anatol **BADACH**
erwin **HOFFMANN**

Technik der IP-NETZE

3. Auflage

INTERNET-KOMMUNIKATION IN THEORIE UND EINSATZ

HANSER

Badach/Hoffmann

Technik der IP-Netze

Anatol Badach
Erwin Hoffmann

Technik der IP-Netze

Internet-Kommunikation in Theorie und Einsatz

3., überarbeitete und erweiterte Auflage

HANSER

Bibliografische Information der Deutschen Nationalbibliothek:

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Lektorat: Brigitte Bauer-Schiewek
Herstellung: Irene Weilhart
Copy editing: Jürgen Dubau, Freiburg
Layout: Erwin Hoffmann mit LaTeX
Umschlagdesign: Marc Müller-Bremer, www.rebranding.de, München
Umschlagrealisation: Stephan Rönigk
Druck und Bindung: Kösel, Krugzell
Ausstattung patentrechtlich geschützt. Kösel FD 351, Patent-Nr. 0748702
Printed in Germany

Print-ISBN: 978-3-446-43976-4
E-Book-ISBN: 978-3-446-43986-3

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

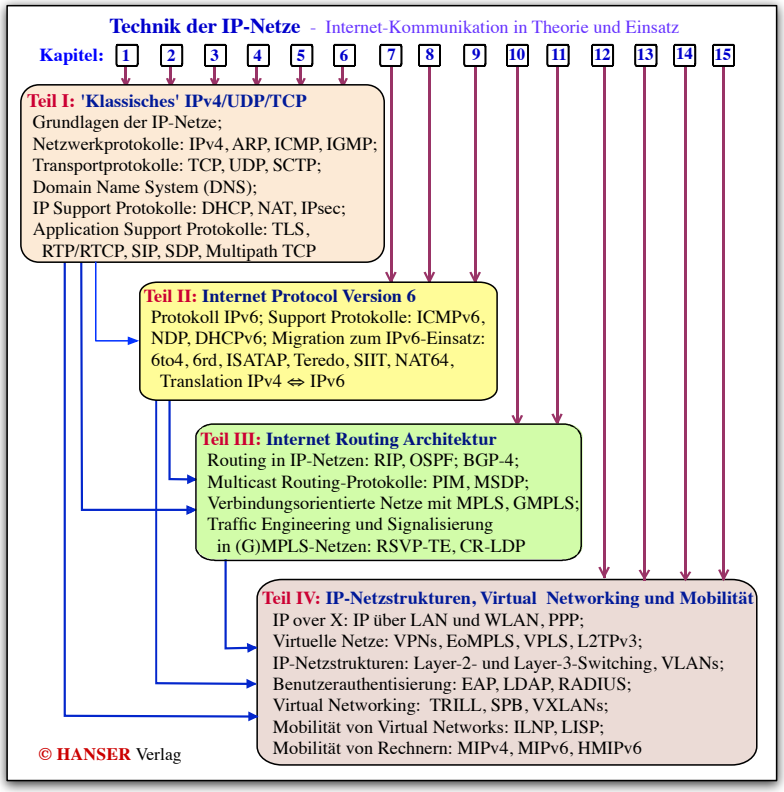
Dieses Buch gibt eine fundierte Darstellung zentraler Komponenten der TCP/IP-Protokollfamilie, wie z.B. IP, TCP, UDP, DNS und DHCP, sowie von Routing sowohl beim klassischen IP, IPv4 genannt, als auch beim neuen IPv6. Das Buch erläutert die Strategien für die Migration zum Einsatz von IPv6, präsentiert 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?

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 12

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 13

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 14

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? Kapitel 15

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

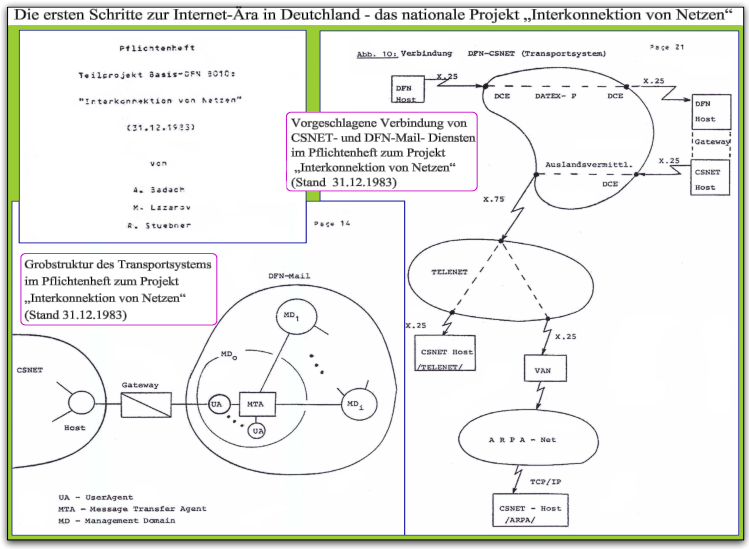
Erste Schritte zur Internet-Ära in Deutschland

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.

Teil I

Klassisches IPv4/UDP/TCP

Wilkomen in CSNET!

Michael,

This is your official welcome to CSNET. We are glad to have you aboard.

From: Laura Breeden
breeden%csnet-sh.arpa@csnet-relay.csnet
To: rotert%germany@csnet-relay.csnet
Via: csnet-relay; 3 Aug 84 10:44-MET

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 erläutertet 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.

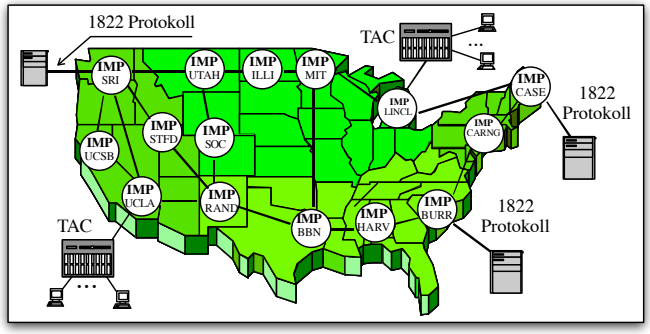


Abb. 1.1-1: Allgemeiner Aufbau von ARPANET – IMP dienen als Knoten
TAC: Terminal Access Controller, IMP: Internet Message Processor

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.

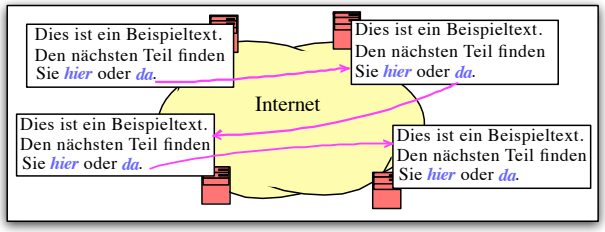


Abb. 1.1-2: Verknüpfung von Dokumenten auf unterschiedlichen Servern mittels Hyperlinks

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.

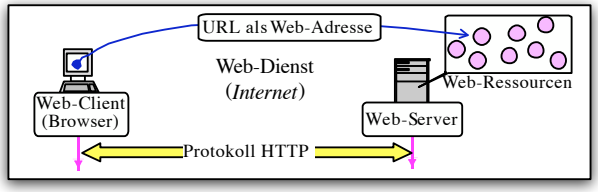


Abb. 1.1-3: Hauptkomponenten des Webdienstes – URL dient als Adresse

Die Grundkomponenten des Webdienstes sind:

Webbrowser

- 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.

URLs als Web-Adressen

- 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.

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.

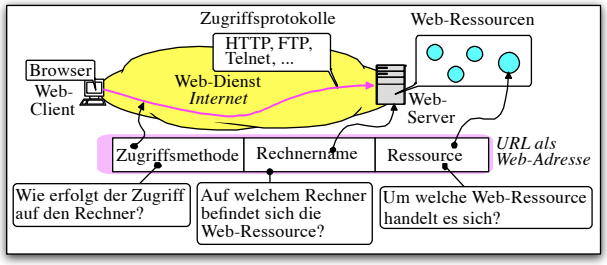


Abb. 1.1-4: Prinzip der Adressierung beim Web-Dienst

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.

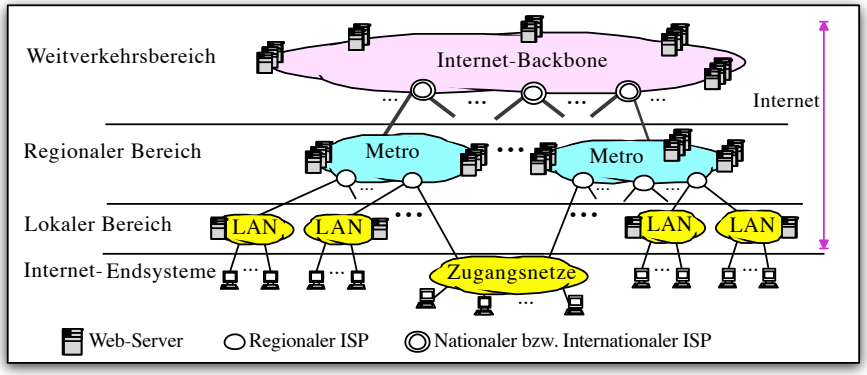


Abb. 1.1-5: Allgemeine Internet-Strukturierung – Aufbau als baumartige Struktur
ISP: Internet Service Provider; LAN: Local Area Network; Metro: Metro(politan)-Netz

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.

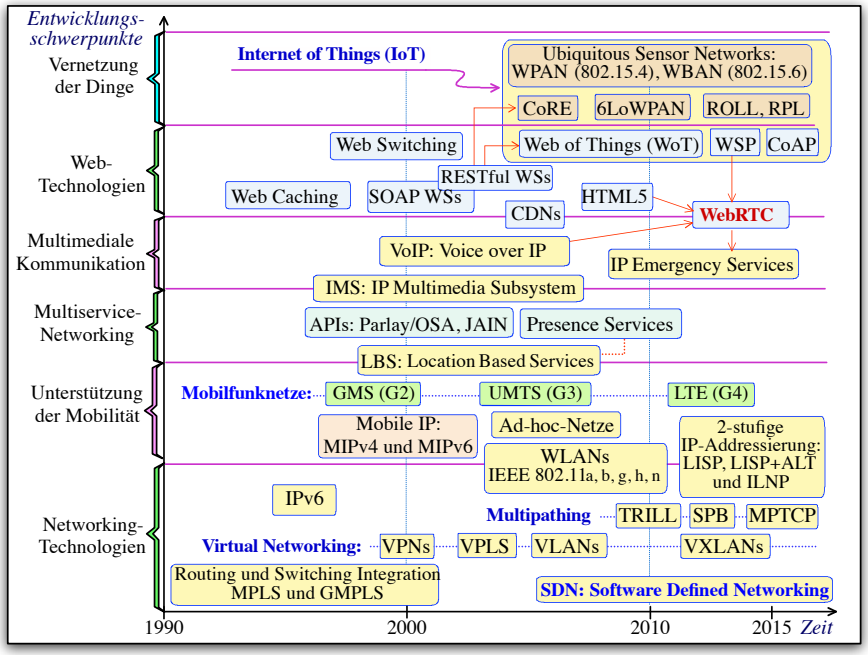


Abb. 1.1-6: Internet und IP-Netze; Meilensteile der bisherigen Entwicklung und Trends
CoRE: Constrained RESTful Environment, G: Generation, ROLL: Routing over Low power and Lossy neworks, RPL: Routing Protocol for Low power and Lossy networks, WPAN: Wireless Personal Area Network, WBAN: Wireless Body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

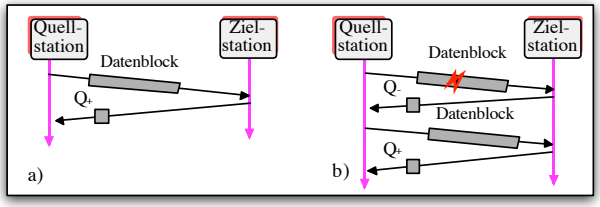


Abb. 1.2-1: Übermittlung eines Datenblocks: a) fehlerlos, b) fehlerhaft
Q+: positive Quittung, Q–: negative Quittung

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.

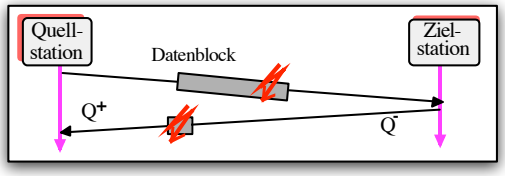


Abb. 1.2-2: Worst case einer Datenmittlung: Daten und Quittung werden verfälscht

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

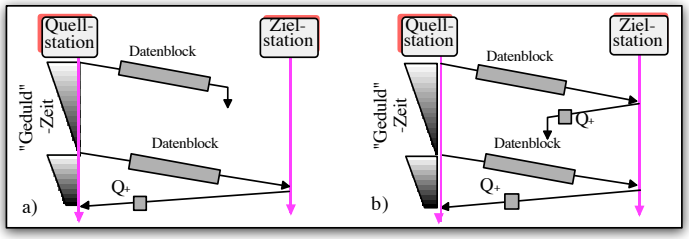


Abb. 1.2-3: Fehlerhafte Übermittlung: a) Datenblockverlust, b) Quittungsverlust

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,

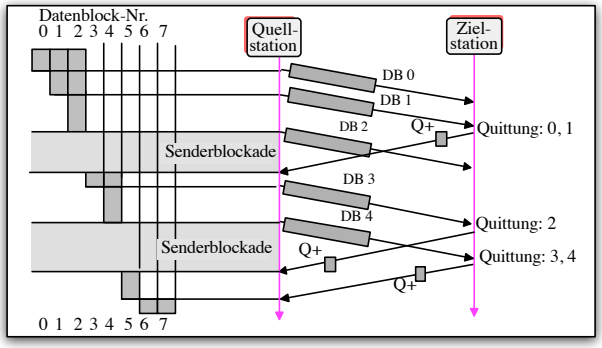


Abb. 1.2-4: Veranschaulichung der Flusskontrolle über den Fenstermechanismus

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

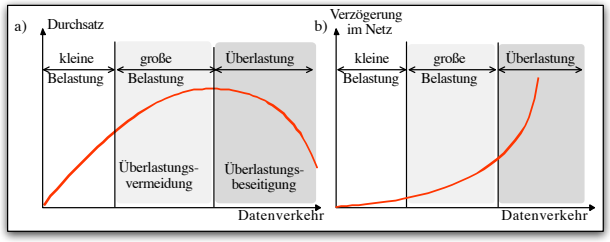


Abb. 1.2-5: Auswirkungen der Netzüberlastung: a) auf den Durchsatz, b) auf die Datenverweilzeit im Netz

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.

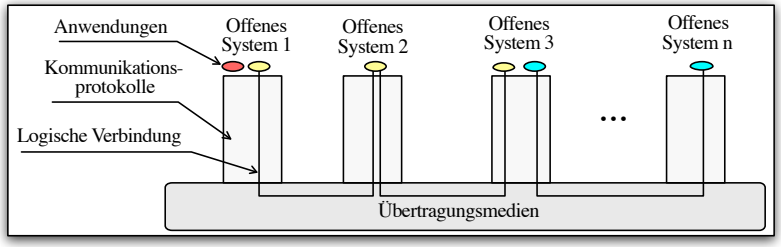


Abb. 1.3-1: Idee von OSI: Jedes System soll mit jedem anderen kommunizieren können

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*).

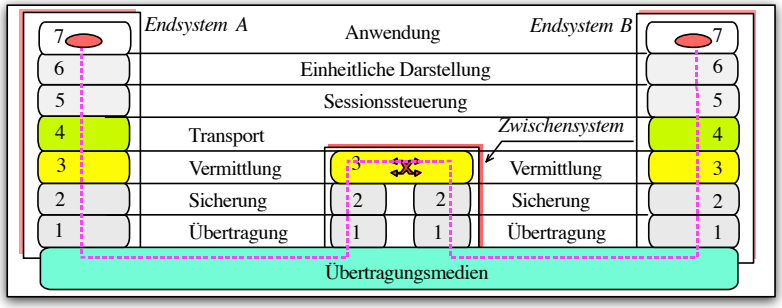


Abb. 1.3-2: OSI-Referenzmodell: Zerlegung der Kommunikationsaufgabe in 7 Schichten

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

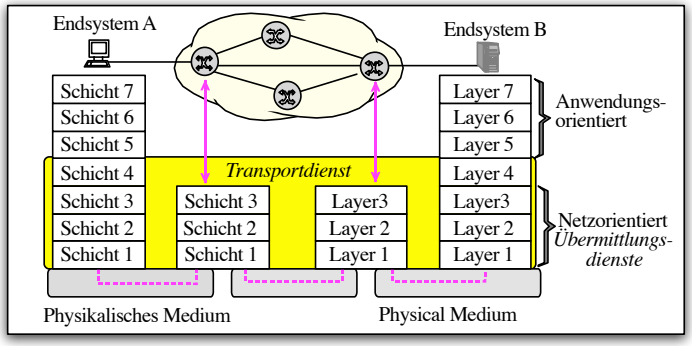


Abb. 1.3-3: Klassen von Aufgaben im OSI-Referenzmodell

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.

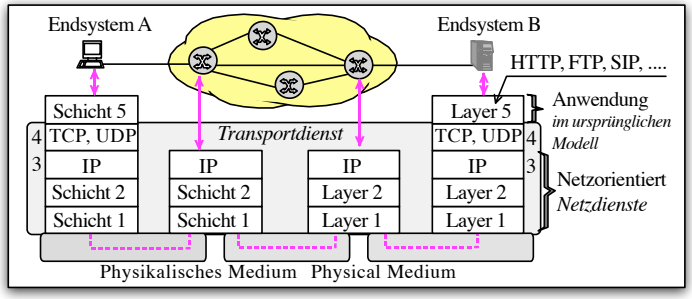


Abb. 1.3-4: Ursprüngliches Schichtenmodell der Protokollfamilie TCP/IP

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.

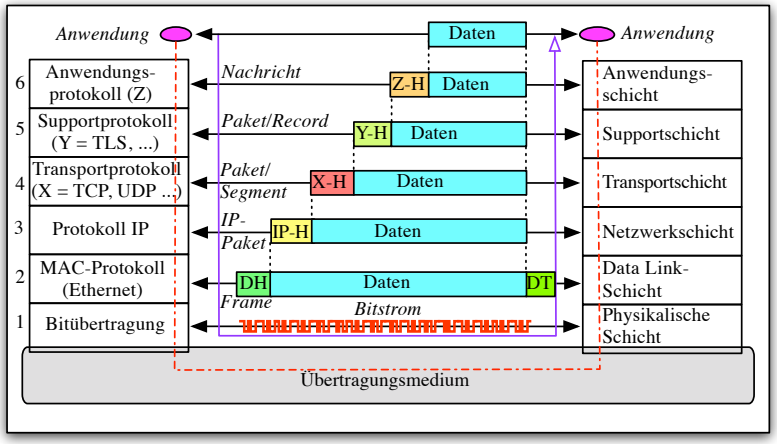


Abb. 1.3-5: Erweitertes Schichtenmodell der Protokollfamilie TCP/IP – Strukturen von zwischen den kommunizierenden Instanzen übermittelten Daten
DH: Data-Link Header, DT: Data-Link Trailer

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.

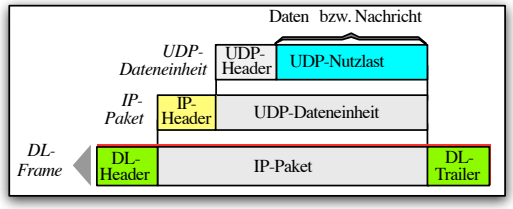


Abb. 1.4-1: Kapselung der Nutzlast beim UDP-Einsatz

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.

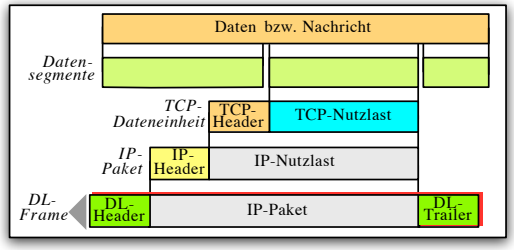


Abb. 1.4-2: Verkapselung der Nutzlast beim TCP-Einsatz

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*.

Verbindungsorientiert

- 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.

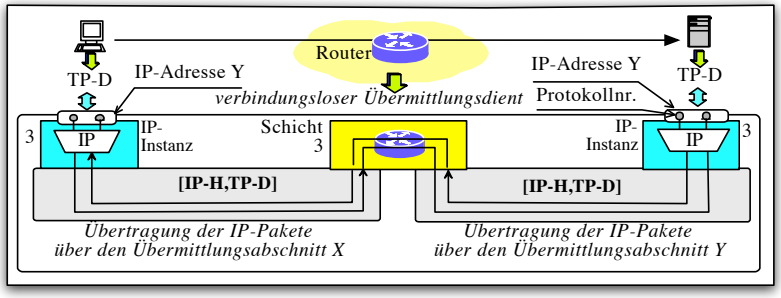


Abb. 1.4-3: Struktur der verbindungslosen Netzwerkschicht in IP-Netzen
IT-H: IP-Header, TP-D: Transportprotokolldateneinheit

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*.

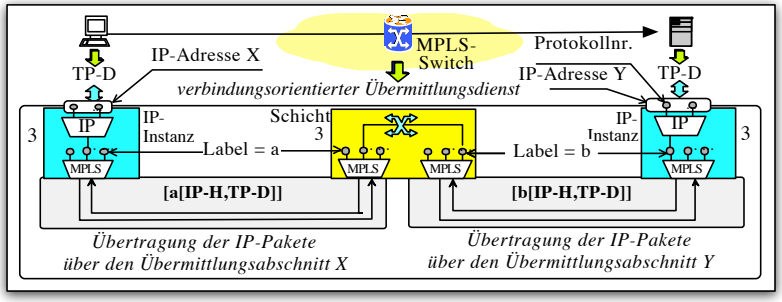


Abb. 1.4-4: Struktur der verbindungsorientierten Netzwerkschicht in IP-Netzen
IT-H: IP-Header, TP-D: Transportprotokolldateneinheit

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].

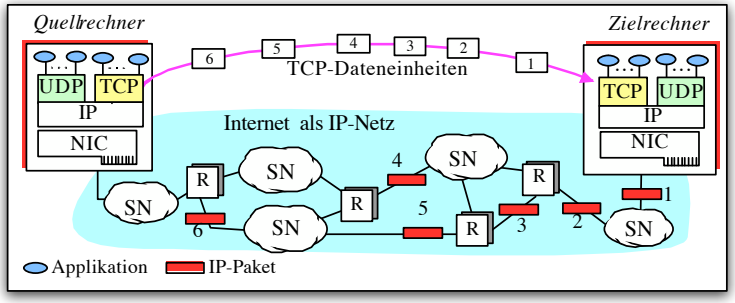


Abb. 1.4-5: Prinzip der Kommunikation im Internet – Datagramm-Prinzip
R: Router, SN: IP-Subnetz, NIC: Network Interface Card (Controller)

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.

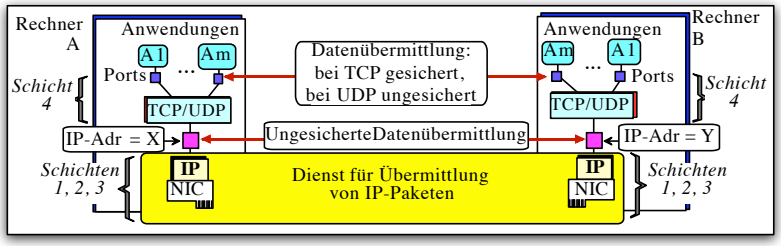


Abb. 1.4-6: Vereinfachte Struktur von Rechnern am IP-Netz
A: Applikation, Adr: Adresse, NIC: Network Interface Controller

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.

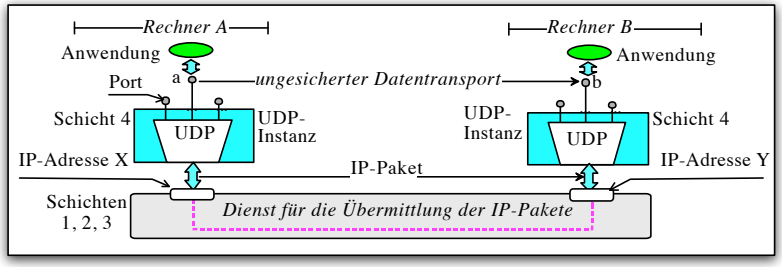


Abb. 1.4-7: Transportschicht mit UDP; ungesicherter Datentransport

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].

Verbindungsorientierte 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].

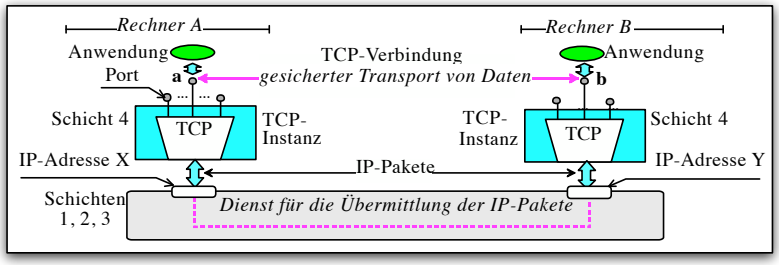


Abb. 1.4-8: Transportschicht mit TCP; gesicherter Datentransport

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

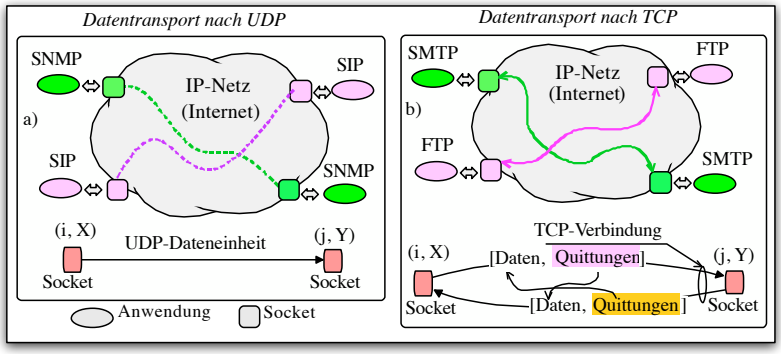


Abb. 1.4-9: Datentransport zwischen Anwendungen: a) beim UDP-Einsatz, b) beim TCP-Einsatz

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`

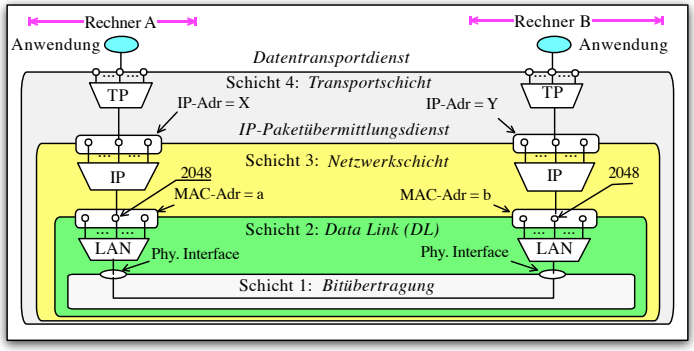


Abb. 1.4-10: Multiplexmodell der Protokollfamilie TCP/IP beim IP-Netz auf LAN-Basis
TP: UDP bzw. TCP

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.

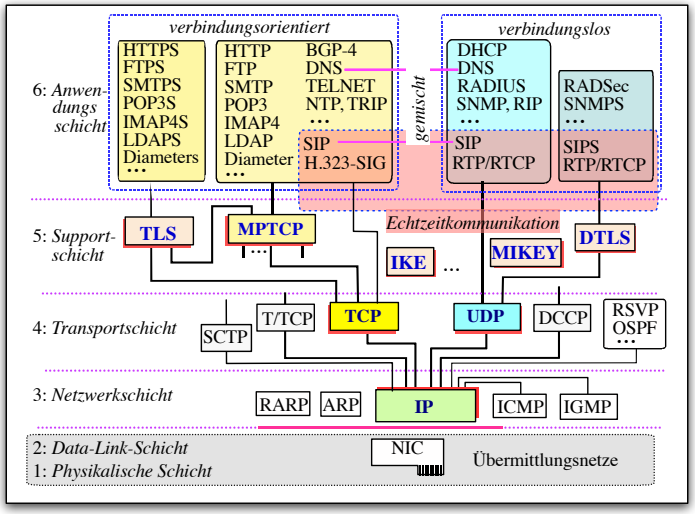


Abb. 1.5-1: Protokolle der Familie TCP/IPv4 im Schichtenmodell
NIC: Network Interface Card (Adapterkarte)

1.5.1 Protokolle der Netzwerkschicht

Die Netzwerkschicht im Schichtenmodell für TCP/IP beschreibt u.a., wie die IP-Netze logisch auf *IP-Subnetze* aufgeteilt werden können und wie die Daten in Form von IP-Paketen in einzelnen IP-Subnetzen und zwischen ihnen übermittelt werden. Die Protokolle der Netzwerkschicht sind:

- **IP**: *Internet Protocol* liegt sowohl in der alten Version 4 (IPv4) als auch in der neuen Version 6 (IPv6) vor. IPv4 und IPv6 sind unterschiedliche Implementierungen auf der Netzwerkschicht und nutzen getrennte Adressräume bzw. Adressierungsverfahren. Im Detail wird IPv4 in Kapitel 2 und IPv6 in Kapitel 7 dargestellt.
- **ARP**: *Address Resolution Protocol* nutzt einen Broadcast-Dienst innerhalb der Schicht 2 zur dynamischen Ermittlung einer MAC-Adresse eines Rechners im LAN, falls seine IP-Adresse bekannt ist [Abschnitt 2.6.1].
- **RARP**: *Reverse Address Resolution Protocol* unterstützt ebenfalls die Adressierung und stellt das Gegenstück zu ARP dar. Es hat die Aufgabe, für eine MAC-Adresse eine IP-Adresse zu bestimmen [Abschnitt 2.6.3].
- **ICMP**: *Internet Control Message Protocol* wird für die Übermittlung von Fehlermeldungen und anderen Kontrollangaben verwendet [Abschnitt 2.7].
- **IGMP**: *Internet Group Management Protocol* gilt als Erweiterung von ICMP und dient vornehmlich dazu, das Management von *Multicast-Gruppen* in IP-Subnetzen zu unterstützen [Abschnitt 2.8.2].

  Bemerkung: Die Protokolle ICMP und IGMP werden üblicherweise der Schicht 3 im TCP/IP-Schichtenmodell zugeordnet. Da die Nachrichten dieser Protokolle in IP-Paketen übermittelt werden, könnte man ICMP und IGMP nach den im OSI-Referenzmodell geltenden Prinzipien zwar der Schicht 4 zuordnen, aber ICMP und IGMP sind keine Transportprotokolle.

1.5.2 Protokolle der Transportschicht

In der Transportschicht befinden sich die Protokolle für die Unterstützung der verbindungsorientierten und der verbindungslosen Kommunikation sowie andere spezielle Protokolle. Die wichtigsten sind:

- **TCP**: *Transmission Control Protocol* ermöglicht die verbindungsorientierte Kommunikation zwischen Rechnern. Hierbei wird zwischen ihnen eine virtuelle Verbindung aufgebaut, die als TCP-Verbindung bezeichnet wird. Eine TCP-Verbindung kann als 'Straße' mit zwei entgegen gerichteten Spuren angesehen werden [Abb. 1.4-9b]. Somit ist TCP ein *verbindungsorientiertes Transportprotokoll*. Durch die Realisierung der Fehler- und der Flusskontrolle garantiert TCP einen zuverlässigen Datentransport. Da bei TCP die zu übertragenden Byte nummeriert werden, ist TCP ein *bytestream-orientiertes Protokoll*. TCP wird in Abschnitt 3.3 beschrieben.
- **UDP**: *User Datagram Protocol* erlaubt lediglich eine verbindungslose Kommunikation zwischen Rechnern, bei der keine virtuelle Verbindung aufgebaut wird. Somit ist UDP ein *verbindungsloses Transportprotokoll*. Bei UDP erfolgt keine Fehler- bzw. Flusskontrolle, sodass UDP im Gegensatz zu TCP keinen zuverlässigen Datentransport garantiert. Auf UDP geht Abschnitt 3.2 ein.
- **T/TCP**: *Transaction TCP* ist eine Ergänzung von TCP im Hinblick auf die Unterstützung sog. Transaktionen. Unter einer Transaktion versteht man einen Kommunikationsvorgang, der aus mehreren Phasen besteht, die alle korrekt durchgeführt werden müssen.
- **SCTP**: *Stream Control Transmission Protocol* ermöglicht genau wie TCP die verbindungsorientierte Kommunikation zwischen Rechnern. Bei SCTP wird eine virtuelle Verbindung, die sog. *SCTP-Assoziation*, aufgebaut [RFC 4960]. Eine SCTP-Assoziation kann als 'Autobahn' mit einer beliebigen Anzahl von entgegen gerichteten Spuren angesehen werden. Daher ist SCTP ein verbindungsorientiertes Transportprotokoll. Auf SCTP geht Abschnitt 3.6 näher ein.
- **RSVP**: *ReSource ReserVation Protocol* ist kein Transportprotokoll, sondern ein Protokoll für die Reservierung von bestimmten Netzressourcen, wie z.B. der Bandbreite in Leitungen, um die Anforderungen der Echtzeitkommunikation zu erfüllen [RFC 2205]. Diese Anforderungen sind unter dem Begriff *Quality of Service* (QoS) bekannt. RSVP wird erweitert und als Signalisierungsprotokoll in (G)MPLS-Netzen verwendet. Dies wird in Abschnitt 11.5 näher dargestellt.
- **OSPF**: *Open Shortest Path First* ist ein Routing-Protokoll, das vor allem bei Internet-Routern Verwendung findet [RFC 5340]. OSPF wird in Abschnitt 10.3 ausführlich dargestellt.

  Bemerkung: Die *Routing-Protokolle* [Kapitel 10] werden in der Literatur der Netzwerkschicht (Schicht 3) zugeordnet, also der Schicht, in der IP angesiedelt ist. Da die OSPF-Nachrichten direkt in IP-Paketen übermittelt werden, lässt sich OSPF nach den im TCP/IP-Schichtenmodell geltenden Prinzipien nicht der Netzwerkschicht zuordnen, sondern der Transportschicht. Bei der Übermittlung von Nachrichten des Routing-Protokolls RIP (*Routing Information Protocol*) wird UDP verwendet; somit ist RIP der Anwendungsschicht zuzuordnen. Das Routing-Protokoll BGP (*Border Gateway Protocol*) nutzt dagegen TCP und ist daher ebenfalls der Anwendungsschicht zuzuordnen.

1.5.3 Protokolle der Supportschicht und für Echtzeitkommunikation

Mit der wachsenden Durchdringung des Internet und der Substitution der klassischen Telefondienste durch VoIP stellte sich die Notwendigkeit, die Transportdienste der Internetprotokolle stärker auf die Eigenschaften der Anwendungen abzustimmen. In diesem Zusammenhang können die Protokolle wie TLS, DTLS und MPTCP je nach Sichtweise als *Application-Support-Protokolle* oder als *Transport-Support-Protokolle* betrachtet werden und die wichtigsten von ihnen sind:

- **MPTCP**: *Multipath TCP* stellt für Anwendungen mehrere TCP-Verbindungen bereit, die über unterschiedliche IP-Adressen und über mehrere parallel verlaufende Datenpfade (*Multipath*) geführt werden können [RFC 6824].
- **TLS**: *Transport Layer Security* ist der Nachfolger der *Secure Socket Layer* (SSL), die von der Firma Netscape entwickelt wurde, um die *Webtransaktionen* zu sichern. Bei TLS [RFC 5246] werden die Daten verschlüsselt, deren Integrität gesichert und die beiden Kommunikationspartner können sich gegenseitig authentisieren [Abschnitt 5.3].
- **DTLS**: *Datagram TLS* ist die UDP-nutzende Variante von TLS [RFC 6347].
- **IKE**: Das *Internet Key Exchange Protokoll* [RFC 5996] ist ein Supportprotokoll und wird bei IPsec (*IP Security*) hierzu verwendet, damit die IP-Instanzen in zwei kommunizierenden Rechnern vereinbaren können, auf welche die Art die Kommunikation zwischen ihnen geschützt werden soll. Dies wird als Sicherheitsvereinbarung *Security Association* (SA) bezeichnet.
- Die weiteren Protokolle der Supportschicht sind **SOCKetS**, genauer SOCKSv5 gemäß RFC 1928, das als Autentisierungsprotokoll zwischen einem Client und einem Proxy-Server dient [Abschnitt 6.1], sowie das Protokoll **NBoT** (*Network Basic Input Output System (NetBIOS) over TCP*), das als Programmschnittstelle (API) für CIFS, also das *Common Internet File System* bzw. SAMBA von Windows- Unix-Endsystemen genutzt wird und gemäß RFC 1001 und 1002 Transportdienste zur Übertragung von NetBIOS über IP-Netze bereit stellt.

Echtzeit-kommunikation

Die *Echtzeitkommunikation* stellt eine besondere Klasse von Anwendungen dar, deren Anforderungen nicht unmittelbar auf der Transportschicht umgesetzt werden können; sondern verlangen spezielle Implementierungen, die mit dem *Real-time Transport Protocol* (**RTP**) [RFC 3550] und den 'zuarbeitenden' *Real-time Transport Control Protocol* (**RTCP**) [RFC 3605] sowie dem *Real Time Streaming Protocol* (**RTSP**) [RFC 2326] realisiert wurden. Bei der Echtzeitkommunikation dient SIP (*Session Initiation Protocol*) als *Signalisierungsprotokoll* dazu, u.a. Verbindungen auf- und abzubauen.

Abgesicherte Echtzeit-kommunikation

Abgesicherte Echtzeitkommunikation kann in Ergänzung zu den unverschlüsselten Echtzeitprotokollen mittels der Pendants **SRTP** (*Secure RTP*) [RFC 3711] und **SRTCP** (*Secure RTCP*) [RFC 3711] erzielt werden. Hierbei wird das Supportprotokoll MIKEY (*Multimedia Internet KEYing*) verwendet, damit kommunizierende Einrichtungen untereinander vereinbaren können, wie die Kommunikation zwischen ihnen geschützt werden soll.

1.5.4 Komponenten der Anwendungsschicht

In der Anwendungsschicht sind, neben den bereits erwähnten Protokollen für die Echtzeitkommunikation, verschiedene Funktionskomponenten angesiedelt. Diese lassen sich in die folgenden vier Gruppen aufteilen:

- **Anwendungsprotokolle** werden im Weiteren als Protokolle wie z.B. FTP und HTTP verstanden, mit dem sich eine bestimmte Anwendung realisieren lässt.
- **Netzdienstprotokolle** bezeichnen Protokolle (z.B. DHCP), mit dem ein bestimmter Netzdienst erbracht wird. Beispielsweise können mit DHCP-Hilfe die IP-Adressen dem Rechner nach Bedarf dynamisch zugeteilt werden. Dies stellt einen Netzdienst dar. Auch Routing-Protokolle, wie z.B RIP, können als Netzdienstprotokolle betrachtet werden. Ein weiteres, in seiner Bedeutung nicht zu unterschätzendes Netzdienstprotokoll ist der *Zeitstempeldienst*, der z.B. in Form des *Network Time Protocols* (NTP) vorliegt [RFC 5905] und zur Synchronisation von Rechnern und Netzknoten (Router) dient.
- **Benutzerdienstprotokolle** sind spezielle Kommandos unter UNIX und LINUX mit denen (entfernte) Netzdienste in Anspruch genommen werden können. Allgemein werden diese als `r`-*Kommandos* bezeichnet, wobei sowohl verbindungslose Anwendungen wie `rwho`, `rexec` und `rsh`, als auch die verbindungsorientierten Kommandos `rlogin`, `rcp`, `rexec` genutzt werden können.

Je nachdem, ob ein Protokoll der Anwendungsschicht das verbindungsorientierte Transportprotokoll TCP oder das verbindungslose UDP verwendet, lassen sich die Protokolle der Anwendungsschicht als *verbindungsorientiert*, *verbindungslos* bzw. *gemischt* klassifizieren [Abb. 1.5-1].

Abb. 1.5-2 benennt die wichtigsten Funktionskomponenten der Anwendungsschicht.

Verbindungsorientierte Anwendungsprotokolle

Verbindungsorientierte Anwendungsprotokolle sind u.a.:

- **HTTP**: *Hypertext Transport Protocol* ist neben SMTP das wichtigste Anwendungsprotokoll im Internet. HTTP sorgt für die Datenübermittlung zwischen Webbrowser und Webserver. *HTTP over TLS* wird als HTTPS bezeichnet.
- **SMTP**: *Simple Mail Transport Protocol* ermöglicht die Übermittlung von E-Mails im Internet. Heute wird in der Regel das *Extended SMTP* (ESMTP) eingesetzt, das eine 8-Bit-transparente Übermittlung ermöglicht.
- **TELNET** ist ein Protokoll, mit dem sich der Anwender in einer interaktiven Sitzung auf einem entfernten Computer einloggen kann und gilt als Urvater der anwendungsbezogenen TCP/IP- Protokolle.
- **FTP**: *File Transfer Protocol* dient zur Übermittlung von Dateien zwischen zwei über ein IP-Netz verbundenen Rechnern. Es ist bewusst einfach und robust aufgebaut, sodass die Datenübertragung auch über in der Qualität schlechte Verbindungen (z.B. Satellitenkommunikation) möglich ist. FTP kann auch die TLS-Funktion nutzen. Man spricht dann von *FTPS*.

Verbindungslose Anwendungsprotokolle

Verbindungslose Anwendungsprotokolle sind u.a.:

- **SNMP**: *Simple Network Management Protocol* ermöglicht die Abfrage der Zustände von Netzwerkkomponenten und liegt dem Netzwerkmanagement zugrunde.

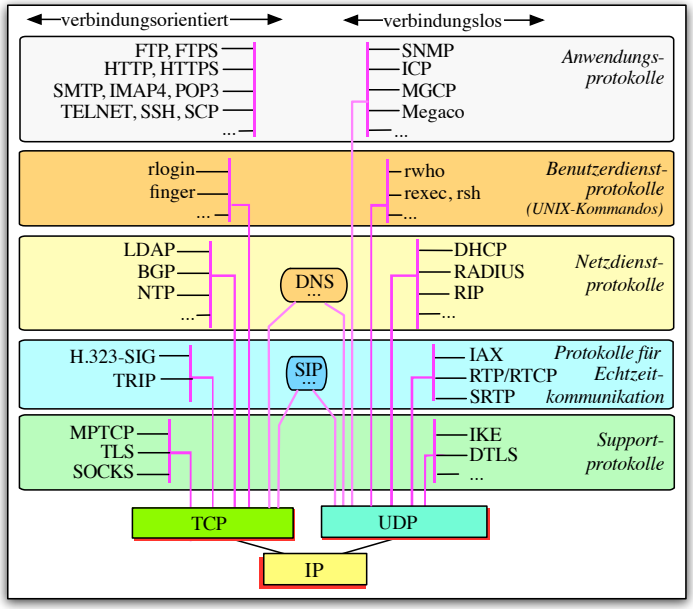


Abb. 1.5-2: Einordnung der Protokolle entsprechend ihrer Kommunikationsschicht

- **ICP**: *Internet Cache Protocol* ist ein Protokoll, nach dem Web-Caching-Systeme im Internet kooperieren [BRS03].
- **MGCP**: *Media Gateway Control Protocol* dient zwischen den *VoIP-Gateways* für die Anbindung herkömmlicher Komponenten an VoIP-Systeme [Bad10]. Das Protokoll Megaco entspricht der Funktion nach dem MGCP.

Verbindungsorientierte Netzdienstprotokolle sind u.a.:

Verbindungsorientierte Netzdienstprotokolle

- **NTP**: *Network Time Protocol* realisiert die Zeitsynchronisation der Rechner und Netzkomponenten im Internet [RFC 5905]. Einige Client/Server-Anwendungen verlangen die gleichen 'Uhrzeit' zu ihrem Funktionieren.
- **BGP**: *Border Gateway Protocol* dient der Übermittlung von Routing-Informationen zwischen autonomen Systemen (AS) [Abschnitt 10.4].
- **LDAP**: *Lightweight Directory Access Protocol* verwendet man bei der Realisierung verteilter Verzeichnisdienste und wird vor allem als Backend für die Benutzerauthentisierung genutzt [Abschnitt 14.3].

Verbindungslose Netzdienstprotokolle sind u.a.:

Verbindungslose Netzdienstprotokolle

- **DHCP**: *Dynamic Host Configuration Protocol* kann die dynamische Vergabe von IP-Adressen und weiterer Netzparameter übernehmen. DHCP wird im Abschnitt 5.2 beschrieben.
- **RADIUS**: *Remote Dial-In User Service* wird in Abschnitt 14.2 besprochen und ermöglicht die Berechtigungsprüfung von Benutzern, die auf Netzressourcen zugreifen wollen. Dies kann beim Provider-Zugang ins Internet, beim Anmelden im WLAN aber bereits auch am Anschluss an einen Ethernet-Switch der Fall sein. Das Nachfolge-

Protokoll *Diameter* [RFC 6733] wird hauptsächlich im IMS (*IP Multimedia Subsystem*) eingesetzt [Bad10].

- **RIP**: *Routing Information Protocol* dient als internes Routing-Protokoll vornehmlich in kleineren IP-Netzen.

Das wohl wichtigste Protokoll im Internet ist **DNS** (*Domain Name System*), das sowohl TCP als auch UDP nutzt [Kapitel 4]. DNS ist ein gemischtes Netzdienstprotokoll.

Protokolle für Echtzeitkommunikation

Protokolle zur Unterstützung der Echtzeitkommunikation sind u.a.:

- **RTP**: *Real-time Transport Protocol* hat die Aufgabe, zeitkritische Anwendungen wie Audio- und Videokommunikation über ein IP-Netz zu unterstützen. Ihm steht RTCP (*RTP Control Protocol*) zur Seite. RTP ist die Grundlage für VoIP und für WebRTC. Eine erweiterte RTP-Version zur sicheren Audio- und Videokommunikation trägt die Bezeichnung SRTP (*Secure RTP*) [Bad10].
- **SIP**: Das *Session Initiation Protocol* dient als sog. Signalisierungsprotokoll bei der Echtzeitkommunikation und wird hauptsächlich über UDP eingesetzt; es kann aber auch TCP nutzen.
- **IAX**: *Inter-Asterisk eXchange* ist ein kombiniertes Protokoll für die Signalisierung (z.B. bei VoIP) und für den Transport von Echtzeitdaten (Audio, Video) über IP-Netze. Die Version 2 von IAX beschreibt das IETF-Dokument [RFC 5456]. IAX2 nutzt UDP für den Transport seiner Nachrichten. Bei IAX2 unterscheidet man zwischen *zuverlässigen* und *unzuverlässigen* Nachrichten. Die zuverlässigen Nachrichten transportieren die *Signalisierungsangaben* und werden von der Empfangsseite bestätigt. Die unzuverlässigen Nachrichten transportieren Echtzeitdaten und werden nicht bestätigt. IAX2 hat viel gemeinsam mit dem Protokoll SCTP.
- **TRIP**: *Telephony Routing over IP* wurde der Übermittlung von Routing-Informationen zwischen autonomen Systemen für die VoIP-Unterstützung vorgesehen und daher gilt TRIP als Bruder von BGP [Bad10].

Signalisierungsprotokolle

Bei der Echtzeitkommunikation wischen kommunizierenden Endeinrichtungen müssen für Verbindungen auf- und abgebaut werden. Folglich benötigt man spezielle Protokolle, die dies zwischen IP-Telefonen bei VoIP, wie auch bei Webbrowsern unter Nutzung des WebRTC-Dienstes realisieren. Diese Protokolle werden als *Signalisierungsprotokolle* bezeichnet. Neben dem Protokoll SIP gehört hierzu die Signalisierung nach dem ITU-T-Standard H.323 (kurz H.323-SIG), die über TCP abgewickelt wird. Für weitere Informationen sei auf [Bad10] verwiesen.

1.6 Sicherheit der IP-Kommunikation

Die Architektur der TCP/IP-Protokollfamilie sah zunächst nur technische *Sicherungsmaßnahmen* für die Kommunikation vor: War zunächst TCP/IP als quasi geschlossenes System im Rahmen des ARPANet vorgesehen, so war die in den 70er und 80er Jahre des letzten Jahrhunderts vorherrschende Nutzung des Internet im wissenschaftlichen Betrieb zu sehen.

Alice, Bob und Eve

Der erste Schritt bestand natürlich darin, das Internet überhaupt für die Anwender bereit zu stellen; also die *Verfügbarkeit* (*Availability*) zu sichern. Die nächste zentrale Anforderung bestand daran, dass die Daten vollständig und unverfälscht von *A* nach *B* (über

C) gelangen und somit die *Integrität* der Nachrichten (= *Datenpakete*) zu gewährleisten. Mit der kommerziellen und quasi privaten Nutzung des Internet gewann die vertrauliche Übertragung von Nachrichten eine bedeutende Rolle: Aus *A* und *B* wurden *Alice* und *Bob*, die ihre Internet-Konversation gegenüber dem *Eavesdropper Eve* schützen müssen.

IT-Security
Schutzziele

Diese Überlegungen bilden den Kern der drei *Schutzziele* der IT-Security:

- **Confidentiality**: Schutz vor *Spionage* durch Sicherstellung der Vertraulichkeit,
- **Integrity**: Schutz vor Verfälschungen der Daten im Hinblick auf *Korrumpierung* und *Manipulation*,
- **Availability**: Schutz vor Datenverlusten und Ausfall der IT-Infrastruktur durch technische Fehler und *Sabotage*.

Die Technik der Netze kann zumindest die *Integrität* der Nachrichten sicherstellen. Bei der *Vertraulichkeit* bedarf es zusätzlicher Anstrengungen, die nur durch eine *Verschlüsselung* der zu übertragenden Informationen gelöst werden kann. Die Frage der *Verfügbarkeit* stellt sich technisch im Hinblick auf verfügbare Ressourcen, wie *redundante Datenpfade* und die Redundanz im Hinblick auf die verarbeitenden Komponenten und die der Datenhaltung.

PKI

Mit der Erfindungen der asymmetrischen Verschlüsselung Mitte der 70er Jahre und der avisierten Kommerzialisierung des Internet Mitte der 90er Jahre, ergab sich die Möglichkeit und zugleich Notwendigkeit, den Internet-Datenverkehr zu verschlüsseln. In den Jahren zuvor war das Konzept der *Public Key Infrastructure* (PKI) entwickelt worden und mit den nun verfügbaren Rechnerressourcen fand dies mittels der *Secure Socket Layer* (*SSLeay*) zunächst Einzug in den damals dominierenden Netscape Webbrowser und den Apache Webserver.

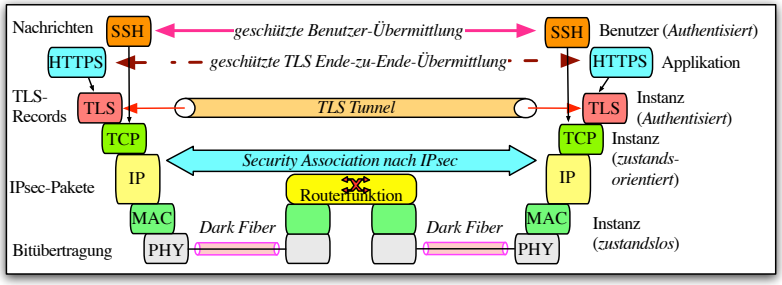


Abb. 1.6-1: Gesicherte IP-Kommunikation auf den verschiedenen Schichten
PHY: Physikalische Schicht, MAC: Media Access Control Schicht, IP: Internet Protocol, IPsec: IP Security, TCP: Transmission Control Protocol, TLS: Transport Layer Security, HTTPS: Hypertext Transport Protocol (Secure), SSH: Secure Shell

Abb.1.6-1 illustriert die heute gängigen Verfahren zur Absicherung der (Internet-)Kommunikation (mittels Verschlüsselung) auf den unterschiedlichen Kommunikationsschichten:

Dark Fiber

- Bei der physikalischen Übermittlung der Signale kann eine *Dark Fiber* genutzt werden. Eine Dark Fiber muss nicht im eigentlichen Sinne ein Glasfaserkabel sein; jedoch beinhaltet der Terminus, dass immer eine *eigene physikalische Übertragungsstrecke* vorhanden ist, die nicht mit anderen Teilnehmern geteilt wird, sondern in der die Bit sozusagen 'privat' transportiert[3] werden.

[3] *Google* verwendet 'Dark Fiber' bei den Providern, um seine Rechenzentren zu koppeln.

Die Übertragung per 'Dark Fiber' bedeutet nicht unbedingt, dass die Signale zusätzlich verschlüsselt sind. Wie Abb. 1.6-1 zeigt, kann die Anbindung per 'Dark Fiber' nur Punkt-zu-Punkt, d.h. zwischen den Netzknoten erfolgen. Zudem ist die Übertragung *zustandslos*, da weder das Verfahren ausgehandelt werden muss, noch die darauf aufbauenden Anwendung hierüber informiert sind.

IPsec

- Auf der Netzwerkschicht kann als Sicherheitsprotokoll *IP Security* (IPsec) [Abschnitt 5.4] genutzt werden. Alternativ kann das von Dan Bernstein erfundene *CurveCP-Verfahren* [http://curvecp.org/] als quasi *Transportverschlüsselung* Verwendung finden. Im Gegensatz zu IPsec, wird hier keine PKI voraus gesetzt.

  Bei beiden Verfahren kann die Verschlüsselung entweder zum nächsten IP-Peer-Knoten, oder zwischen Quelle und Ziel aufgesetzt werden. Auch hier gilt, dass die Verschlüsselung für die Anwendung nicht 'sichtbar', also transparent ist, wenn auch eine Verhandlung zum Aufbau der notwendigen Sicherheitsverfahren gefordert ist; bei IPsec als *Security Association* bezeichnet.

TLS

- Das auf TCP aufbauende Verfahren der *Transport Layer Security* (TLS) stellt den Nachfolger des *Secure Socket Layer* (SSL) Protokolls dar und bildet das heutige Rückgrat des *Internet E-Commerce* [Abschnitt 6.2]. TLS ist immer an eine Anwendung (*Applikation*) gebunden und muss von dieser explizit angefordert werden.

  TLS setzt im Grunde eine funktionierende PKI voraus, wobei einige Verschlüsselungsverfahren auch ohne diese auskommen können. TLS ist ein *zustandsorientiertes* Sicherheitssystem; auch wenn es zunächst als transparente Anwendungs-Supportschicht gedacht war. Gebräuchlich sind neben HTTPS vor allem die E-Mail-Protokolle SMTPS, POP3S und IMAPS.

SSH, PGP

- Einige verbreitete Protokolle der Anwendungsschicht besitzen intrinsische Mechanismen, um die zu übertragenden *Nachrichteninhalte* automatisch zu verschlüsseln. Zunächst sei hier das Protokoll *SSH* mit seinen 'Ablegern', *SCP*, *SFTP* und *rsync* genannt. Auch das E-Mail-Verschlüsselungsprotokoll *Pretty Good Privacy PGP* gehört zu dieser Kategorie.

Neben der Frage der *Verschlüsselung* kommt auch immer die der *Integrität*, d.h. Unverfälschtheit der übertragenen Daten hinzu, sowie inwiefern die *Authentizität* der Kommunikationspartner – im Sinne von 'der richtige' – gewährleistet ist. Vertraulichkeit durch Verschlüsselung und die Integrität der übertragenen Informationen können durch technische Maßnahmen garantiert werden; Authentizität des *Kommunikationspartners* allerdings nur im Rahmen eines Vertrauensmodells.

Während wir die ersten beiden Aspekte im folgenden Abschnitt diskutieren möchten; greifen wir das Thema der Benutzer-Identifikation erst im Kapitel 13 wieder auf.

1.6.1 Gesicherte und sichere Datenübertragung

Im OSI-Referenzmodell ist vor allem die *Datensicherungsschicht* verantwortlich für die Sicherstellung der Integrität der übertragenen Nutzdaten in Form der *DL-Frames* (vgl. Abschnitt 1.3). In den vergangenen Jahren hat sich jedoch ein erweitertes Verständnis dieser Sicherungsaufgaben entwickelt:

- *Bitfehler*: Auf der *physikalischen Schicht* lässt sich mit geeigneten *Kodierungsverfahren Bitfehler* erkennen und korrigieren. Heutige Verfahren nutzen komplexe Kodierungsschemata, die eine *Forward Error Correction* (FEC) ermöglichen.
- *Paketfehler*: Die Datenpakete auf den Schichten 2, 3 und 4 können mittels eines *Cyclic Redundancy Codes* (CRC) im *Trailer* ergänzt werden. Es lassen sich hierdurch nicht nur Fehler in den Paketen erkennen, sondern diese auch prinzipiell korrigieren, wobei in der Regel aber die fehlerhaften Datenpakete neu angefordert werden.
- *Nachrichtenintegrität*: Die Integrität der gesamten übermittelten Nachricht lässt sich über eine *Nachrichtenverschränkung* der einzelnen Pakete realisieren, wie dies beispielsweise mittels des *Cipher Block Chaining* geschieht. Hierfür ist ein *Initialisierungsvektor* (IV) notwendig, den die beiden kommunizierenden Parteien vereinbaren müssen.
- *Nachrichtenauthentizität*: Auf der *Anwendungsschicht* besteht die Anforderung nicht nur darin, evtl. *Korruption* der Nutzdaten zu erkennen, sondern auch – im gleichen Zug – die *Authentizität* des Absenders zu ermöglichen. Hierzu kann eine un-umkehrbare *Hashfunktion* genutzt werden. Im einfachen Fall ist der Eingangswert der Hashfunktion die Nachricht selbst; wird aber ergänzend ein vereinbartes *Schlüsselwort* hinzugefügt, erhält man einen *keyed Hashwert*. Der *Hashwert*, als Ergebnis der Hashfunktion, hat immer eine konstante Länge unabhängig vom Eingangswert.

Abb. 1.6-2 veranschaulicht die möglichen Verfahren zur Realisierung einer sicheren und gesicherten Datenübertragen auf den verschiedenen Kommunikationsschichten.

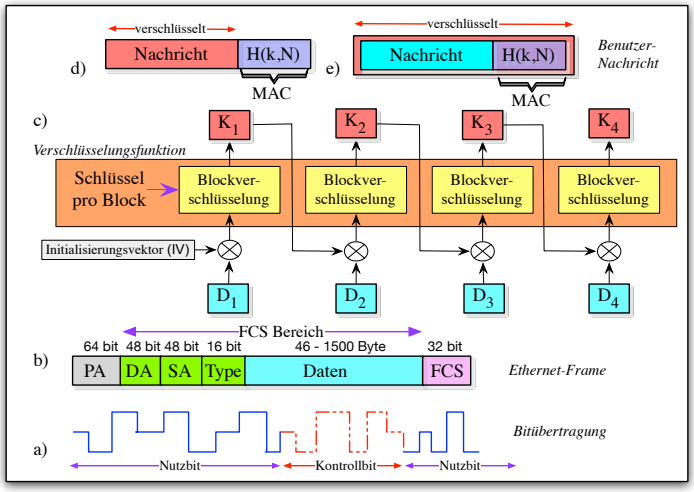


Abb. 1.6-2: Methoden zur sicheren und gesicherten Datenübertragung a) auf physikalischer Schicht durch Hinzufügen von *Kontrollbit* b) auf Data-Link durch eine *Frame Control Sequence* (FCS), c) auf der Supportschicht durch Verschränkung der Records und d) durch Verschlüsselung und anschließendes Hinzufügen eines *Message Authentication Checks* (MAC) bzw. e) durch Hinzufügung eines MAC und gemeinsamer Verschlüsselung mit der Nutznachricht (vgl. Abb. 6.3-5)
PA: Präambel, DA: Destination Address, SA: Source Address, FCS: Frame Control Sequence, H(k,N): Hashfunktion mit Schlüssel k für Nachricht N, $D_1 \ldots D_4$: Klartext-Datenblöcke, $K_1 \ldots K_4$: verschränkte (und ggf. verschlüsselte) Kryptodatenblöcke

Mechanismen zur Bitfehlerkorrektur

Die zu übermittelnden *Informationen* (Nachrichten, Pakete) werden zunächst in eine strukturierte Bitfolge umgesetzt und auf der physikalischen Schicht letztlich als elektrisches/optisches Signal übertragen. Diese Signale erleiden bei der Übertragung Verluste und Verfälschungen; generell wird das Signal entlang der Übertragungsstrecke geschwächt. Signale können aber auch verformt und durch Fremdeinflüsse (Störstrahlung) überdeckt werden. Will man digitale Informationen über eine Übertragungsstrecke versenden, sind Vorkehrungen zu treffen, sodass die Ausgangsinformation des Senders unverfälscht und vollständig als Eingangsinformation beim Empfänger ankommt.

Hierbei geht es um zwei zentrale Fragen:

BER
1. Wie hoch ist die Bitfehlerrate (*Bit Error Rate*)?

PER
2. Wie viele resultierende Paketfehler gibt es (*Packet Error Rate*), sodass das Paket neu übertragen werden muss?

Längsparität
Die einfachste Möglichkeit besteht darin, nach einer Anzahl von Nutzbit *Kontroll-* bzw. *Paritätsbit* folgen zu lassen [Abb.1.6-2a]. Heute werden mehrstufige Verfahren eingesetzt, wie z.B. die *4B/5B* Codierung bei 100 Mbit/s (Fast-)Ethernet, die so gewählt sind, dass nicht nur statistisch gleichförmig auftretende Fehler, sonder auch *burst-artige* Störungen erkannt werde können. Hierzu zählt im Besonderen das *Reed-Solomon-Verfahren*, das beispielsweise bei der Signalübertragung auf Monomode-Glasfaserkabeln entsprechend dem Standard OC-192 einsetzt wird. Auch beim aktuellen Entwurf von 10 Gbit/s Ethernet findet der *Reed-Solomon Code* entsprechend `RS(255,239)`, d.h. mit 239 Nutzbit und 15 Kontrollbit Einsatz.

FEC
Diese *Forward Error Correction* führt zu einer substanziellen Reduktion der resultierenden Bitfehlerrate, was gerade für potentielle störanfällige Übertragungsmedien, wie auch z.B. WLAN von Bedeutung ist. So können bspw. Bitfehlerraten `BER` = 10^{-4} (ohne FEC) auf lediglich 10^{-8} (mit FEC) gesenkt werden.

Sicherstellung der Integrität eines Datenpakets

Während die Bedeutung der modernen Verfahren der *Forward Error Correction* erst mit der Entwicklung der schnellen LANs erkannt wurde, war die Sicherstellung der Paket-Integrität originärer Bestandteil der OSI-Schicht 2, also der *Sicherungsschicht*.

Ein *Paketfehler* ist dann zu verzeichnen, falls zumindest ein nicht-korrigierter *Bitfehler* im Paket auftritt. Treten die Bitfehler statistisch unabhängig auf, ist die Anzahl der Paketfehler proportional der Bitzahl im Paket, also der *Paketlänge*.

CRC
Zur Erkennung und ggf. Korrektur von Paketfehlern wird von einer *Cyclic Redundancy Checksum* (CRC) Gebrauch gemacht [Abb.1.6-2b]. Ausgegangen wird üblicherweise von einem *Generator Polynom* $G(x)$, das ein Bit länger sein muss, als die zu überprüfende Bit-Sequenz. Die Nutzdaten werden durch dieses Polynom dividiert und der Teilerrest als *Checksumme* zu den Eingangsdaten für die Überprüfung zu den Nutzdaten hinzugefügt.

Querparität
Bei dieser *Querparität* wird der ermittelte Wert für das Frame, d.h. die *Checksumme* als sog. *Frame Check Sequence* (FCS) dem Payload zugefügt [Abb. 1.6-2b].

IP *Checksumme*
Die Berechnung der *Checksumme* bei IP-Paketen und TCP-Segmenten wird entsprechend RFC 1071 als 'Eins-Komplement' vorgenommen und umfasst mit dem resultierenden 16-Bit CRC lediglich die jeweilige Header-Informationen.

Die *Checksumme* hat die bemerkenswerte Eigenschaft der *Additivität*, d.h. für zwei Nachrichtenteile N_1 und N_2, die gemeinsam die Nachricht N bilden, gilt:

$$CRC(N) = CRC(N_1 \parallel N_2) = CRC(N_1 \parallel CRC(N_2)) \text{ bzw.}$$
$$CRC(N_1 \oplus N_2) = CRC(N_1) \oplus CRC(N_2)$$

Hierbei ist $\parallel$ der Konkatinierungs- und $\oplus$ der Additions-Operator für eine Bitfolge. Wird also der Inhalt der Nutzinformation um den Wert '1' geändert, ändert sich auch der resultierende CRC um genau diesen Wert. Diese Eigenschaft machen sich die Internet-Router zu eigen, indem sie bei der Herabsetzung eines 'Hops' im IP-Paket nicht die gesamte Header-*Checksumme* neu berechnen, sondern nur den eingetragenen Wert dekrementieren.

Im Umkehrschluß ist es möglich, durch geeignete Manipulation der Nutzdaten, z.B. indem ein geeignetes *Padding* hinzugefügt wird, den Wert der *Checksumme* auf den gewünschten Wert 'zu trimmen'. Aus diesem Grund ist eine *Checksumme* nur ein notwendiger Garant einer Unverfälschtheit, aber kein hinreichender.

Padding = Trimmen des CRC

Datenblockverschränkung

Mit den vorgestellten Mitteln der *Forward Error Correction* und der Berechnung einer *Checksumme* ist es möglich, die technische Integrität jedes einzelnen, übertragenen Datenpakete (bis auf einen sehr kleinen Fehlerquotienten) sicher zu stellen. Eine Nachricht auf der Applikationsschicht umfasst jedoch in der Regel mehrere Datensegmente bzw. IP-Pakete: Wie kann nun sicher gestellt werden, dass die Gesamtnachricht unverfälscht und komplett übertragen wurde?

TCP stellt als verbindungsorientiertes Transportprotokoll die Reihenfolge und den Umfang der übertragenen Datenbyte sicher; will man jedoch ein erhöhtes Sicherungsniveau erzielen, ist man auf die zusätzlichen Dienste der *Transport Layer Security* (TLS) angewiesen. TLS bietet bei der Übermittlung seiner *Records* (vgl. Abschnitt 6.2) neben der Sicherung der einzelnen Pakete auch die Möglichkeit, die zu übertragenen Daten blockweise zu *verschränken*. Ist die Länge einer Nachricht bekannt, kann wie folgt vorgegangen werden:

TLS-Records

1. Zunächst wird die Nachricht in Datenblöcke konstanter Länge aufgeteilt, z.B. je 128 Bit.
2. Sender und Empfänger kennen einen ebenfalls 128 Bit langen *Initialisierungsvektor IV*, der ein zufälliges Bitmuster enthält.
3. Der erste Datenblock D_1 wird mit dem IV per XOR Operation in einen Datenblock K_1 überführt: $K_1 = D_1 \otimes IV$.
4. Der nachfolgende Datenblock D_2 wird nun mit K_1 per XOR verschränkt: $K_2 = D_2 \otimes K_1$ (allgemein: $K_i = D_i \otimes K_{i-1}$).
5. Dieses Verfahren findet für alle folgenden Datenblöcke Verwendung; der letzte zu übertragende Datenblock muss dann mittels *Padding* auf die 128 Bit-Länge aufgefüllt werden.

IV

Abb. 1.6-2c illustriert den Algorithmus anhand von drei Datenblöcken. Die verschränkten Datenblöcke werden dann wie üblich als Nutzlast in den TLS-Records übertragen. Wie wir in Abschnitt 5.2 sehen werden, findet bei TLS eine zusätzliche Verschlüsselung der einzelnen Datenblöcke statt, die dem CBC-Mechanismus (*Cipher Block Chaining*) seinen Namen gab und in Abb. 1.6-2c durch die farbige Box angedeutet ist.

CBC

Unter Kenntnis des IV kann der Empfänger nur dann die Gesamtnachricht erfolgreich in ihr Original überführen, falls alle Datenblöcke vorhanden sind und korrekt übertragen wurden. Daher ist die Methode der Datenblockverschränkung nur für Nachrichten von bekannter Länge nutzbar, nicht jedoch für die *Datenstromübertragung*.

Verschlüsselung und Authentisierung von Nachrichten
Schließlich zeigt Abb. 1.6-2d, wie Nachrichten durch Verschlüsselung (mit einem geheimen Schlüssel) geschützt übertragen und der Datenhalt zugleich über eine *Hashfunktion* und einem weiteren Schlüssel *k* als *Message Authentication Check* (MAC) gesichert werden kann. Die Details dieser Verfahren wollen wir im Folgenden vorstellen. Abb. 1.6-2d zeigt aber, dass prinzipiell entweder zunächst die Verschlüsselung und dann die MAC-Authentisierung oder umgekehrt, erst die MAC-Authentisierung und abschließend die Verschlüsselung der Gesamtnachricht erfolgen kann. Die korrekte Reihenfolge ist derzeit umstritten.

1.6.2 Hashfunktionen und Nachrichtenauthentisierung

Checksummen dienen als Mittel, sowohl die Integrität von Daten überprüfbar zu machen, als auch im Bedarfsfall fehlerhafte Daten wieder zu rekonstruieren. Sie werden nicht nur bei der Datenübertragung eingesetzt, sondern auch beim (blockweise) Speichern von Daten auf einem physikalischen Datenträger wie Festplatten und USB-Sticks, sowie zur Sicherstellung der Integrität des Hauptspeichers bei Rechnern, im Rahmen eines *Error Correction Codes* (ECC).

Hashfunktionen

Aufgrund der Additivität der *Checksummen*, sind sie jedoch für *kryptographische Sicherung* ungeeignet. Hierzu werden *Einweg-* bzw. *Hashfunktionen* eingesetzt.

Hashfunktion ⇒ Hashwert

Eine Hashfunktion ist eine Rechenvorschrift, mit der eine 'Eingangs'-Zeichenfolge beliebiger Länge in eine 'Ausgangs'-Zeichenfolge konstanter (im Allgemeinen kürzerer) Länge umgewandelt wird: der *Hashwert* oder die *Hashsumme*. Eine Einweg-Hashfunktion funktioniert immer nur in eine Richtung, d.h. aus der 'Eingangs'-Zeichenfolge lässt sich einfach die 'Ausgangs'-Zeichenfolge berechnen, aber es ist aufgrund des *Lawinen-Effekts* praktisch unmöglich, zu einer 'Ausgangs'-Zeichenfolge passende 'Eingangs'-Zeichenfolgen zu berechnen: Ändert sich bei der 'Eingangs'-Zeichenfolge auch nur ein Bit, besitzt der erzeugte Hashwert einen vollkommen anderen, unvorhersehbaren Wert. Dies unterscheidet eine 'Hashsumme' fundamental von einer '*Checksumme*'!

Rainbow Tables und Salt

In der Praxis ist es allerdings möglich, die Hashsumme allgemein verwendeter und fester Eingabegrößen zu tabellieren: *Rainbow Tables*. Aus diesem Grund wird den Eingabedaten häufig ein zusätzlicher Zufallswert, ein *Salt*, hinzugefügt, um diese Umkehrung zu unterbinden.

Die aktuell benutzten Einweg-Hashfunktionen sind:

- MD5 (*Message Digest 5*) [RFC 1321]: Die erzeugt die 'Ausgangs'-Zeichenfolge, der Hashwert, hat eine Länge von 128 Bit (32 hexadezimale Zeichen).
- SHA (*Secure Hash Algorithm*) [RFC 3174] und einem Hashwert von 224 Bit, entsprechend 40 hexadezimalen Zeichen in der Ausgabe.
- SHA-2 *Secure Hash Algorithm* mit Längen von 256, 384 und 512 Bit [RFC 4634].

Message Authentication Check – MAC

Mit *Checksummen* und Hashsummen kann die technische Integrität der übertragenen Nachrichten gewährleistet werden. Häufig ist aber zudem gefordert, einen *Echtheitsnachweis* für den Sender der Datenquelle zu erbringen. Der Zielrechner soll in die Lage versetzt werden, herauszufinden, ob das betreffende IP-Paket wirklich von der (bekannten) Quelle stammt. Für diesen Zweck können beide Kommunikationspartner einen gemeinsamen Schlüssel zur Authentisierung der Datenpakete vereinbaren.

Keyed Hash

Fügt man diesen Schlüssel zum Datenpaket hinzu und berechnet anschließend den Hashwert (*Keyed Hashwert*), ermöglicht dies, dem Empfänger sowohl die Datenintegrität der empfangenen Pakete als auch dessen Herkunft zu überprüfen. D.h. es kann festgestellt werden, ob das IP-Paket unterwegs manipuliert wurde und ob das IP-Paket auch aus der vorgeblichen Quelle stammt. Die Authentisierung der Datenquelle und die Prüfung der Datenintegrität beim IP-Paket können als Prüfung der Echtheit dieses Pakets angesehen werden.

MAC

Mittels der Hashfunktion wird eine Prüfsumme `MAC` (*Message Authentication Code*) berechnet, mit der ein zu sendendes IP-Paket bzw. die gesamte Nachricht signiert werden kann. Somit ist diese Prüfsequenz mit der digitalen Signatur vergleichbar.

HMAC

Eine besondere Form der Keyed-Hashfunktion wird als HMAC-Funktion (*Hash based Message Authentication Code*) bezeichnet. Sie kann mit jeder beliebigen Einweg-Hashfunktion (z.B. MD5 als auch SHA) benutzt werden. Die HMAC-Funktion auf der Basis der Einweg-Hashfunktion `H` ist folgendermaßen definiert [RFC 2104]:

$$Y = HMAC(k, X) = H(k \otimes opad, H(k \otimes ipad, X))$$

Erläuterung: k: gemeinsamer und geheimer Schlüssel, X: die zu übertragenden Daten, Y: Prüfsumme als Signatur der Daten; $\otimes$: Operation `Bitwise_Exclusive_OR`, *ipad*: Sequence von B Byte '`0x36`', *opad*: Sequence von B Byte '`0x5c`'.

Zur Berechnung der Hashfunktion `H` werden zunächst die Daten in Blöcke mit der Länge von B Byte aufgeteilt ($B = 64$ und ggf. durch die Padding-Byte `0x00` aufgerundet). Anschließend erfolgt die Berechnung des Funktionswerts iterativ.

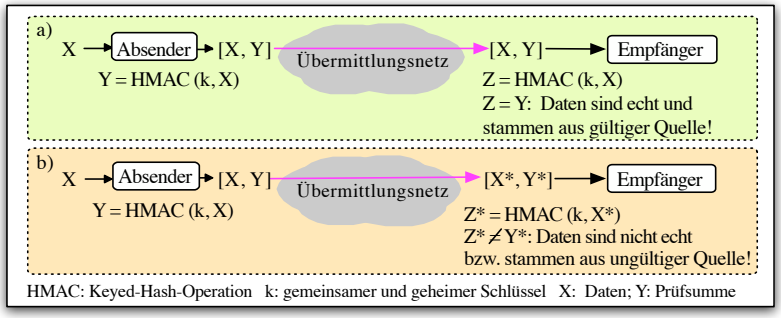


Abb. 1.6-3: Authentisierung der Datenquelle und Überprüfung der Datenintegrität
a) Ergebnis ist positiv, b) Ergebnis ist negativ

Beispiel: Abbildung Abb. 1.6-3 erläutert das Prinzip, wie die HMAC-Funktion genutzt werden kann, um die Echtheit und die Herkunft von Daten zu überprüfen, bei der beide Kommunikationspartner zunächst ein *Shared Secret* `K` besitzen. Wie ersichtlich, erstellt der Absender (Quellrechner) zusätzlich zur eigentlichen Nachricht `X` eine Prüfsequenz `Y`

mittels der HMAC-Funktion, in die die Nachricht `X` und das Shared Secret `K` einfließt und fügt diese der Nachricht bei. Der Empfänger der Nachricht entnimmt nun die Nachricht `X` und berechnet wiederum mittels des *Shared Secrets* `k` auf die gleiche Weise den Hashwert `Z`. Sind `Y` und `Z` gleich, [Abb. 1.6-3a] ist sowohl die Herkunft der Nachricht als auch ihre Unverfälschtheit sichergestellt. Ist der ermittelte Wert `Z*` nicht identisch mit dem Wert `Y*` für die Nachricht `X*`, wurde diese entweder bei der Übertragung verändert, oder aber der Absender besitzt nicht das gleiche Shared Secret [Abb. 1.6-3b].

HMAC-Funktionen

Üblicherweise stehen folgende HMAC-Funktionen zur Verfügung:

- *HMAC-MD5*:
  Sie generiert aus Daten variabler Länge eine Prüfsequenz mit der Länge von 128 Bit. Man verwendet hier den *Message Digest Algorithm 5* (MD5) als Hashfunktion `H`.
- *HMAC-SHA-1*:
  Sie generiert aus den Daten variabler Länge eine Prüfsequenz mit der Länge von 160 Bit. Hierbei findet SHA (*Secure Hash Algorithm*) als Hashfunktion `H` Verwendung.
- *RIPEMD-160*:
  RIPEMD (*RIPE Message Digest*) wurde für das RIPE-Projekt (*RACE Integrity Primitives Evaluation*) der EU entwickelt und 1996 erstmals veröffentlicht. RIPEMD-160 ist eine Hashfunktion, die eine Prüfsequenz mit der Länge von 160 Bit erzeugt. Es existieren auch die 128, 256 und 320 Bit Versionen von RIPEMD, die man entsprechend als RIPEMD-128, RIPEMD-256 bzw. RIPEMD-320 bezeichnet.

1.6.3 Grundzüge der symmetrische Verschlüsselung

Verschlüsselung ist über die letzten Jahrtausende immer eine Anforderung der Kriegsführung gewesen. Bereits seit der Antike sind *Chiffren* (Verschlüsselungsverfahren) bekannt. Zu den bekanntesten zählen die Verschiebeschiffren, von denen verbrieft ist, dass diese schon von Julius Caesar (100 bis 44 v.d.Z.) eingesetzt wurde. Historisch von Bedeutung sind beispielsweise

- Verschiebechiffren,
- multiplikative Chiffren sowie
- affine bzw. Tauschchiffren

die als *text-alfabetische* Chiffren bekannt sind und allesamt den Nachteil besitzen, dass mittels Analyse der statistischen Häufigkeiten der Buchstaben leicht festgestellt werden kann, aus welcher Sprache sie entstammen.

Kerckhoff'sches Prinzip (1883)

Später wurden *CodeBooks*, also Kodierbücher entwickelt, die eine Übersetzung der Texte ermöglichen. Die *Enigma*-Maschine stellte den Versuch dar, die Verschlüsselung zu automatisieren und zugleich einer zentralen Anforderung der modernen Kryptographie zu entsprechen:

> Bei kryptographisch 'sicheren' Verschlüsselungsmethoden darf die Sicherheit bzw. der Schutz der verschlüsselt übertragenen Nachricht nicht vom *Verschlüsselungsalgorithmus* abhängen, sondern ausschließlich von der Geheimhaltung des *Schlüssels* sowie der *Schlüssellänge*.

Das 'Brechen' des Enigma-Codes, durch polnische Informatiker aber auch vor allen durch den Briten *Alan Turing*[4], einem der Grundväter der Informatik, fußt zum Grossteil auf der Missachtung eines der folgenden Grundprinzipien der Verschlüsselung:

- **Diffusion**: Die Eingabewerte zur Verschlüsselung sollen auf unterschiedliche Chiffrebits verteilt werden, sodass kein Zusammenhang zwischen einem Chiffretextteils und dem Schlüssel gebildet werden kann.
- **Konfusion**: Die statistischen Eigenschaften des Chiffretextes sollen denen einer Zufallsfolge möglichst gleichen; unabhängig von den Eigenschaften der Eingabewerte.

Diese Aussagen gelten für *Blockverschlüsselungsverfahren*, die auch bereits von der Enigma genutzt wurde.

Überblick über die symmetrischen Verschlüsselungsverfahren

Abb. 1.6-4 gibt eine Überblick über die aktuellen Verfahren der symmetrischen Verschlüsselung, die wir generell einteilen in Verfahren der

- *Blockverschlüsselung* und
- *Datenstromverschlüsselung*.

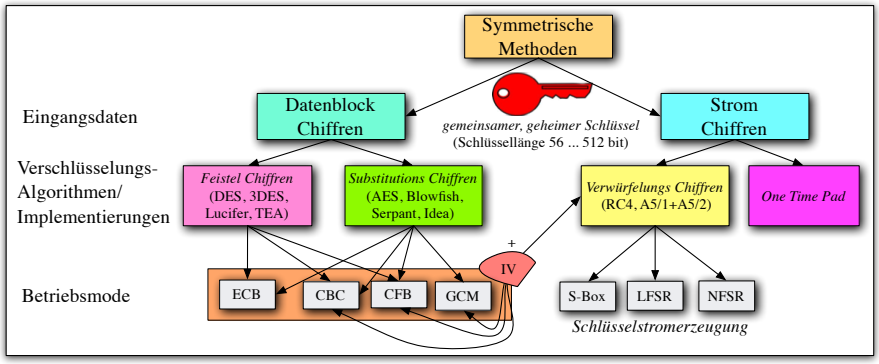


Abb. 1.6-4: Überblick über die symmetrischen Verschlüsselungsverfahren
ECB: Encoding Code Book, CBC: Cipher Block Chaining, CFB: Cipher Feedback Mode, GCM: Galois Counter Mode, LFSR: Linear Feedback Shift Register, NFSR: None-linear Feedback Shift Register, IV: Initialisierungsvektor

Strom-Chiffren, RC4

Strom-Chiffren werden eingesetzt für Nachrichten/Datenblöcke mit unbekannter Länge, z.B. einem Telefongespräch in einem digitales Netz. Sie finden daher u.a. bei den Mobilfunk-Netzen GSM und UMTS Verwendung (mit den Verfahren A5/1 und A5/2). Im Bereich des Internet ist nahezu ausschließlich die Variante (A)RC4 im Einsatz.

Initialisierungsvektor

Bei allen Strom-Chiffren muss neben dem initialen Schlüssel, der typischerweise mit 56 Bit relativ kurz ist, deterministisch neue Schlüssel generiert werden. Dies muss unabhängig von den Eingangsdaten erfolgen und geschieht entweder per Software in *S-Boxen* (Substitutions-Boxen) oder in Hardware mittels *Linear-* bzw. *None-linear Feedback Shift Register*. In allen Fällen bedarf es für die erste Sequenz der Daten einen *Initialisierungsvektor* (IV).

OTP

Eine besondere Strom-Chiffre stellt das *One Time Pad* dar. Hier ist der geheime Schlüssel gleich lang wie die Nachricht selbst und darf darüber hinaus nicht wieder verwendet werden. Unter diesen Umständen lässt sich beweisen, dass der resultierende Chiffretext nicht algorithmisch entschlüsselt werden kann: Der Code ist nicht 'knackbar'.

Datenblock-Chiffren

Bei *Block-Chiffren* muss vor der Block-Generierung die Gesamtlänge bekannt sein und der letzte Block auf die Länge des Schlüssels mittels eines *Padding* aufgefüllt werden. Die eigentliche Verschlüsselung erfolgt hier über *Substitutions-Boxen*, d.h. Verwürfelung. Neben dem alten DES-Mechanismus mit effektiven Schlüssellängen von 56 Bit, wird heute das AES-Verfahren eingesetzt, das unterschiedlich lange Schlüssel ermöglicht.

Diffusion, Verschlüsselung, Konfusion

Alle Verfahren sind so gewählt, dass

- die Verwürfelung (*Diffusion*) der Eingangsdaten,
- die eigentliche *Verschlüsselung* der so erzeugten Datensequenz und
- eine abschließende *Konfusion* der Ausgangsdaten

gewährleistet wird. Aus dem erzeugten Chiffretext kann also unter keinen Umständen mehr auf den den ursprünglichen Inhalt geschlossen werden. Weder *Known* noch *Chosen Plaintext* Attacken sind daher von Erfolg gekrönt.

Betriebsmode

Der Chiffretext kann entweder wie erzeugt in die Datenpakte gepackt werden, oder aber die einzelnen Blöcke der Nachricht werden untereinander verschränkt. Hierbei kann von den Betriebsmodi *Cipher Block Chaining* (CBC), dem *Cipher Block Feedback* (CFB), bzw. vom *Galois Counter Mode* (GCM) Gebrauch gemacht werden. Vergleichbar den Strom-Chiffren, muss auch hier ein *Initialisierungsvektor* beiden Kommunikationspartnern zu Anfang bekannt sein.

Eine Frage des Schlüssels

Schlüssel = zufällige Bitequenz

Unter der Annahme, dass der Schlüssel eine weitgehend zufällige Bitsequenz[5] darstellt und daher nicht trivial geraten werden kann, ist die Schlüssellänge Garant für die Verschlüsselung selbst. Will man eine Verschlüsselung brechen, sind statistisch gesehen $2^{Schlüssellänge/2}$ Operationen notwendig, bis der 'richtige' Schlüssel durch Probieren gefunden wird. Der unter diesen Umständen aus dem Chiffretext abgeleitete Klartext sollte anschließend 'Sinn' machen. Unter den Bedingungen der aktuellen Rechnerperformance, sind Schlüssellängen unter 100 Bit angreifbar.

Im praktischen Einsatz gibt es aber zwei Schlüssel:

- $A \xrightarrow{Schlüssel(A \to B)} B$
- $B \xrightarrow{Schlüssel(B \to A)} A$

Schlüssel-verteilungs-problem

Das Problem, das sich beim Einsatz der symmetrischen Verschlüsselung unmittelbar ergibt, ist das folgende:

> Wie kann ich den bzw. die Schlüssel meinem Kommunikationspartner zukommen lassen, sodass kein Anderer diesen mitlesen kann?

[4] vgl. Wikipedia http://de.wikipedia.org/wiki/Alan_Turing

[5] Eine solche Bit- bzw. Ziffernfolge wird im Folgenden auch als *Nonce* (oNly use once) bezeichnet.

Dieses Problem wurde durch die *asymmetrischen Verschlüsselungsverfahren* gelöst, was einen immensen Durchbruch in der Kryptographie darstellt und letztlich zum kommerziell nutzbaren Internet geführt hat.

1.6.4 Methoden der asymmetrischen Verschlüsselung

Die Entwicklung der *asymmetrischen Verschlüsselung* hat für die Kryptographie eine vergleichbare Bedeutung wie die *Kopernikanische Wende* für die Astronomie und stellt neben der *Digitalisierung* der Informationen und dem *Internet* als Kommunikationsplattform, ein Fundament des dritten Jahrtausends dar.

Asymmetrische Verschlüsselung

Erst Mitte der 70er Jahre von *Diffie-Hellman*[6] (DH) und *Rivest/Shamir/Adleman* (RSA) mit unterschiedlichen Ansätzen 'erfunden', wurden die Konsequenzen der asymmetrischen Verschlüsslung und des sichern 'Schüsseltauschs' sehr schnell erkannt. Anfang der 90er Jahre des letzten Jahrhunderts wurden entsprechende Algorithmen in Software umgesetzt und als Internetanwendungen und -protokolle (RFC 2437/RSA und 2631/DH) allgemein verfügbar gemacht.

Bedeutung von Public Key/Private Key

Bei den symmetrischen Verschlüsselungsverfahren besitzen beide Kommunikationspartner *einen gemeinsamen* Schlüssel, der sowohl zum *Entschlüsseln* als auch zum *Verschlüsseln* genutzt werden kann und zugleich zur *Authentisierung* des Kommunikationspartners dient.

Bei der *klassischen* asymmetrischen (RSA-)Verschlüsselung werden zwei Schlüssel gebraucht, die folgende Bedeutung besitzen:

- Der **public key** wird dem Kommunikationspartner bekannt gemacht und dient diesem zum *Verschlüsseln* von Nachrichten an den Besitzer des *private key*.
- Der **private key** dient zum *Entschlüsseln*, der mit dem *public key* verschlüsselten Nachrichten, aber auch
- zum *Signieren* eigener Nachrichten an potentielle Kommunikationspartner.

D.h. der *public key* erfüllt die Aufgabe der *Verschlüsselung*; der *private key* dient sowohl zur *Entschlüsselung* als auch zur *Authentisierung*.

Einordnung der asymmetrischen Verfahren

Abb. 1.6-5 liefert den Versuch des Überblicks über die gängigen asymmetrischen Verfahren und ihren Einsatz. Folgende wichtige Schritte wollen wir kurz diskutieren:

- **Schlüsselerzeugung**: Die RSA-Schlüssel, also der *private key* und der *public key* sind entweder *statisch* zur Verfügung gestellt oder werden *ephemeral*, also *während* des Verbindungsaufbaus erzeugt.
  Bei *Diffie-Hellman* besteht die öffentliche Schlüsselkomponente aus den *DH-Domain-Parameter* (DH), die in der Regel einmalig gebildet werden. Einen eigentlichen *private key* gibt es bei DH nicht, sondern die für den Ablauf des DH-Verfahrens notwendigen Zufallszahlen werden von beiden Kommunikationspartnern erst nach der Verbindungsaufnahme gebildet, sodass man auch von *Perfect Forward Secrecy* (PFS) spricht.

  PFS

[6] DH basiert auf *Merkle's puzzle* [Mer80], das als 'The Problem – Ready for Ralph' sogar als Hommage zum Gegenstand eines Popsongs wurde [Godley & Creme: *Ismism*, 1981].

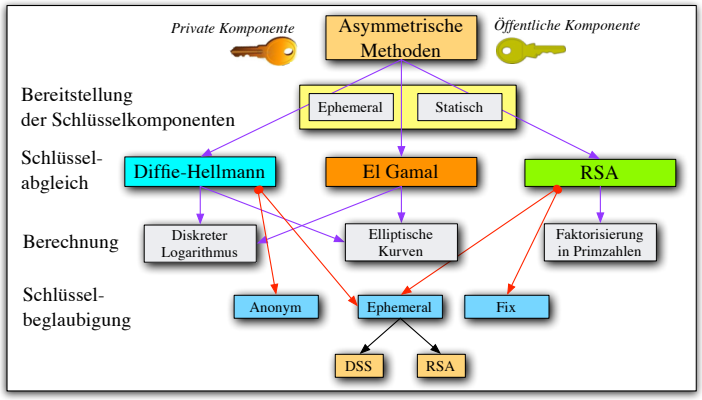


Abb. 1.6-5: Überblick über die asymmetrischen Verschlüsselungsverfahren
DSS: Digital Signature Standard, RSA: Rivest/Shamir/Adleman Algorithmus

- **Schlüsselabgleichsverfahren**: Beim Schlüsselabgleich findet entweder *Diffie-Hellman* oder aber *RSA* Verwendung. *El Gamal* – als Weiterführung von Diffie-Hellman – wird nur bei wenigen Protokollen eingesetzt.
- **Schlüsselabgleich**: Ausgehend von der vorliegenden Schlüsselkomponente, muss nun überprüft werden, ob der Kommunikationspartner das passende Gegenstück besitzt. Das mathematische Gerüst fundiert bei RSA auf der Faktorisierung in große Primzahlen bzw. bei Diffie-Hellman und El Gamal entweder auf dem Problem der Lösung von Problemen des *Diskreten Logarithmus* oder auf *Elliptischen Kurven*.
  Alle asymmetrischen Verfahren fußen auf der Annahme, dass die Berechnung des Gegenstücks, also z.B. des *private key* für jemanden, der lediglich den *public key* kennt, ausgesprochen schwierig ist. d.h. nicht in sinnvoller Zeit realisiert werden kann – außer bei der Erzeugung der Schlüsselkomponenten sind grobe Fehler gemacht worden. Bei RSA ist diese Aussage mit dem Auffinden großer Primzahlen verknüpft. Inwieweit große Primzahlen jedoch 'gefunden' werden können, hängt von den heutigen Algorithmen ab, sich aber morgen als unzureichend erweisen.
- **Schlüsselbeglaubigung**: In der Regel werden der *public key* bzw. die *DH-Domain-Parameter* während der Verbindungsaufnahme zusätzlich *beglaubigt*. Dies erfordert ein *Vertrauenssystem*, das in Form der *Public Key Infrastructure* (PKI) existiert, aber zunehmend kompromittiert ist.

Anonymous DH

Die Beglaubigung kann auch entfallen; hierbei sprechen wir dann von einem *anonymen Schlüsseltausch*, der aber die Gefahr von *Man-in-the-Middle* (MitM) Angriffen mit sich bringt.

Ablauf des RSA-Verfahrens

Beim RSA-Verfahren braucht nur der Server sowohl einen *public* als auch einen *private key* zu besitzen, die beide statisch sein können. In der Regel liegt der *public key* in einem *X.509-Zertifikat* vor [Abschnitt 6.2]. Die Sicherheit des RSA-Verfahrens basiert auf dem Problem der Zerlegung großer Zahlen in Primfaktoren; was eine sehr aufwändige Rechenoperation darstellt im Vergleich zur Multiplikation zweier Primzahlen. Der Hauptaufwand besteht in der Erzeugung qualifizierter Primzahlen.

Beispiel: Wesentliche Schritte beim RSA-Verfahren.

1. *Schlüsselerzeugung*:
   Zunächst werden zwei (große) Primzahlen p und q mit deutlich unterschiedlichen Längen benötigt und deren Produkt berechnet: $n = p{*}q$ sowie $\varphi(n) = (p-1){*}(q-1)$.

   RSA Schlüsselerzeugung

   - **Public key** $P(n, e)$:
     Anschließend wird eine Zahl e gesucht, die zu $(p-1) * (q-1)$ relativ prim ist.
     - e ist der *encryption Exponent*.
     - n der *Modulus*.

     Die Zahlen e und n bilden gemeinsam den öffentlichen Schlüssel $P(n, e)$.
   - **Private key** d:
     Es wird $d \equiv e^{-1} \bmod \varphi(n)$ über den *Euklid'schen Algorithmus* berechnet.
     Die Zahl d ist der *private Schlüssel* und es gilt: $d * e \equiv 1 \bmod \varphi(n)$
2. *Schlüsselabgleich*:
   Der Server teilt dem Client bei der Verbindungsaufnahme seinen *public key* für die nachfolgenden kryptographischen Operationen einfach mit. Mit dem *public key* des Servers kann der Client jetzt eine Nachricht verschlüsseln und an den Server versenden, welche dieser mit seinem privaten Schlüssel entschlüsselt. Die Nachricht kann z.B. einen geheimen, gemeinsamen Sitzungsschlüssel enthalten.

   RSA Schlüsselabgleich

Ablauf des DH-Verfahrens

Diffie-Hellman (kurz DH) ist ein Verfahren zur Verständigung auf einen *gemeinsamen Schlüssel*, wobei (derzeit) zwei unterschiedliche mathematische Verfahren genutzt werden können:

1. Der *Diskrete Logarithmus* (DL), wobei der Sachverhalt ausgenutzt wird, dass das Potenzieren schnell, die Umkehrfunktion – der Logarithmus – aber eine vergleichsweise viel schwierigere mathematische Operation ist.

   Diskreter Logarithmus
2. *Elliptische Kurven*, bei der geometrische Transformationen auf geeigneten Kurven zweiten Grades vorgenommen werden.

   Elliptische Kurven

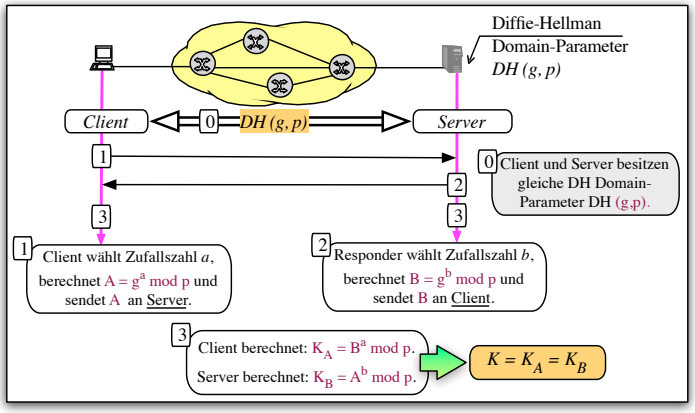


Abb. 1.6-6: Ablauf des Diffie-Hellman-Verfahrens zum Schlüsseltausch
DH (g,p): g ist Generator und p ist Primzahl der Gruppe

Ausgangspunkt des DH-Verfahrens ist zunächst die Bereitstellung der Diffie-Hellman *Domain-Parameter* $DH(g, p)$ [Abb. 1.6-6, Schritt 0]. Diese sind *per constructionem*

DH-Domain-Parameter

öffentlich und müssen sowohl Server als auch Client bekannt sein. Üblicherweise werden die Domain-Parameter beim Server erzeugt und dieser teilt diese beim Verbindungsaufbau dem Client mit.

DH-Gruppen

Statt die Diffie-Hellman-Parameter am Anfang selbst zu erzeugen, können diese einfach aus einer Tabelle entnommen werden [Tab. 1.6-1]. Beide Zahlen stellen die sogenannten *DH-Gruppenparameter* dar und sind zur Vereinfachung des DH-Verfahrens für vorgegebene Primzahlen p tabelliert. Statt die DH-Parameter zu berechnen, kann somit einfach auf einen bekannten Tabellenwert zurück gegriffen und dem Kommunikationspartner gesagt werden, welchen Eintrag er dabei nehmen soll, ohne die eigentlichen DH-Parameter zu übertragen.

Gruppe	Bitlänge	Primzahl bzw. Funktion + irreduzibles Polynom	Bezeichnung	RFC
1	768	$2^{768} - 2^{704} - 1 + 2^{64} * \{[2^{638} * \pi] + 149686\}$	768-Bit MODP	2409
2	1024	$2^{1024} - 2^{960} - 1 + 2^{64} * \{[2^{894} * \pi] + 129093\}$	1024-Bit MODP	2409
3	155	$y^2 + xy = x^3 + ax^2 + b; u^{155} + u^{62} + 1$	Oakley 3 (ECC)	2409
4	185	$y^2 + xy = x^3 + ax^2 + b; u^{185} + u^{69} + 1$	Oakley 4 (ECC)	2409
5	1536	$2^{1536} - 2^{14727} - 1 + 2^{64} * \{[2^{1406} * \pi] + 741804\}$	1536-Bit MODP	3526
14	2048	$2^{2048} - 2^{1984} - 1 + 2^{64} * \{[2^{1918} * \pi] + 124476\}$	2048-Bit MODP	3526
15	3072	$2^{3072} - 2^{3008} - 1 + 2^{64} * \{[2^{2942} * \pi] + 1690314\}$	3072-Bit MODP	3526
16	4069	$2^{4096} - 2^{4032} - 1 + 2^{64} * \{[2^{3966} * \pi] + 240904\}$	4096-Bit MODP	3526
17	6144	$2^{6144} - 2^{6080} - 1 + 2^{64} * \{[2^{6014} * \pi] + 929484\}$	6144-Bit MODP	3526
18	8192	$2^{8192} - 2^{8128} - 1 + 2^{64} * \{[2^{8062} * \pi] + 4743158\}$	8192-Bit MODP	3526
19	192		secp192r1 (ECC)	5114
21	224		secp224r1 (ECC)	5114
23	256		secp256r1 (ECC)	5114
24	384		secp384r1 (ECC)	5114
25	521		secp512r1 (ECC)	5114

Tab. 1.6-1: Einige Diffie-Hellman Gruppen (für TLS) und ihre Berechnungsgrundlage
MODP = Modular Exponentiation, ECC = Elliptic Curve Cryptography; [ ] ist die Gauß-Klammer, die eine ganze Zahl liefert; die Zahl π dient als 'Entropiequelle'

Perfect Forward Secrecy

Wie Abb. 1.6-6 zeigt, müssen bei Diffie-Hellmann beide Kommunikationspartner *pro Verbindung* Zufallszahlen erzeugen und sich gegenseitig mitteilen, um den gemeinsamen Sitzungsschlüssel zu bilden, sodass wir in diesem Zusammenhang von *Perfect Forward Secrecy* (PFS) sprechen: Selbst wenn dieser Schlüssel einmalig bekannt wurde, kann hierüber weder auf die in der Vergangenheit genutzten, noch in der Zukunft gebildeten Schlüssel geschlossen werden.

DH mit Diskretem Logarithmus

Beispiel: Ablauf des DH-Verfahrens beim Problem des *Diskreten Logarithmus*.

1. *Erzeugung der öffentlichen Schlüsselkomponenten* :
   - *Initialisierung* [Abb. 1.6-6, Schritt 0]:
     Zunächst wird eine (große) Primzahl p gesucht und eine Primitivwurzel g modulo p. p und g sind *nicht* geheim. Der Server übermittelt diese dem Client, bzw. teilt dem Client mit, welche tabellierten Werte er genutzt hat.
2. *Schlüsselabgleich* [Abb. 1.6-6, Schritte 1 und 2]:
   - *Verbindungsaufnahme*:
     Client und Server würfeln jeweils unabhängig voneinander Zufallszahlen a und b mit $a, b \in 1, \ldots, p-2$. a und b stellen die geheimen Schlüsselkomponenten dar.

- *Abgleich*:
  Der Client berechnet $A \equiv g^a \bmod p$ und der Server $B \equiv g^b \bmod p$ und beide senden sich A bzw. B gegenseitig zu.
- *Sitzungsschlüssel* K [Abb. 1.6-6, Schritt 3]:
  Nun berechnet der Client $K_A \equiv B^a \bmod p$ und der Server $K_B \equiv A^b \bmod p$:
  $K_A \equiv B^a \bmod p \equiv (g^b \bmod p)^a \bmod p \equiv g^{(ba)} \bmod p$, sowie
  $K_B \equiv A^b \bmod p \equiv (g^a \bmod p)^b \bmod p \equiv g^{(ab)} \bmod p$.
  Also gilt $K_A = K_B = K_{AB}$; dies ist der *gemeinsame Sitzungsschlüssel*.

Bei diesem sog. *Diffie-Hellman Key Exchange* (DHE) wird der gemeinsame Sitzungsschlüssel nicht im eigentlichen Sinn als Schlüssel genutzt, sondern vielmehr als *Sitzung-Token* interpretiert, das als Ausgangspunkt für weitere kryptographische Operationen dient, wie z.B. der Erzeugung der symmetrischen Schlüssel (nächster Abschnitt). DHE

Bei der Berechnung des *Diskreten Logarithmus* (DL) haben wir uns im Zahlenraum der natürlichen Zahlen aufgehalten. Als mathematische Operationen kamen nur das Potenzieren und das Berechnen des Teilerrests (*modulo*) in Anwendung. Ferner brauchten wir als Ausgangspunkt für die Berechnung eine *Primzahl*. Elliptische Kurven

Erläuterung: Wir stellen uns nun vor, den 'Zahlenstrahl' zu verbiegen und zwar so, dass die Zahlen entlang einer *elliptischen Kurve* liegen, wie in Abb. 1.6-7 dargestellt. Eine 'Zahl' auf dieser Kurve ist eigentlich eine Koordinate $C = (x_C, y_C)$. Wie man Abb. 1.6-7 entnehmen kann, lassen sich damit sowohl die positiven, als auch die negativen Zahlen abbilden, da gilt: $\underline{C} = -C = (x_C, C - y_C)$. Ist die elliptische Kurve geschickt gewählt, können auch weitere mathematische Operationen hierauf sehr einfach vorgenommen werden. Beispiel hierfür ist die Multiplikation eines Punktes C entlang der Kurve, die wir als $C + C + C + ... +$ schreiben.

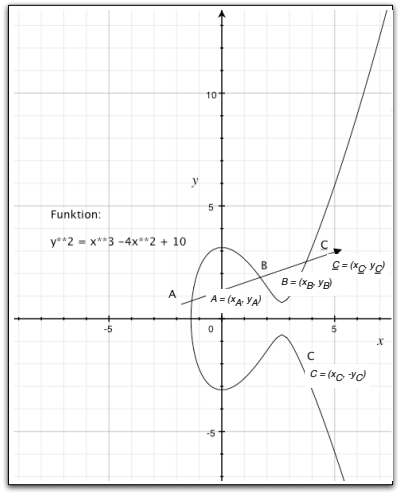


Abb. 1.6-7: Beispiel für eine elliptische Kurve unter Einbeziehung der Punkte A, B und $\underline{C}$

Beispiel: Diffie-Hellman beim Ablauf auf elliptischen Kurven. ECDH

1. *Schlüsselbereitstellung*:
   - *Elliptische Kurve*: Zunächst braucht man eine geeignete Elliptische Kurve, z.B. *Curve25519*, deren Koeffizienten beiden Partnern bekannt sind.
   - *Primitives Element*: Es wird ein 'primitives Element' c mit den Koordinaten $C = (x_C, y_C)$ festgelegt.

   Beide Teile stellen die öffentlichen *Domain-Parameter* dar.
2. *Schlüsselabgleich*:

- *Verbindungsaufnahme*:
  Der Client und der Server bestimmen jeweils getrennt je einen weiteren 'logischen' Punkt auf der Kurve, wobei a und b sind jeweils kleiner sind als C und beide geheim.
- *Abgleich*:
  Der Client berechnet $A = aC = (x_A, y_A)$ und der Server $B = bC = (x_B, y_B)$ und beide senden sich A bzw. B gegenseitig zu.
- *Sitzungsschlüssel* T:
  Nun berechnet der Client $aB = T_{AB}$ sowie der Server $bA = T_{AB}$.
  T_{AB} ist der *gemeinsame (geheime) Sitzungsschlüssel*.

Das DH-Verfahren mit Elliptischen Kurven unterscheidet sich somit lediglich vom ursprünglichen Verfahren durch die Wahl anderer DH-Domain-Parameter und dem Algorithmus. Allerdings sind die Schlüsselgrößen (bei gleichem Schwierigkeitsgrad) deutlich kleiner und das Verfahren kann mathematisch effizienter als beim Problem des DL realisiert werden.

Insbesondere können wir die Frage stellen, wie oft die Operation für C (also $C + C + ...$, z.B. d-mal) wiederholt werden muss, um den Wert D zu erreichen, der die Koordinate $D = (x_D, y_D)$ auf der Kurve besitzt. Dies ist das Problem des *Diskreten Logarithmus auf dieser elliptischen Kurve*: $D = dC$.

Nicht alle elliptischen Kurven sind geeignet für derartige Operationen, speziell die in Abb. 1.6-6 dargestellte Kurve nicht. Einige diese Kurven sind von der NIST standardisiert. Gerne benutzt wird die Montgomery-Kurve *Curve25519* [Ber14]:

$$y^2 = x^3 + Ax^2 + d\,;\ (A-2)/4 \text{ kleiner Integer } (A = 486662)$$

Der zweite Domain-Parameter ist die Primzahl $p = 2^{255} - 19$, von dem die Kurve ihren Namen hat.

DH-Domain-parameter

Statt die Diffie-Hellman-Parameter am Anfang selbst zu erzeugen, können diese einfach aus einer Tabelle entnommen werden. Beide Zahlen stellen die sogenannten *DH-Gruppenparameter* dar und sind zur Vereinfachung des DH-Verfahrens für unterschiedliche Werte von p tabelliert. Statt die DH-Parameter zu berechnen, kann somit einfach auf einen bekannten Tabellenwert zurück gegriffen und dem Kommunikationspartner gesagt werden, welcher dieser Werte er nehmen soll, ohne die eigentlichen DH-Parameter zu übertragen. Wie gezeigt, stellt Diffie-Hellman einen Ansatz für unterschiedliche mathematische Lösungsverfahren dar, während diese beim RSA-Algorithmus festgelegt sind.

1.6.5 Einsatz und Systematik hybrider Verschlüsselungsmethoden

Die asymmetrischen Verschlüsselungstechniken haben je nach mathematischem Verfahren eine Schlüssellänge von 128 bis zu 2048 Bit. Will man längere Dateninhalte verschlüsseln, müssen die zugrunde liegenden Operationen mehrfach durchgeführt werden, was rechentechnisch zu aufwendig wäre. Statt dessen werden *hybride Verfahren* eingesetzt:

Schlüsselmaterial

- Mittels der asymmetrischen Verfahren wird zunächst ein *Schlüsselmaterial* ausgetauscht bzw. sich darauf verständigt.

- Aus diesem Schlüsselmaterial werden die *symmetrischen Schlüssel* und ggf. der *Initialisierungsvektor* abgeleitet und anschließend
- die Daten entsprechend per *symmetrischer Verschlüsselung* übertragen und ggf. durch einen MAC (*Message Authentication Code*) abgesichert.

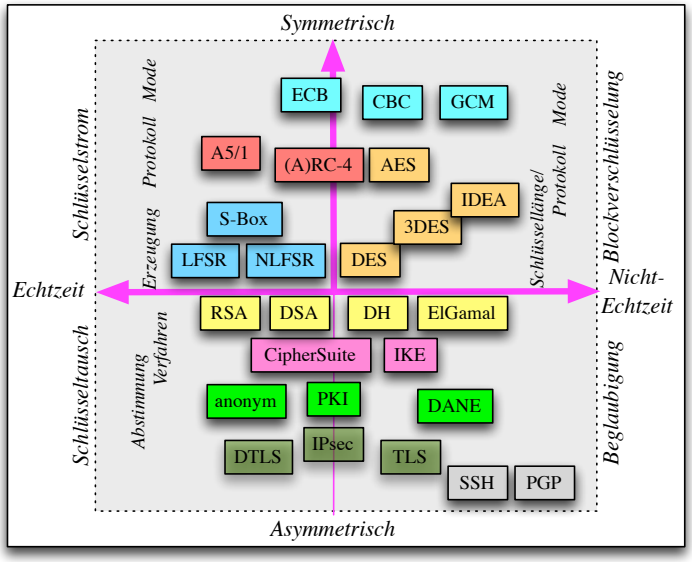


Abb. 1.6-8: Dimensionen des Einsatzes der Verschlüsselungs-, Schlüsselabgleich- und Authentisierungsverfahren
ECB: Electronic Code Book, CBC: Cipher Block Code, GCM: Galois Counter Mode, AES: Advanced Encryption Standard, DES: Data Encryption Standard, LFSR: Linear Feedback Shift Register, NFSR: None-linear Feedback Shift Register, RSA: Rives/Shamir/Adleman, DSA: Data Signature Algorithm, DH: Diffie-Hellman, IKE: Internet Key Exchange, PKI: Public Key Infrastructure, DANE: DNS-Based Authentication of Named Entities, DTLS: Datagram TLS, TLS: Transport Layer Security, SSH: Secure Shell, PGP: Pretty Good Privacy

Abb. 1.6-8 illustriert, dass für Echtzeit- und Nicht-Echtzeitkommunikation unterschiedliche symmetrische Verschlüsselungstechniken zum Einsatz kommen: Für *Echtzeitverschlüsselung* ist die Bereitstellung einer kontinuierliche Folge von Zufallswerten maßgeblich. Ausgehend von einem initialen Schlüssel und einem Initialisierungsvektor, werden diese Werte deterministisch entweder per Software (durch *Substitutions-Boxen*) oder per Hardware realisiert.

Blockverschlüsselung ist demgegenüber nur bei Nachrichten mit bekannter Länge möglich. Hierbei kann aber zusätzlich von der Verschränkung der einzelnen Datenblöcke Gebrauch gemacht werden.

Zum Aufbau der kryptographisch gesicherten Verbindung wird verlangt, dass sich bei Partner nicht nur über die Schlüssel, sondern auch über die Kryptoverfahren einigen. Diese Anforderung gilt für alle Kommunikationsprinzipien ganz allgemein, und die Aushandlung muss daher sowohl bei den Echtzeit- als auch den Nicht-Echtzeitprotokollen zu Beginn der Verbindungsaufnahme vorgenommen werden.