelnen beleuchten.

9.1.1 IPv6-Kommunikation über IPv4-Netze

Solange noch keine flächendeckende IPv6-Infrastruktur von den ISPs bereit gestellt wird, kann die bestehende IPv4-Netzinfrastruktur zur Unterstützung der IPv6-Kommunikation verwendet werden. In der ersten Phase der Migration zum Einsatz von IPv6 fungieren die bestehenden IPv4-Netze als *Transitnetze*. Abb. 9.1-3 zeigt eine Zusammenstellung

[1] `http://tools.ietf.org/html/draft-ietf-v6ops-6to4-to-historic-04`
[2] `http://tools.ietf.org/search/draft-bound-dstm-exp-04`

von Lösungen, bei denen IPv4-Netze als Transitnetze für die Unterstützung der IPv6-Kommunikation dienen.

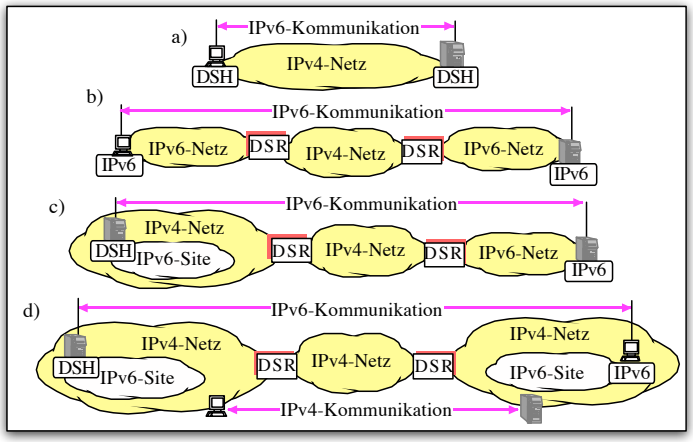


Abb. 9.1-3: IPv4-Netze als Transitnetze für die Unterstützung der IPv6-Kommunikation:
a) Dual-Stack-Rechner am IPv4-Netz, b) IPv4-Netz als Transitnetz für IPv6-Netze,
c) IPv4-Netz als Zubringer zum IPv6-Netz, d) Vernetzung von IPv6-Sites
DSH: Dual-Stack-Host, DSR: Dual-Stack-Router

IPv4-Netz als Transitnetz

Dient ein IPv4-Netz als Transitnetz bei der IPv6-Kommunikation, so kann es sich um folgende Vernetzungsarten handeln:

a. Einsatz von Dual-Stack-Rechnern (-Hosts) am IPv4-Netz [Abb. 9.1-3a]
Für die IPv6-Kommunikation zwischen zwei Dual-Stack-Rechnern am IPv4-Netz wird ein IPv6-in-IPv4-Tunnel aufgebaut [Abb. 9.1-3b], worauf Abschnitt 9.2.2 eingeht.

b. Ein IPv4-Netz fungiert als Transitnetz für IPv6-Netze [Abb. 9.1-3b]
Um die IPv6-Kommunikation bei dieser Vernetzungsart zu ermöglichen, wird ebenfalls das IPv6-in-IPv4-Tunneling eingesetzt. Abschnitt 9.3 präsentiert dies näher.

IPv6-Insel als IPv6-Site

c. Ein IPv4-Netz dient als Zubringer zum IPv6-Netz [Abb. 9.1-3c]
In einem IPv4-Netz kann eine 'IPv6-Insel' eingerichtet werden. Sie wird auch IPv6-Site genannt. Ein IPv4-Netz kann dann für Rechner aus der IPv6-Site als Zubringer zu einem IPv6-Netz dienen. Um dies zu ermöglichen, steht das in Abschnitt 9.4 dargestellte Konzept *6rd* zur Verfügung.

d. Vernetzung von IPv6-Sites über IPv4-Netze [Abb. 9.1-3d]
Diese Vernetzungsart ist auch mittels von 6rd möglich. Dies wird in Abschnitt 9.4 detailliert dargestellt.

Dual-Stack-Betrieb

Wird ein IPv4-Netz um ein IPv6-Netz bzw. um eine IPv6-Site erweitert [Abb. 9.1-4], so handelt es sich um die IPv6-Kommunikation zwischen einem Dual-Stack-Rechner am IPv4-Netz, und es kommen die folgenden Möglichkeiten in Betracht:

- Ein IPv6-Rechner in einem IPv6-Netz [Abb. 9.1-4a]:
Diese IPv6-Kommunikation wird durch das IPv6-in-IPv4-Tunneling realisiert [Abschnitt 9.3].
- Ein IPv6-Rechner in einer IPv6-Site [Abb. 9.1-4b]:
Diese IPv6-Kommunikation ermöglicht das Konzept 6to4 [Abschnitt 9.4].

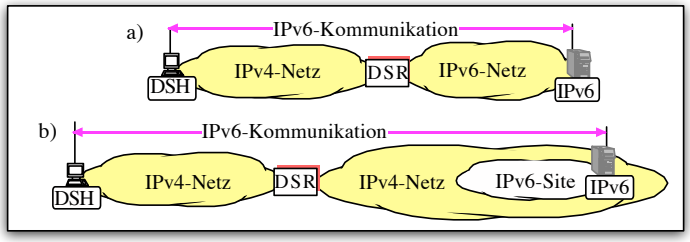


Abb. 9.1-4: IPv6-Kommunikation zwischen Dual-Stack-Rechner am IPv4-Netz und: a) IPv6-Rechner in einem IPv6-Netz, b) IPv6-Rechner in einer IPv6-Site
Abkürzungen wie in Abb. 9.1-3

IPv6 in IPv4-Encapsulation

Bei der Nutzung von IPv4-Netzen für den Transit von IPv6-Paketen ergibt sich die Anforderung, den IPv6-Unicast- und Multicast-Datenverkehr über IPv4-Netze (transparent) transportieren zu können. Hierfür sind im wesentlichen drei Mechanismen in Gebrauch:

- *Protocol 41 Encapsulation*: Hierbei wird dem zu übertragenden IPv6-Paket unmittelbar ein IPv4-Header voranstellt und die Nutzlast mit dem Protocol-Typ 41 bezeichnet. Hierdurch ist die Nutzlast der IPv4-Pakete als 'IPv6' gekennzeichnet, und diese können von einem Dual-Stack-Host am Ziel entpackt und weiterverarbeitet werden. Dies ist die grundlegende Idee der Lösungen *6to4* und *6in4*. Auch der alte Standard *6over4* [RFC 2529] nutzt diese Art der Verkapselung. — Proto 41
- *Protocol 47 Encapsulation*: Es wird der *Generic Encapsulation Routing* (GRE)-Mechanismus genutzt, sodass das IPv6-Paket zunächst mit einem GRE-Header ergänzt und anschließend das Gesamtpaket durch den notwendigen IPv4-Header vervollständigt wird. Den Einsatz von GRE stellen wir in Abschnitt 12.4 vor. — Proto 47
- *UDP Encapsulation*: Nach diesem Konzept wird ein IPv6-Datenpaket zunächst in ein UDP-Paket eingekapselt. Hierbei kann eine beliebige Portnummer gewählt werden, die allerdings beim NAT-Einsatz eventuell geändert wird, wie bei Teredo [Abschnitt 9.6] üblich. Abschließend ist das UDP-Paket in einem IPv4-Paket verkapselt. — UDP
- *AYIYA*: Das *Anything-in-Anything*-Konzept geht noch einen Schritt weiter und steckt das IPv6-Paket nicht nur in ein UDP-Datagramm, sondern stellt diesem einen AYIYA-Header voraus, der Informationen über die Netzwerkadressierung trägt. Das IPv4-Paket trägt nun als Nutzlast ein UDP-Datagramm mit hinzugefügten AYIYA-Header und abschließend das IPv6-Datenpaket [Abb. 9.3-6]. — AYIYA

9.1.2 IPv4-Kommunikation über IPv6-Netze

Es sollte möglich sein, dass Rechner in IPv6-Netzen auf die Ressourcen im IPv4-Internet zugreifen können. Dafür muss in Rechnern im IPv6-Netz zusätzlich IPv4 installiert werden. Daher ist der Betrieb von Dual-Stack-Rechnern im IPv6-Netz von großer Bedeutung – genauso wie die Möglichkeit, dass sie die IPv4-Kommunikation zu Rechnern — Bedeutung von IPv4 über IPv6

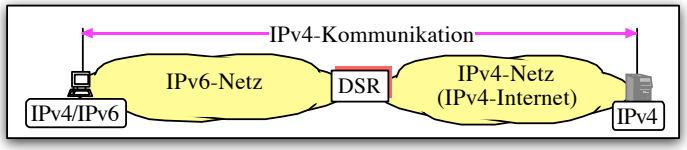


Abb. 9.1-5: IPv4-Kommunikation zwischen Dual-Stack-Rechnern im IPv6-Netz und IPv4-Rechnern am IPv4-Netz
DSR: Dual-Stack-Router

in IPv4-Netzen initiieren können. Wie Abb. 9.1-5 zeigt, handelt es sich hier um die IPv4-Kommunikation über ein IPv6-Netz, also um eine Art von *IPv4 over IPv6*.

Um IPv4 über IPv6 zu unterstützen, wurde das Konzept DSTM (*Dual Stack Transition Mechanism*) entwickelt. Dieses Verfahren kann bestenfalls dazu genutzt werden, alte Rechner mit lediglich einem IPv4-Netzwerkstack mit einem Dual-Stack-Rechner kommunizieren zu lassen. Diese Voraussetzung ist mittlerweile entfallen und ließe sich zudem durch eine *Protokoll-Translation* genau so gut realisieren.

9.1.3 IP-Kommunikation durch Translation IPv4 ⇔ IPv6

Auch die Kommunikation zwischen IPv4-Rechnern im IPv4-Netz und IPv6-Rechnern im IPv6-Netz ist möglich. Hierfür ist eine *Translation IPv4 ⇔ IPv6* in einem Router zwischen diesen beiden Netzen notwendig. Abb. 9.1-6 illustriert diesen Ansatz.

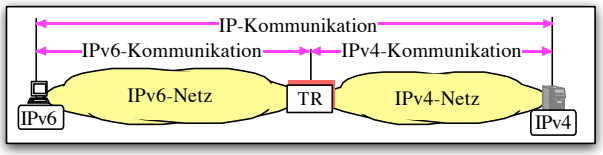


Abb. 9.1-6: IP-Kommunikation durch die Translation IPv4 ⇔ IPv6 im Router
TR: Translation Router

Die Translation *IPv4 ⇔ IPv6* ist Bestandteil des Konzepts *SIIT* (*Stateless IP/ICMP Translation Algorithm*) und dem aktuellen Ansatz *NAT64* mit *DNS64* (vgl. Abschnitt 9.8).

9.2 Dual-Stack-Verfahren

Dual-Stack-Rechner und -Router

Netzwerkkomponenten mit den beiden Protokollstacks IPv4 und IPv6 bezeichnet man als *Dual-Stack-Rechner* (*Dual-Stack-Host*) und *Dual-Stack-Router*. Beim Einsatz von Dual-Stack-Systemen wollen wir zwischen der direkten IPv4/IPv6-Kopplung auf Rechnerbasis und der Anbindung über einen Dual-Stack-Router unterscheiden, ein Verfahren das von *Dual-Stack Lite* (DS-Lite) genutzt wird.

9.2.1 Dual-Stack-Rechner in einem LAN-Segment

Den Einsatz von IPv4 und IPv6 in einem physikalischen LAN-Segment illustriert Abb. 9.1-2. Zwischen IPv4-Rechnern findet die Kommunikation nach IPv4 und zwischen IPv6-Rechnern nach IPv6 statt. Hier wird das ganze Netzwerk in zwei logische 'Netzwerkteile' aufgeteilt, sodass IPv4-Rechner einen IPv4-Netzwerkteil und entsprechend IPv6-Rechner einen IPv6-Netzwerkteil bilden.

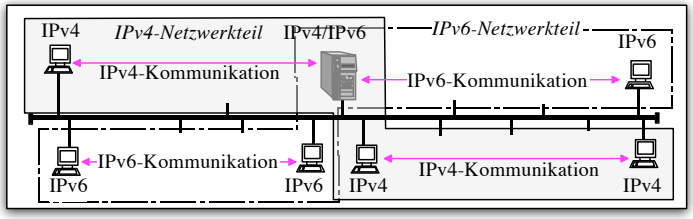


Abb. 9.2-1: Paralleler Einsatz von IPv4 und IPv6 in einem LAN-Segment

Ein Dual-Stack-Rechner mit IPv4 und IPv6 kann sowohl nach IPv4 mit IPv4-Rechnern als auch nach IPv6 mit IPv6-Rechnern kommunizieren. Jeder Dual-Stack-Rechner ist daher gleichzeitig aktiver Knoten sowohl im IPv4- als auch im IPv6-Netz. In der in Abb. 9.1-2 gezeigten Situation können den IPv6-Rechnern beliebige Unicast-IPv6-Adressen zugeteilt werden.

9.2.2 Betrieb von Dual-Stack-Rechnern in IPv4-Netzen

Dual-Stack-Rechner = IPv4/IPv6-Rechner

Sollen neue IPv6-Applikationen in einem IPv4-Netz eingesetzt werden, müssen einige IPv4-Rechner um IPv6 erweitert werden. Sie werden damit zu Dual-Stack-Rechnern umgerüstet, die man auch als *IPv4/IPv6-Rechner* bezeichnet. Ein IPv4-Netz kann somit als *Transitnetz* für die IPv6-Kommunikation zwischen den derart erweiterten Rechnern eingesetzt werden. Abb. 9.2-2a zeigt ein Beispiel, in dem zwei Ethernet-Segmente über einen Router miteinander verbunden sind. Da der Router nur IPv4 unterstützt, handelt es sich hierbei um ein 'reines' IPv4-Netz. Werden an diesem Netz auch die Dual-Stack-Rechner angeschlossen, stellt sich die Frage: Wie erfolgt die IPv6-Kommunikation zwischen ihnen? Die Antwort gibt Abb. 9.2-2b.

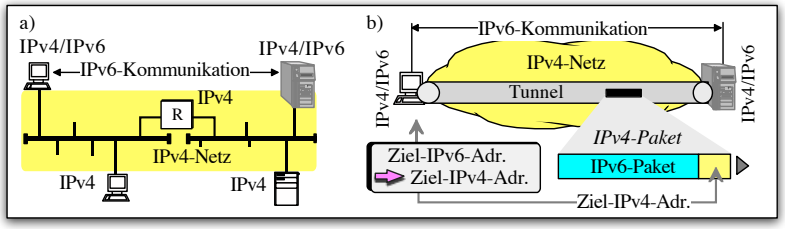


Abb. 9.2-2: IPv6-Kommunikation zwischen Dual-Stack-Rechnern am IPv4-Netz:
a) physikalische Konfiguration, b) Prinzip der IPv6-Kommunikation
R: Router

Logischer IPv6-in-IPv4-Tunnel

Bei der IPv6-Kommunikation zwischen IPv4/IPv6-Rechnern über ein IPv4-Netz werden die IPv6-Pakete in IPv4-Pakete eingebettet und als Nutzlast transportiert. Auf diese Art

und Weise entsteht ein logischer *IPv6-in-IPv4-Tunnel* über das IPv4-Netz als Transitnetz zwischen den beteiligten IPv4/IPv6-Rechnern. Beginn und Ende des Tunnels über ein IPv4-Netz bestimmen die IPv4-Adressen. Da Datenquelle und -senke bei der IPv6-Kommunikation durch die IPv6-Adressen festgelegt werden, muss der Quellrechner eine Adressermittlungstabelle besitzen, die folgende Zuordnung beschreibt:

IPv6-Zieladresse ⇒ IPv4-Zieladresse

IPv4-compatible IPv6-Adressen

Ursprünglich war vorgesehen, für ein *automatisches Tunneling* spezielle, *IPv4-kompatible IPv6-Adressen* zu nutzen Abb. 7.9-7a. Diese besitzen das allgemeine Format `::IPv4-Adresse/96` (mit den letzten 32 Bit als IPv4-Adresse). Hierbei verlangt entsprechend RFC 4291, dass die zugeordnete IPv4-Adresse eine öffentliche, d.h. keine private IPv4-Adresse ist. Dies ermöglicht zwar das Routing der Pakete im IPv4-Netzwerk [Abb. 9.2-2a], da aber IPv6-Pakete mit IPv4-kompatiblen IPv6-Adressen in IPv6-Netzwerken weiter nicht geroutet werden können, sind diese Adressen gemäß RFC 4291 *deprecated* und sollen nicht mehr eingesetzt werden.

9.2.3 Dual-Stack Lite

Dual-Stack Lite (kurz *DS-Lite*) [RFC 6333] ist keine IPv6-Migration im eigentlichen Sinne, sondern bietet viel mehr für den *Internet Service Provider* (ISP) die Möglichkeit, seine IPv4-Kundennetze über ein IPv6-Transitnetz an das IPv4-Internet anzubinden und die notwendige Umsetzung privater IPv4-Adressen auf öffentliche an einer zentralen Stelle, dem *Address Family Transition Router* (AFTR) vorzunehmen, der zugleich NAT-Funktionen bereit stellt.

Der Name DS-Lite ist dem Umstand zu verdanken, dass im Kundennetz nun lediglich eine Netzwerkkomponenten, das *Customer Premises Equipment* (CPE) – also der Heim-Router beim Kunden –, die geforderte Dual-Stack-Funktion aufweist, die auch nur in Richtung zum ISP genutzt wird.

DS-Lite Komponenten

Bei Dual-Stack Lite kommen folgende Funktionsgruppen zum Einsatz:

- Das CPE, das neben den Routing-Diensten (als Default-Gateway im Kundennetz)
  - auf der LAN-, d.h. Client-Seite einen DHCPv4-Server für das interne Kundennetz bereit stellt, und hier die üblichen privaten IPv4-Adressen vergibt. Zugleich bietet das CPE DNS-Resolver- bzw. Proxydienste an (siehe Abschnitt 4.1.4).
  - Auf der WAN-Seite (dem IPv6-*B4-Interface* siehe Abb. 9.2-3) erfolgt die notwendige *4in6-Encapsulation* in das Netz des ISP. Die notwendigen Konfigurationsdaten erhält der CPE per DHCPv6 vom Provider, d.h. er besitzt einen DHCPv6-Client. Dies wird in RFC 6333 auch als *Provisionierung* bezeichnet. Zur Vereinfachung der IPv4/IPv6-Umsetzung lautet die IPv4-Adresse *jedes* B4-Interfaces immer `192.0.0.2`.
- Der ISP betreibt ein oder mehrere Gateways ins IPv4-Netz, deren Aufgabe es ist,
  - die IPv4- aus den IPv6-Paketen zu entnehmen (*Decapsulation*) bzw. die aus dem IPv4-Internet kommenden Pakete in IPv6-Datagramme zu verpacken (*Encapsulation*) und über den IPv6-Tunnel zum Kunden weiterzuleiten sowie
  - an dieser zentralen Stelle die NAT-Funktion für die IPv4-Adressen in beide Richtungen vorzunehmen (was auch als *NAT44* bezeichnet wird). Für das NAT44-Gateway

wird die reservierte Adresse `192.0.0.1` genutzt, wobei aber auch die übrigen aus dem Netz `192.0.0.0/29` verfügbaren Adressen herangezogen werden können.

Beispiel: Wir betrachten die Situation, die sich aus Abb. 9.2-3 ergibt. Die Gateway-Komponenten CPE und AFTR/NAT44 wurden bereits vorgestellt. Wir sehen, dass die Adressen `192.0.0.1` und `192.0.0.2` praktisch nur symbolische Bedeutung im IPv6-Transitnetz haben: Sowohl das Routing im ISP-Netz als auch die Zuordnung einer öffentlichen IPv4-Adresse kann auf Grundlage der IPv6-Adresse des CPE erfolgen. Als *IPv4-Quelladresse* wird immer `192.0.0.2` genutzt. Wie bisher sind alle IPv4-Knoten im Kunden-LAN (mit unterschiedlichen privaten Adressen aus dem Netz `192.168.1.0/24`) immer auf eine öffentliche IPv4-Adresse abgebildet (hier: `9.254.253.252`) – mit allen bekannten Nachteilen.

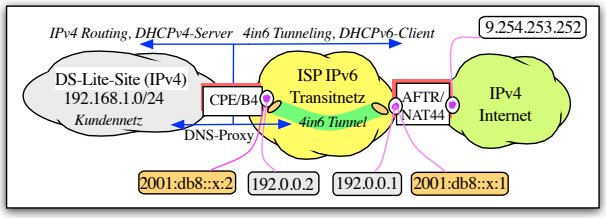


Abb. 9.2-3: Grundlegende Idee des Einsatzes von DS-Lite
AFTR: Address Family Transition Router, B4: IPv4-Bridge,
CPE: Customer Premises Equipment

Der AFTR realisiert also beim ISP die Zuordnung

AFTR-Adressen-Zuordnung

öffentliche IPv4-Adresse ⇔ (provisionierte) IPv6-Adresse

mittels einer internen Tabelle, wobei ansonsten (nach Entnahme des IPv4-Pakets aus dem IPv6-Paket im ISP-Netzwerk) das NAT-Verfahren gemäß RFC 1918 vorgenommen wird, was wir bereits in Abschnitt 5.3 beschrieben haben. Die Auslagerung der NAT-Funktion weg vom Kundenrouter und hin zu einer zentralen Instanz erlaubt zunächst eine ökonomischere Ausnutzung der verfügbaren öffentlichen IPv4-Adressen, was aktuell auch als *Carrier-Grade NAT* (CGN) diskutiert wird (vgl. Abschnitt 5.3.7).

9.3 Tunneling-Protokolle: IPv6 über X

IPv6-in-IPv4-Tunneling wird in der Regel verwendet, um die IPv6-Kommunikation über ein IPv4-Netz in folgenden Situationen zu realisieren:

- bei einer Erweiterung eines IPv4-Netzes um ein IPv6-Netz bzw.
- bei einer Kopplung der IPv6-Netze über ein IPv4-Netz.

Die Einsatzmöglichkeiten des IPv6-in-IPv4-Tunnelings werden nun im Detail dargestellt.

9.3.1 Erweiterung eines IPv4-Netzes um ein IPv6-Netz

Bei der Erweiterung eines IPv4-Netzes um ein IPv6-Netz sind die Fälle zu unterscheiden:

- *Im IPv4-Netz werden ausschließlich IPv4-Rechner installiert*: In diesem Fall [Abb. 9.1-6] kann die Kommunikation zwischen IPv4-Rechnern im IPv4-Netz und IPv6-Rechnern im IPv6-Netz auch erreicht werden. Hierfür muss die Translation *IPv4 ⇔ IPv6* vom Router vorgenommen werden. Darauf geht Abschnitt 9.7 näher ein.
- *Im IPv4-Netz sind einige Dual-Stack-Rechner vorhanden*: In diesem Fall [Abb. 9.1-4a] kann die IPv6-Kommunikation zwischen Dual-Stack-Rechnern im IPv4-Netz und IPv6-Rechnern im IPv6-Netz stattfinden.

Einsatz von IPv4/IPv6-Routern

Die Übermittlung der IPv6-Pakete vom Dual-Stack-Rechner (IPv4/IPv6-Rechner) im IPv4-Netz zum IPv6-Rechner im IPv6-Netz illustriert Abb. 9.3-1a. Hier werden die IPv6-Pakete für die Übermittlung über das IPv4-Netz in IPv4-Pakete eingebettet. Auf diese Weise entsteht ein IPv6-in-IPv4-Tunnel zwischen IPv4/IPv6-Rechner im IPv4-Netz und -Router. Es können mehrere Router zwischen IPv4-Netz und IPv6-Netz vorhanden sein, sodass der Quellrechner über eine Adresstabelle mit den Zuordnungen *IPv6-Zieladresse ⇒ IPv4-Routeradresse* verfügen muss. Existiert nur ein Router zwischen den beiden Netzen, reduziert sich diese Tabelle zu einer Zuordnung, in der nur eine IPv4-Routeradresse allen IPv6-Zieladressen zugeordnet wird.

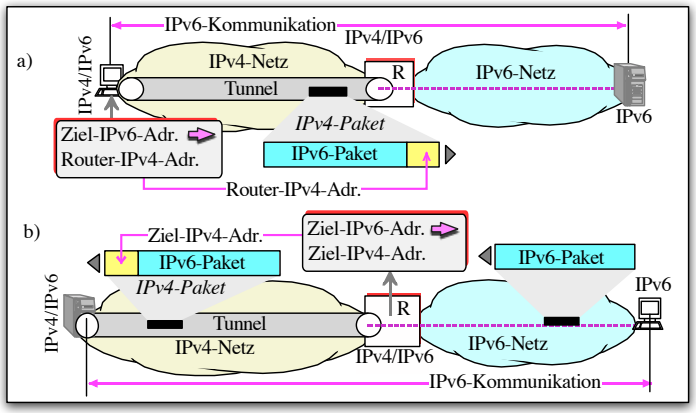


Abb. 9.3-1: IPv6-Kommunikation bei Erweiterung eines IPv4-Netzes mit einem IPv6-Netz; Tunnel führt: a) zu einem Router, b) zu einem Dual-Stack-Rechner im IPv4-Netz

IPv6-in-IPv4-Tunnel

Abb. 9.3-1b zeigt die umgekehrte Situation, in der die Datenquelle ein IPv6-Rechner im IPv6-Netz ist. Hier werden die IPv6-Pakete zuerst an den Router übermittelt. Danach werden sie im Router in IPv4-Pakete eingebettet und über das IPv4-Netz transportiert. Diesmal wird der IPv6-in-IPv4-Tunnel vom Router initiiert, sodass der Router über die Adressermittlungstabelle mit den Zuordnungen *IPv6-Zieladresse ⇒ IPv4-Zieladresse* verfügen muss.

Im Allgemeinen ist bei IPv6-in-IPv4-Tunneling zu differenzieren zwischen

- einem konfigurierbaren Tunneling und
- einem automatischen Tunneling.

Konfigurierbares Tunneling

Wird ein IPv6-in-IPv4-Tunnel zu einem Router aufgebaut, ist die IPv6-Adresse vom Tunnelende anders als die IPv6-Zieladresse des über den Tunnel übertragenen Pakets. Hier ist die IPv6-Adresse vom Tunnelende aufgrund der Information zu bestimmen, die nur während der Konfiguration angegeben werden kann. Kommen mehrere Router in

Frage, muss der richtige Router bei der Konfiguration festgelegt werden. Wird der Tunnel zu einem Router aufgebaut, spricht man von *konfigurierbarem Tunneling* [Abb. 9.3-2]. Diese Art der Kopplung wird über einen *Tunnel-Broker* realisiert (vgl. Abschnitt 9.3.3).

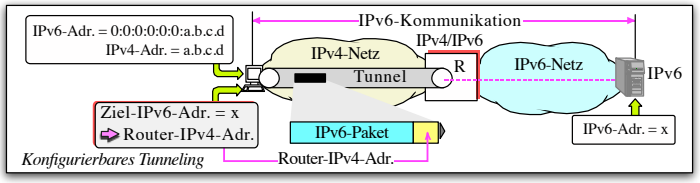


Abb. 9.3-2: Konfigurierbares IPv6-in-IPv4-Tunneling

Automatisches Tunneling

Führt ein IPv6-in-IPv4-Tunnel zu einem Rechner und ist die IPv6-Zieladresse des über den Tunnel übertragenen IPv6-Pakets IPv4-kompatibel, lässt sich das Tunnelende aus der IPv4-kompatiblen IPv6-Adresse ableiten [Abb. 7.9-7a]. Man spricht in diesem Fall von *automatischem Tunneling*, was Abb. 9.3-3 veranschaulicht. Die erwähnte Adressermittlungstabelle ist hierbei nicht nötig. Wie wir bereits in Abschnitt 9.2.2 erwähnt haben, können IPv4-kompatible IPv6-Adressen nicht für öffentliche Netzstrukturen genutzt werden, da sie nicht routbar sind, und deren Einsatz ist daher *deprecated*.

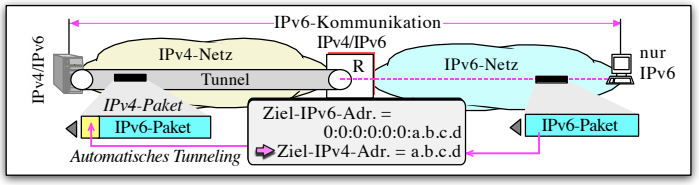


Abb. 9.3-3: Automatisches IPv6-in-IPv4-Tunneling

9.3.2 Kopplung der IPv6-Netze über ein IPv4-Netz

6in4-Tunnel

Ein IPv4-Netz kann als Transitnetz für die Kopplung der IPv6-Netze eingesetzt werden. Abb. 9.3-4 zeigt das Prinzip der IPv6-Kommunikation über ein IPv4-Transitnetz. Hier werden die IPv6-Pakete in IPv4-Pakete eingebettet und als Nutzlast zwischen den beteiligten Routern transportiert. Hierdurch entsteht ein IPv6-in-IPv4-Tunnel (*6in4-Tunnel*) zwischen diesen Routern.

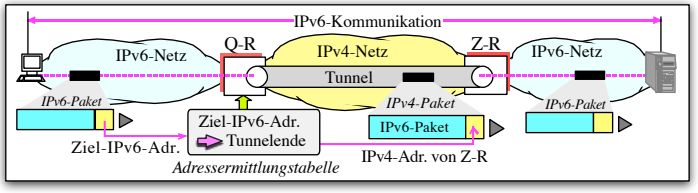


Abb. 9.3-4: IPv6-Kommunikation über ein IPv4-Transitnetz
Q-R: Quellrouter, Z-R: Zielrouter

IPv4-Transitnetz

Beginn und Ende dieses Tunnels bestimmen die IPv4-Adressen der Quell- und Zielrouter. Da Datenquelle und -senke bei der IPv6-Kommunikation durch die IPv6-Adressen

Stichwortverzeichnis

N