HANSER

Leseprobe

zu

Technik der IP-NETZE

von Anatol Badach, Erwin Hoffmann

E-Book-ISBN: 978-3-446-45511-5

Weitere Informationen und Bestellungen unter
https://www.hanser-fachbuch.de/buch/Technik+der+IP+Netze/9783446455115

sowie im Buchhandel

Inhaltsverzeichnis

Vorwort

Begriff: IP-Netze

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

Komplexität und Weiterentwicklung

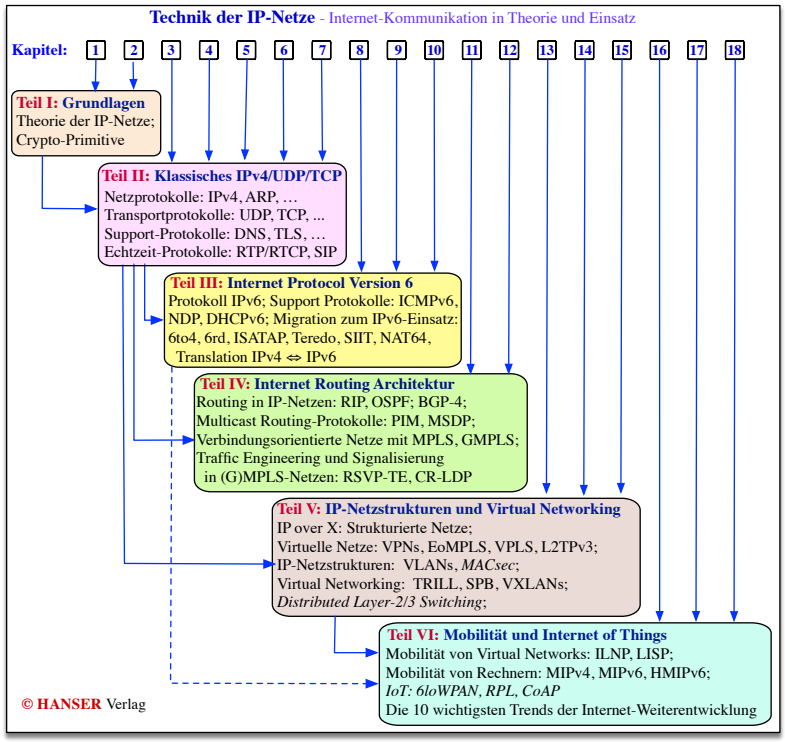
Ein IP-Netz – und insbesondere das Internet – besteht nicht nur aus mehreren Rechnern und IP/TCP dazwischen, sondern dahinter verbergen sich sehr komplexe Vorgänge. Das Internet stellt einen weltweiten Dienst zur Übermittlung nicht nur von Daten, sondern auch von audiovisuellen Informationen, also von Audio und Video, in Form von IP-Paketen dar. Vergleicht man diesen Dienst mit dem Briefdienst der Post, so entspricht ein IP-Paket einem Brief und die sog. IP-Adresse einer postalischen Adresse. Das massive Wachstum des Internet und die dabei entstehenden Probleme und neuen Anforderungen haben die Entwicklung sowohl eines neuen Internetprotokolls, des IPv6, als auch von Techniken MPLS und GMPLS für die Übermittlung der IP-Pakete über Hochgeschwindigkeitsnetze, insbesondere über optische Netze, vorangetrieben. Noch in der ersten Dekade dieses Jahrhunderts hat man von *Next Generation IP Networks* gesprochen und sie sind bereits 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 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, und vom Internet of Things, 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 18 Kapiteln, die auf fünf 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 benötigt, 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 verbindungslo-

se 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 führt den Leser in die Problematik der IT-Security ein: Netze sind *Vermittlungsnetze*, die Nachrichten *zuverlässig* und *unveränderlich* von *A* nach *B* transportieren sollen. Heute besteht der Bedarf aber auch darin, dass der Nachrichtentransport *vertraulich* erfolgt. Wie das zu bewerkstelligen ist, versucht Kapitel 2 zu vermitteln, indem vier Krypto-Primitive eingeführt und erklärt werden. Auch die Frage der Benutzer- und Rechnerauthentifizierung als 'Digitale Identitäten' mit und ohne *X.509 Zertifikate* wird beleuchtet. Kapitel 2

Kapitel 3 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 3

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 4 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 4

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 5 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 5

Kapitel 6 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 Kapitel 6

authentisierten Austausch von IP-Paketen genutzt werden? Welche Probleme verursacht NAT bei IPsec und wie können diese bewältigt werden? Zudem greifen wir das Problem Überprüfung von 'Digitalen Identitäten' unter Vorstellung der Lösungen EAP, RADIUS und LDAP wieder auf.

Kapitel 7 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 7 präsentiert diese, erläutert deren Aufgaben und geht u.a. auf folgende Fragestellungen ein: Wie kann der Datentransport zwischen Applikationen mittels TLS (1.3) 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 8 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 ermöglicht die automatische Konfiguration von Rechnern, sodass man sogar von *Plug&Play-Konfiguration* spricht. Kapitel 8 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 9 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 9 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 10 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 10 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 11 Router fungieren in IP-Netzen als Knoten, ermitteln optimale Übermittlungswege, die sog. *Routen*, für die empfangenen IP-Pakete und leiten sie weiter. Kapitel 11 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 12

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 12 stellt die Konzepte und Protokolle zum Aufbau der IP-Netze mit dem MPLS und dem GMPLS vor und geht u.a. auf die folgenden Probleme ein: Worin bestehen die Konzepte MPLS und GMPLS und welche Möglichkeiten entstehen durch deren Einsatz? Welche Services werden durch *Traffic Engineering* in IP-Netzen erbracht? Wie werden (G)MPLS-Netze aufgebaut und wie wird die IP-Kommunikation über sie realisiert? Wie erfolgt die IP-Kommunikation über optische Netze? Wie können Datenpfade über (G)MPLS-Netze dynamisch eingerichtet werden?

Kapitel 13

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 13 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 14

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 14 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 15

Der Gedanke der Virtuellen Netze kann aber von einer IP-basierten Infrastruktur gelöst werden und sehr effizient auf dem Layer 2 umgesetzt werden. Hiermit ergeben sich Konvergenzen von Layer 2 und Layer 3 Verfahren, die unter dem Stichwort 'Distributed Layer-2/3 Switching in Kapitel 15 vorgestellt werden, die bis zu auf Ethernet basierten virtuellen Netzen reicht, sowie so wie sie heute im Provider-Umfeld angeboten werden.

Kapitel 16

Um die Mobilität in IP-Netzen zu ermöglichen, wurden die Protokolle MIP (*Mobile IP*), MIPv6 (*Mobile IPv6*) und HMIPv6 (*Hierarchical MIPv6*) entwickelt. Kapitel 16 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 17

Ein in der Tat neues 'Kapitel' haben wir mit der Diskussion um das 'Internet of Things' IoT Kapitel 17 aufgeschlagen. Hier stellen wir die wesentlichen Konzepte de 'Internet der Dings', die Adaption von IPv6 für IoT – 6LoWAPN – die nun notwendigen Routingvefahren, sowie das Applikationsprotokoll CoAP vor.

Kapitel 18

Der Entstehung und der Zukunft des Internet sowie seine möglichen Anwendungen und Perspektiven ist Kapitel 18 gewidmet. Wir wissen ja: "Schwer zu sehen, in ständiger Bewegung die Zukunft ist." (Meister Joda). Trotzdem ist hier der Versuch gewagt, die beherrschenden Tendenzen in systematischer Weise zu betrachten und die aktuelle Diskussion in den Internet-Gremien zu referenzieren.

Verständnisfragen

Zu jedem einzelnen Kapitel gibt es ergänzend 'Verständnisfragen', durch die der geneigte Leser sein Wissen um zentrale Punkte der einzelnen Kapiteln überprüfen und ggf. vertiefen kann. Die Antworten auf diese Fragen finden sich Online auf der Webseite des Buches.

Vorwort zur vierten Auflage

Geschuldet der schnellen Entwicklung der Internet-Technologien und der guten Akzeptanz unserer 'Technik der IP-Netze' haben wir unsern 'siebenjährigen' Update-

zyklus (erste, zweite und dritte Auflage) etwas beschleunigt und stellen nun mit der vierten Auflage eine aktualisierte Version zur Verfügung. Neben obligatorischen Verbesserungen und Richtigstellungen in einigen Details, sind folgende Aspekte neu hinzugekommen:

- Die heutzutage als unentbehrliche IT-Security bei der Internetnutzung wird auf dem aktuellen Stand umfangreich diskutiert und in Kapitel 2 an zentraler Stelle untergebracht. Wir führen hier die vier zentralen *kryptographischen Primitiven* vor, die eine hervorgehobene Rolle einnehmen.
- Die Switching-Technologien für IP-Netze umfassen nun Layer-2 und Layer-3 Eigenschaften, deren Würdigung in Kapitel 16 zu finden ist.
- Ein wichtiger neuer Aspekt stellt das *Internet of Things* (IoT) dar. Dieses stellen wir in den Kapiteln 17 und 18 dar. Während zunächst Lösungen für Komponenten mit Ressourcen-beschränkter und im Besonderen geringer elektrischer Leistung (6LoWAPN) und den hieraus erwachsenden Konsequenzen im Vordergrund stehen, die für das Routing und für Applikationen, wie dem *Constrained Application Protocol* (CoAP) maßgeblich sind, spannen wir in Kapitel 18 dieses neue Umfeld mit seinen vielfältigen Facetten in Gänze auf und verweisen auf die aktuelle Literatur.

Die vierte Auflage wurde auch typographisch aufgefrischt:

Layout-Anpassungen und Beispiele

- Das Layout wurde über die vielen Kapitel vereinheitlicht und die Tabellen mit einem modernen Design versehen.
- Beispiele im Buch werden – wie in diesem Fall – mit einem links-seitigen Balken hervorgehoben.

Technik der IP-Netze – Homepage

Die Homepage des Buches ist unter https://www.fehcom.de/pub/tipn.html 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 auf Anfrage zur Verfügung.

Danksagung

Ein so umfangreiches Buch kann ohne Anregungen von außen und einen entsprechenden Erfahrungsaustausch nicht geschrieben werden.

Ein besonderer Dank gilt Herrn Dipl. math. Jürgen Müller (Darmstadt) und Herrn Dirk Müller (Koblenz) für die notwendige Sorgfalt, die dritte Auflage des Buchs intensiv durchzuarbeiten und uns Korrekturen vorzuschlagen. Ebenso möchten wir Herrn Jürgen Dubau für sein sorgfältiges Lektorat danken. Bei einem so langatmigen und umfangreichen Projekt ergeben sich immer Inkonsistenzen, die ohne das aufmerksame Zutun Dritter nicht ausgeschlossen werden können.

Die Autoren

Prof. Dr.-Ing. Anatol Badach
ist ehemaliger Professor im Fachbereich Angewandte Informatik der Hochschule Fulda. Die Schwerpunkte seiner Tätigkeit in Lehre und Forschung sind Netzwerktechnologien und -protokolle, *Internet of Things* und *Next Generation Networking*.
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.

https://www.researchgate.net/profile/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. Heute ist er Vertretungsprofessor an der Frankfurt University of Applied Sciences mit den Schwerpunkten Rechnernetze, Betriebssysteme, IT-Security sowie Verteilte Systeme.

http://www.fehcom.de

Dieses Buch möchten wir all jenen widmen, die dank ihrer technischen Schöpfungen zur Entstehung des Internet beigetragen haben, wozu im Besonderen Leonard Kleinrock und der Ende 2018 verstorbene Larry Roberts zählen, die die Grundlagen der Paketvermittlung geschaffen haben. Auch das Engagement vieler Freiwilliger sei gewürdigt, die das Internet weiterentwickeln und es aufrecht und offen erhalten.

Prof. Anatol Badach (Fulda)
Prof. Erwin Hoffmann (Höhn) – im November 2018

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 geht 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 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.

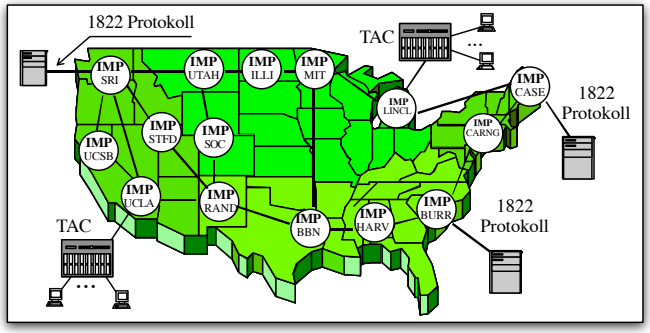


Abb. 1.1-1: Allgemeiner Aufbau von ARPANET – IMP dienen als Knoten
TAC: Terminal Access Controller, IMP: Internet Message Processor

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 und der Anwender, als auch 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 [GC02] die Idee,

- die Dokumente in einer speziellen Seitenbeschreibungssprache HTML (*Hypertext Markup Language*) aufzubereiten und diese untereinander durch Hyperlinks zu verbinden, wobei HTML
- die Dokumenten-Referenzen über einheitliche Adressen URL (*Uniform Resource Locator*) erfolgen sollten und URL
- die Verknüpfung über ein neues, einfaches Protokoll HTTP (*Hypertext Transfer Protocol*) abgewickelt werden sollte. HTTP

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. 1.1-2]. 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.

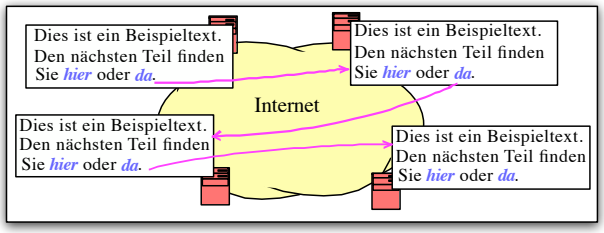


Abb. 1.1-2: Verknüpfung von Dokumenten auf unterschiedlichen Servern mittels Hyperlinks

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*. WWW als Web-Dienst

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. Hauptkomponenten des Webdienstes

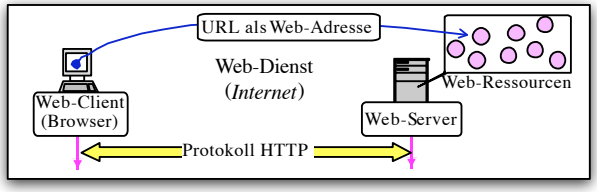


Abb. 1.1-3: Hauptkomponenten des Webdienstes – URL dient als Adresse

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-Adresse wird eine *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 dies 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 die Übertragung von Web-Inhalten zwischen Browsern und Webservern. Hierfür dient das HTTP (*Hypertext Transfer 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

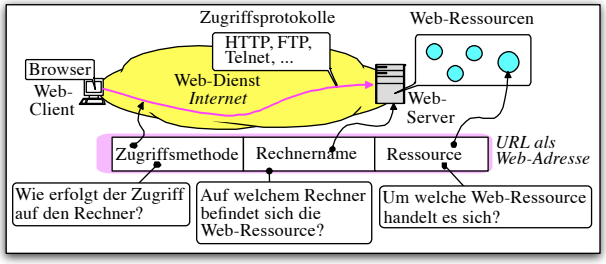


Abb. 1.1-4: Prinzip der Adressierung beim Web-Dienst

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.

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.

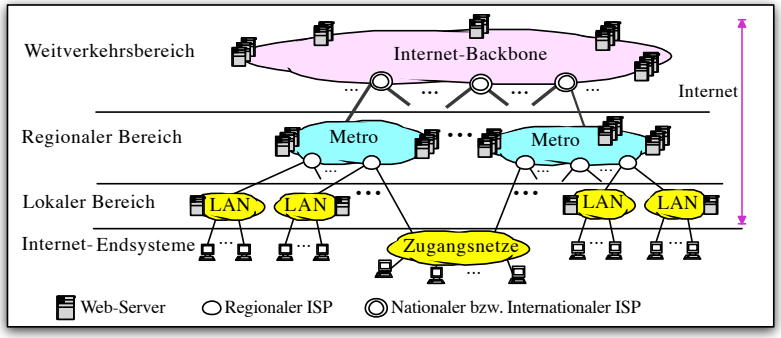


Abb. 1.1-5: Allgemeine Internet-Strukturierung – Aufbau als baumartige Struktur
ISP: Internet Service Provider; LAN: Local Area Network; Metro: Metro(politan)-Netz

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 Internetdienstanbietern, 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.

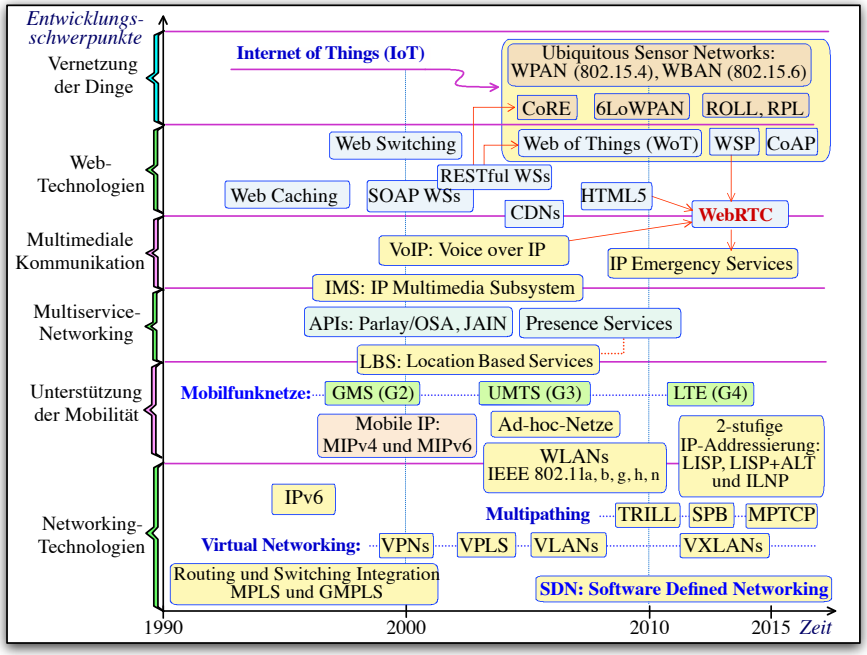


Abb. 1.1-6: Internet und IP-Netze; Meilensteile der bisherigen Entwicklung und Trends
CoRE: Constrained RESTful Environment, G: Generation, ROLL: Routing over Low power and Lossy neworks, RPL: Routing Protocol for Low power and Lossy networks, WPAN: Wireless Personal Area Network, WBAN: Wireless Body 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 quasi im 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 3.3.3]. 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 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 13.1]). 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. 15.2-2]) anbinden und auf diese Weise ein standortübergreifendes, verteiltes Ethernet einrichten; damit wurde VPLS (*Virtual Private LAN Service* [Abschnitt 13.2-10]) geboren. Etwa zur gleichen Zeit bildete man IP-Subnetze in lokalen Netzwerken als beliebige Gruppen von Rechnern und bezeichnete 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 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. Hierfür sind Konzepte nötig, um eine aus mehreren virtuellen 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 14.7 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 ist ein bedeutendes 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 14.7].

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 zu konvergieren begannen, 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 es, die Server der den Nicht-Netzanbieter, 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 Webbasierte *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, das 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 von *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. Vergleichbar, wie es ursprünglich bei Einführung des WWW vorgesehen war. 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, was besonders zeitkritischen Content, also beim Streaming-Media (Video, Internet-TV, -Spiele etc.) betrifft. 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 HTML5-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: *Web of Things* (WoT); siehe hierzu ITU-T Y.2063.

Sensornetze werden oft nach IEEE-Standards IEEE 802.15.4 (WPAN, *Wireless Personal Area Network*) und `IEEE 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.

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

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.

1.2.1 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.

Erste 'eiserne Regel'

Allen Kommunikationsprotokollen liegen zwei 'eiserne Regeln' zugrunde:

> 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.

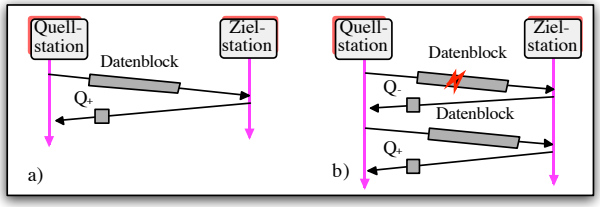


Abb. 1.2-1: Übermittlung eines Datenblocks: a) fehlerlos, b) fehlerhaft
Q+: positive Quittung, Q–: negative Quittung

Negative Quittung bei Störungen

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.

Verfälschung von Quittungen

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.

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 aufgrund 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.

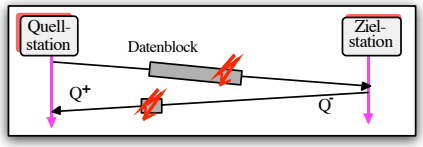


Abb. 1.2-2: Worst case einer Datenmittlung: Daten und Quittung werden verfälscht

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

> 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.

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. Geduldzeit

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 Nummerierung von Datenblöcken

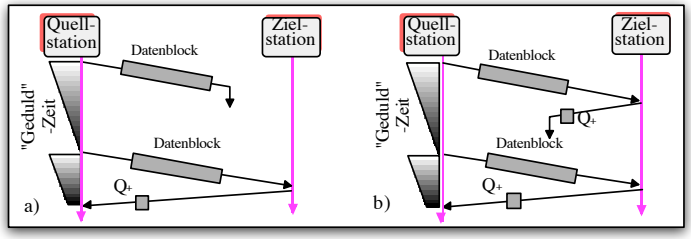


Abb. 1.2-3: Fehlerhafte Übermittlung: a) Datenblockverlust, b) Quittungsverlust

mit 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.

Flusskontrolle per Fenstermechanismus

> 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 verwen-

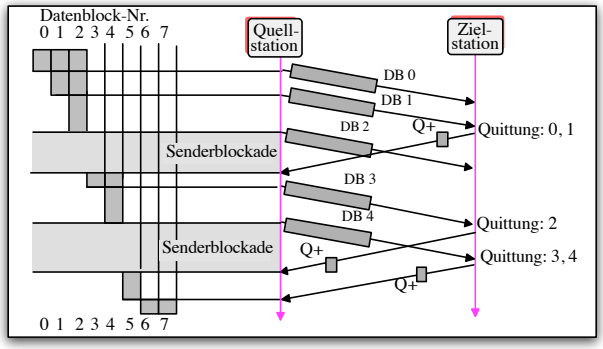


Abb. 1.2-4: Veranschaulichung der Flusskontrolle über den Fenstermechanismus

> den 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, 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.

Congestion Control

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:

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

Durchsatz

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

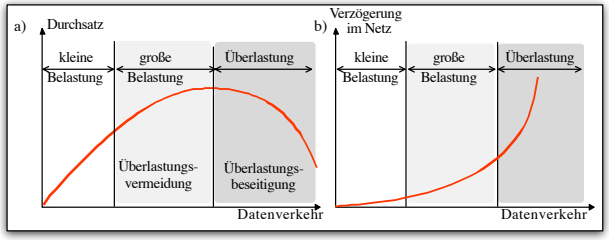


Abb. 1.2-5: Auswirkungen der Netzüberlastung: a) auf den Durchsatz, b) auf die Datenverweilzeit im Netz

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 4.5 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 (*Interna-*

tional 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.

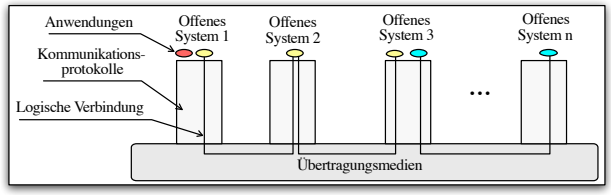


Abb. 1.3-1: Idee von OSI: Jedes System soll mit jedem anderen kommunizieren können

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-Schichtenmodell

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*).

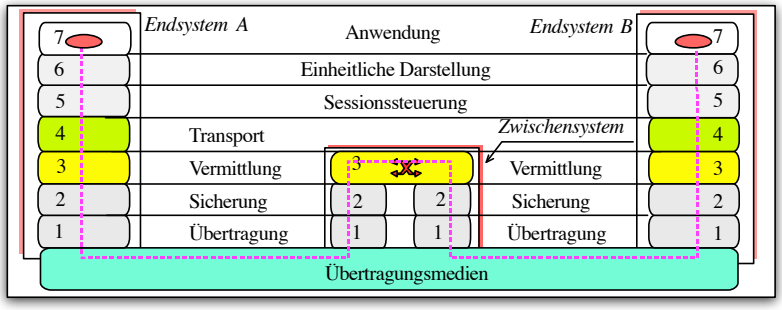


Abb. 1.3-2: OSI-Referenzmodell: Zerlegung der Kommunikationsaufgabe in 7 Schichten

Die einzelnen Schichten im OSI-Referenzmodell sind:

Funktionen der Schichten

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

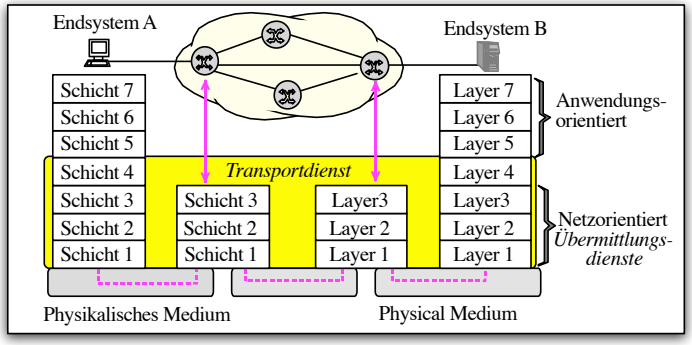


Abb. 1.3-3: Klassen von Aufgaben im OSI-Referenzmodell

Daten in Form von Frames zwischen zwei direkt verbundenen Stationen [Abb. 1.3-5]. Die ersten 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 – also 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. 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.

Verbindungslose vs. verbindungsorientierte Kommunikation

Verbindungs-orientiert, Verbindungslos

Protokollieren zwei Kommunikationsinstanzen auf der jeweiligen Schicht *Zustandsinformationen* über ihren Partner, und tauschen 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.

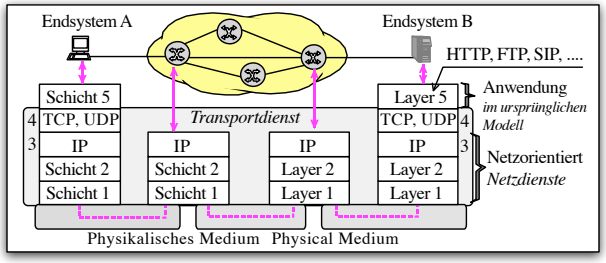


Abb. 1.3-4: Ursprüngliches Schichtenmodell der Protokollfamilie TCP/IP

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 (*Trans-*

mission Control Protocol) und UDP (*User Datagram Protocol*) eingesetzt. TCP ist ein verbindungsorientiertes; 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.

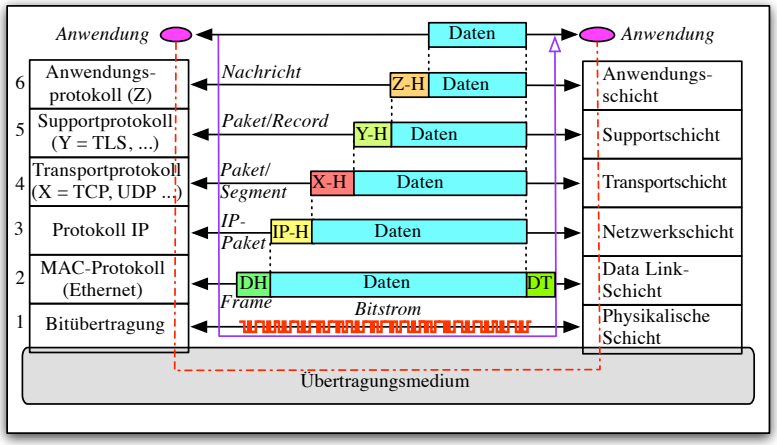


Abb. 1.3-5: Erweitertes Schichtenmodell der Protokollfamilie TCP/IP – Strukturen von zwischen den kommunizierenden Instanzen übermittelten Daten
DH: Data-Link Header, DT: Data-Link Trailer

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 dieses 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.

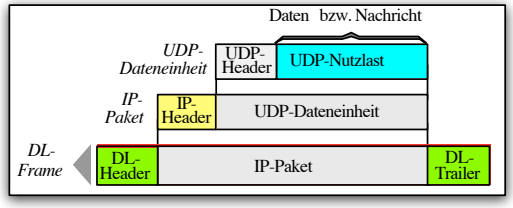


Abb. 1.4-1: Kapselung der Nutzlast beim UDP-Einsatz

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 13.2).

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.

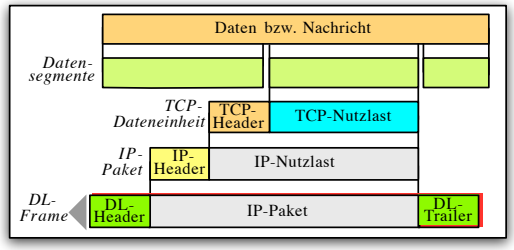


Abb. 1.4-2: Verkapselung der Nutzlast beim TCP-Einsatz

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.

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 Pro-

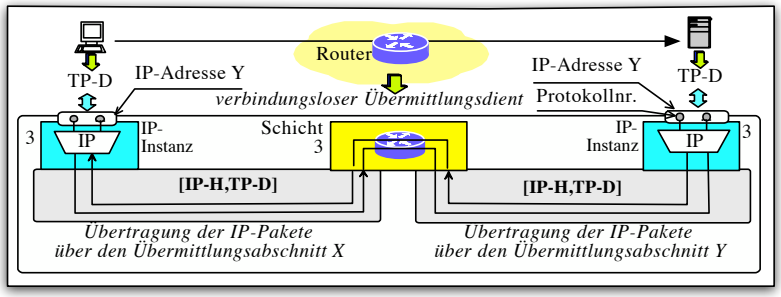


Abb. 1.4-3: Struktur der verbindungslosen Netzwerkschicht in IP-Netzen
IT-H: IP-Header, TP-D: Transportprotokolldateneinheit

tokoll der Schicht 4 wird daher von der IANA (*Internet Assigned Numbers Authority*) eine feste und weltweit eindeutige Nummer zugewiesen.

Verbindungsorientierte Netzwerkschicht

Verbindungs-orientierte 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 im '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*.

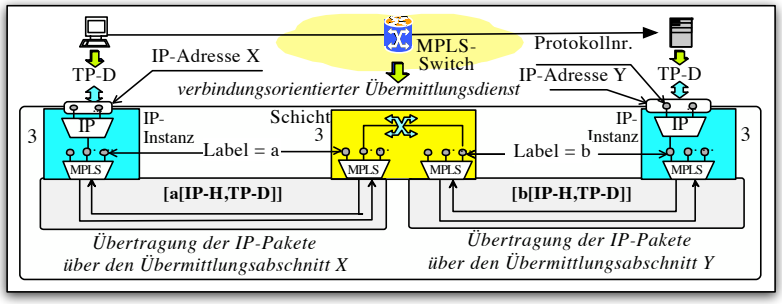


Abb. 1.4-4: Struktur der verbindungsorientierten Netzwerkschicht in IP-Netzen
IT-H: IP-Header, TP-D: Transportprotokolldateneinheit

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 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 4.3].

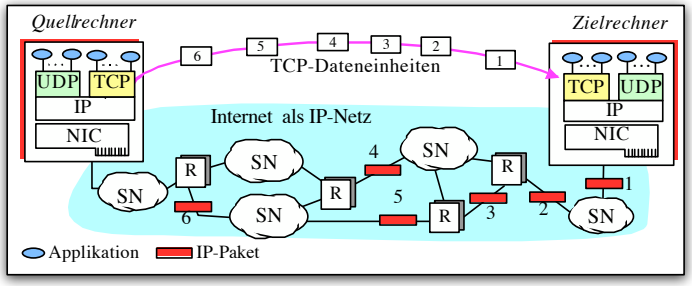


Abb. 1.4-5: Prinzip der Kommunikation im Internet – Datagramm-Prinzip
R: Router, SN: IP-Subnetz, NIC: Network Interface Card (Controller)

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.

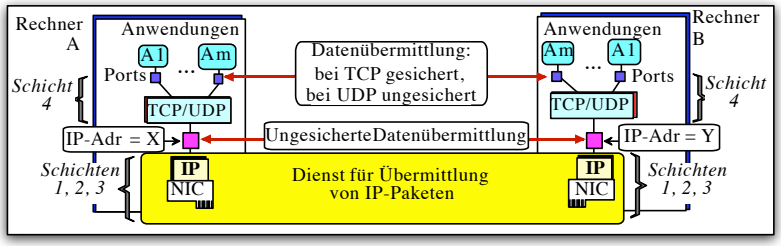


Abb. 1.4-6: Vereinfachte Struktur von Rechnern am IP-Netz
A: Applikation, Adr: Adresse, NIC: Network Interface Controller

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. Da-

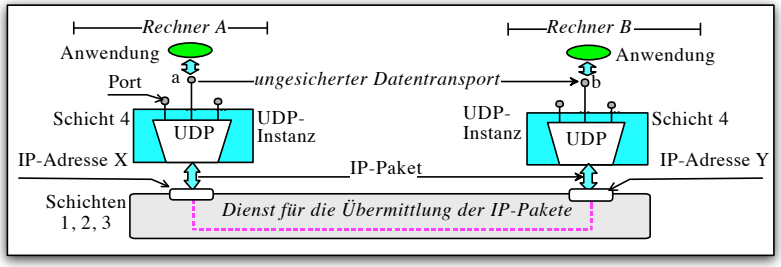


Abb. 1.4-7: Transportschicht mit UDP; ungesicherter Datentransport

mit können mehrere UDP-Anwendungen parallel auf den Dienst für die Übermittlung der IP-Pakete zugreifen.

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 empfangen. 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].

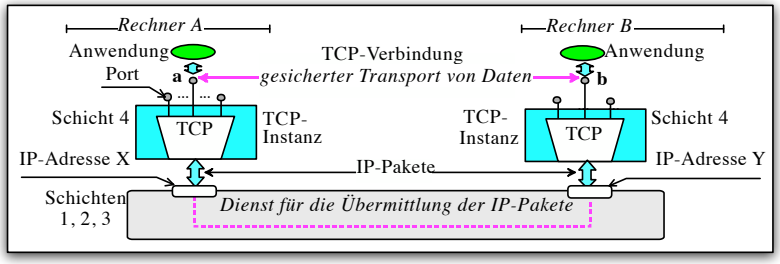


Abb. 1.4-8: Transportschicht mit TCP; gesicherter Datentransport

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.

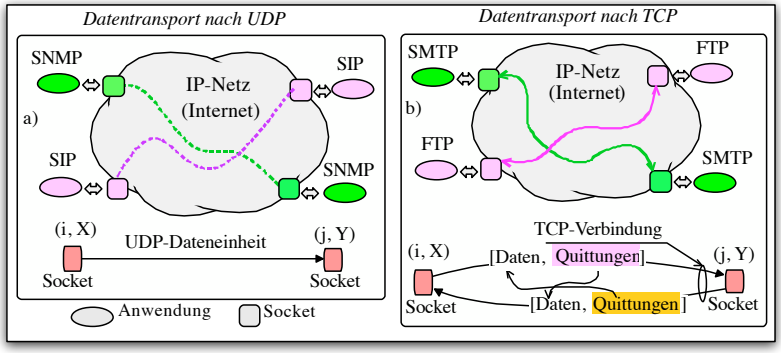


Abb. 1.4-9: Datentransport zwischen Anwendungen: a) beim UDP-Einsatz, b) beim TCP-Einsatz

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 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 Stra-

ße' über ein IP-Netz verstanden werden, über die ein gesicherter Datentransport erfolgt indem die empfangenen Daten quittiert werden [Abschnitt 4.3].

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.

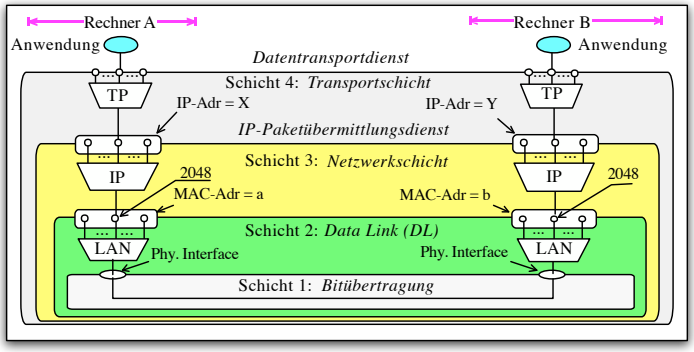


Abb. 1.4-10: Multiplexmodell der Protokollfamilie TCP/IP beim IP-Netz auf LAN-Basis
TP: UDP bzw. TCP

Hier soll u.a. gezeigt werden, dass alle Schichten von `1` bis `n-1` einen Übermittlungsdienst für die 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` 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 bereit. Eine IP-Adresse kann als Zugangspunkt zu diesem Dienst betrachtet werden, und über sie können mehrere Protokolle der

[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.

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.

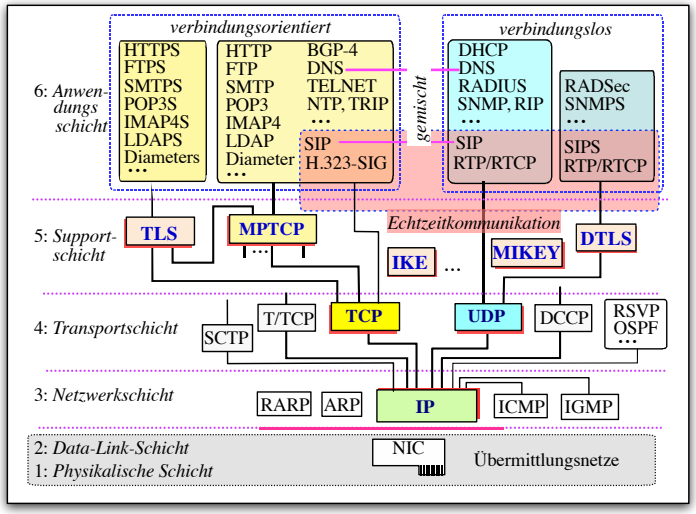


Abb. 1.5-1: Protokolle der Familie TCP/IPv4 im Schichtenmodell
NIC: Network Interface Card (Adapterkarte)

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.

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 3.6.1].
- **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 3.6.3].
- **ICMP**: *Internet Control Message Protocol* wird für die Übermittlung von Fehlermeldungen und anderen Kontrollangaben verwendet [Abschnitt 3.7].
- **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 3.8.2].

  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 4.3 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 4.2 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 4.6 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 12.5 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 11.3 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 6.3].
- **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 7.1], 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* fasst eine besondere Klasse von Anwendungen zusammen, deren Anforderungen nicht unmittelbar auf der Transportschicht umgesetzt werden können: Diese 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**, Version 2.0) [RFC 7826] realisiert wurden. Bei der Echtzeitkommunikation dient SIP (*Session Initiation Protocol*) als *Signalisierungsprotokoll* dazu, Verbindungen auf- und wieder 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.

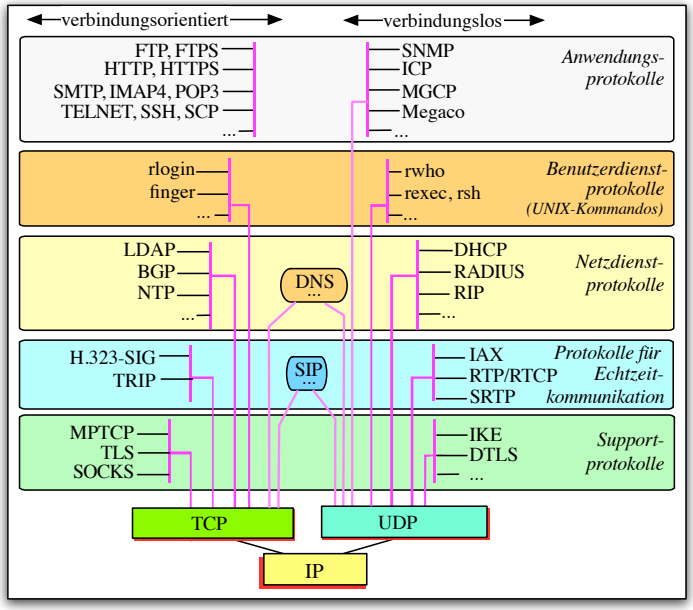


Abb. 1.5-2: Einordnung der Protokolle entsprechend ihrer Kommunikationsschicht

Verbindungs-orientierte Anwendungs-protokolle

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 Anwendungs-protokolle

Verbindungslose Anwendungsprotokolle sind u.a.:

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

- **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 11.4].
- **LDAP**: *Lightweight Directory Access Protocol* verwendet man bei der Realisierung verteilter Verzeichnisdienste und wird vor allem als Backend für die Benutzerauthentisierung genutzt [Abschnitt 15.3].

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 6.2 beschrieben.
- **RADIUS**: *Remote Dial-In User Service* wird in Abschnitt 15.2 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 zur Unterstützung der Echtzeitkommunikation sind u.a.:

Protokolle für Echtzeitkommunikation

- **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 IETF und Internet-Standards

IETF

Um die Weiterentwicklung des Internet und seine Anwendungen voranzutreiben, wurde die Organisation *Internet Engineering Task Force* (IETF) gegründet. Zu ihren Aufgaben gehört die Koordination sämtlicher Aktivitäten, die mit der technologischen Weiterentwicklung und der Standardisierung der Internetdienste und -protokolle zusammenhängen. Die IETF-Dokumente werden als RFC (*Request for Comments*) im Internet veröffentlicht.

RFC als Internet-Standards

Ein Schlüssel zur raschen Entwicklung des Internet und der IP-Netze ist vor allem der offene Zugang zu den als RFC im Internet veröffentlichten IETF-Dokumenten, die als *Internet-Standards* dienen. Außerdem kann jeder einen neuen RFC vorschlagen, wobei die Vorgehensweise RFC 5000 festlegt.

Verzeichnis der RFC

RFC reichen bis ins Jahr 1969 zum Vorläufer des Internet zurück. In Oktober 2018 liegen bereits über 8500 veröffentlichte RFC vor. Alle RFC sind auf mehreren Rechnern im Internet abgespeichert und kostenlos für jeden Nutzer verfügbar. Ein Verzeichnis aller RFC, die vom *RFC Editor* verwaltet wird, ist unter der Adresse `http://www.rfc-editor.org/rfcsearch.html` zu finden. Die Suche in dieser Datenbank kann durch die Angabe der Nummer des gesuchten RFC oder durch die Angabe eines Suchkriteriums (z.B. Name eines Protokolls wie IP, TCP, OSPF, ...) erfolgen oder alternativ über `http://www.rfc-editor.org/rfc-index2.html`.

Organisation der IETF

Der Erfolg des Internet ist teilweise der gut durchdachten Organisation der Zusammenarbeit zwischen der IETF und den anderen Institutionen zu verdanken. Welche Institutionen an der Entstehung von Internet-Standards beteiligt sind und wie sie zueinander stehen, zeigt Abb. 1.6-1.

IAB und RFC Editor

Die Entwicklung des Internet wird vom *Internet Architecture Board* (IAB) in Zusammenarbeit mit der *Internet Research Task Force* (IRTF) [`http://ietf.org`] koordiniert. Dem Vorsitzenden des IAB wurde der Titel *Internet Architect* verliehen. Außerdem wurde im IAB der Posten des RFC Editor eingerichtet, der jeden RFC prüfen und zur Veröffentlichung vorbereiten soll.

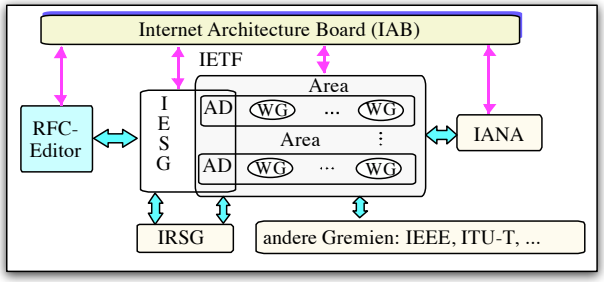


Abb. 1.6-1: Organisation der IETF und die Zusammenarbeit mit anderen Internet-Gremien
AD: Area Director, IANA: Internet Assigned Numbers Authority, IESG: Internet Engineering Steering Group, IRSG: Internet Research Steering Group, WG: Working Group

Area als Themenbereich

Da die Palette von Entwicklungen um das Internet und deren Anwendungsaspekte herum sehr breit ist, werden bei der IETF bestimmte Themenbereiche definiert. Ein Themenbereich wird als *Area* bezeichnet. In jeder Area wird ein *Area Director* (AD) benannt, der die Aktivitäten innerhalb der Area koordiniert. Es existieren u.a. folgende Areas: *Applications Area*, *Internet Area*, *Routing Area*, *Security Area*, *Transport Area*.

Working Groups

Für die Entwicklung von Standards zu den einzelnen Themen in jeder Area werden mehrere *Working Groups* (WGs) gebildet. Eine WG übernimmt die Verantwortung für die Entwicklung von Standards, die in der Regel ein Thema (z.B. ein Protokoll oder eine Applikation) betreffen. Eine Auflistung von WGs findet man unter: `http://www.ietf.org/html.charters/wg-dir.html`

Hervorzuheben sind u.a. folgende aktive WGs (Stand Oktober 2013):

- Internet Area: dhc (*Dynamic Host Configuration*), 6man (*IPv6 Maintenane 6*), dmm (*Distributed Mobility Management*), mip4 (*Mobility for IPv4*)
- Routing Area: ccamp (*Common Control and Measurement Plane*), mpls (*Multiprotocol Label Switching*), ospf (*Open Shortest Path First*), pim (*Protocol Independent Multicast*), pwe3 (*Pseudowire Emulation Edge to Edge*), rtgwg (*Routing Area Working Group*), sidr (*Secure Inter-Domain Routing*)
- Transport Area: aqm (*Active Queue Management and Packet Scheduling*), ippm (*IP Performance Metrics*), mptcp (*Multipath TCP*), tcpm (*TCP Maintenance and Minor Extensions*), tsvwg (*Transport Area Working Group*)
- Real-Time Applications and Infrastructure Area: avtcore (*Audio/Video Transport COre Maintenance*), rtcweb (*Real-Time Communication in WEB-browsers*), sipcore (*Session Initiation Protocol Core*)

Um die Entwicklung der Internet-Standards zu verfolgen und eine gut strukturierte Übersicht über die Internet-Drafts zu erhalten, verweisen wir auf die Seite `http://www.potaroo.net/ietf/html/xids-all.html`. Für einen schnellen und übersichtlichen Zugriff auf alle IETF Working-Groups und deren Dokumente, ist die Adresse `http://www.in2eps.com/x0/tk-ietf-wg-lists.html` zu empfehlen.

IESG

Für die technische Verwaltung von IETF-Aktivitäten ist die *Internet Engineering Steering Group* IESG verantwortlich. Zur IESG gehören die Direktoren der einzelnen

Areas, die ADs. Der Entwurf jedes Internet-Standards, den man als *Internet Draft* bezeichnet, wird vor seiner Spezifikation als RFC innerhalb der IESG diskutiert. Ein Internet Draft wird nur mit der Zustimmung der IESG als Internet-Standard veröffentlicht. Die IESG arbeitet mit dem RFC-Editor zusammen, der für die Veröffentlichung der RFC zuständig ist.

IANA

Eine besondere Rolle unter den Internet-Gremien spielt die *Internet Assigned Numbers Authority* IANA. Sie dient als zentrale Stelle für die Registrierung von Internet-Adressen, -Namen, Protokollnummern und anderen Parametern, die weltweit eindeutig sein müssen [http://www.iana.org/numbers.html].

1.7 Schlussbemerkungen

In diesem Kapitel wurden in komprimierter Form vor allem die notwendigen Grundlagen dargestellt, die für die Beschreibung von Ideen, Kommunikationsprotokollen und System- und Sicherheitslösungen für IP-Netze in den weiteren Kapiteln hilfreich sind. Abschließend sei noch auf Folgendes hingewiesen:

Web-Technologien

- Das Internet verdankt die heutige Popularität hauptsächlich dem *Web* mit dem Protokoll HTTP. Der Webdienst bedeutet heute nicht nur TCP/IP und HTTP. Für seine effiziente Realisierung werden verschiedene Technologien eingesetzt, sodass man von *Webtechnologien* spricht. Zu ihnen gehören u.a. die Konzepte und Protokolle für *Web-Switching*, *Web-Caching*, *Web-Sockets* sowie *Content Delivery Networks* für verschiedene Arten von Web-Services. Für eine vertiefte Diskussion sei auf [BRS03] verwiesen.

CDN-Idee

- Seit einiger Zeit werden verschiedene Arten des zeitkritischen Contents, u.a. in Form von sog. *Streaming-Medien*, über das Internet zum Abruf angeboten. Um alle Formen der Streaming-Medien weltweit so anbieten zu können, dass sie bei den Internetnutzern überall und immer in guter Qualität ankommen, werden sog. *Content Delivery Networks* (CDNs) auf Basis des Internet eingerichtet.

  Abb. 1.7-1 illustriert das allgemeine CDN-Konzept mit den drei Schritten:

  - `A`: Content Distribution auf zahlreiche Replica-Server
  - `B`: Ermittlung der IP-Adresse eines Replica-Servers
  - `C`: Content-Abruf vom bestgeeigneten Replica-Server

  Die Aufgabe des CDN besteht in der Vervielfachung des bei den Content-Anbietern auf Ursprungs-Servern (*Origin Servers*) gespeicherten Contents, also in der Erzeugung von dessen Replikationen (*Replicas*), und in deren Verteilung mithilfe eines Content Distribution Systems auf eine Vielzahl von im Internet verstreuten und möglichst in der Nähe zu Nutzern installierten Servern. Da auf diesen Servern die Content-Replikationen für den Abruf vorgehalten werden, bezeichnet man diese Server als Replica-Server.

  Jedes CDN verhält sich einerseits als Replikator (Vervielfacher) von Content-Masterkopien und als deren Verteiler auf mehrere Replica-Server, andererseits muss jedes CDN auch dafür sorgen, dass der Content als Internetressource unter einer Adresse – d.h. unter einem auf den Ursprungs-Server bei einem

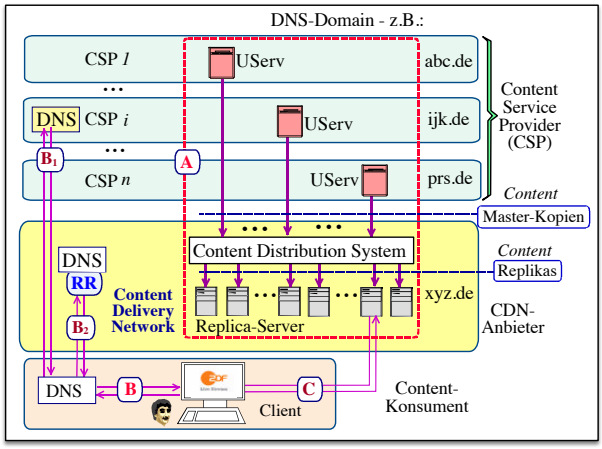


Abb. 1.7-1: Allgemeine Struktur eines CDN und dessen Basisfunktionen: A) Content Distribution auf zahlreiche Replica-Server, B) Ermittlung der IP-Adresse des Servers, C) Content-Abruf vom bestgeeigneten Replica-Server
DNS: Domain Name System, RR: Request Router, UServ: Ursprungs-Server

Content-Anbieter verweisenden URL (*Uniform Resource Locator*) – nicht vom Ursprungs-Server abgerufen wird, sondern von einem nach bestimmten Kriterien ausgewählten Replica-Server beim CDN-Anbieter. Um solche Umleitungen vom Ursprungs-Server auf einen entsprechenden Replica-Server zu ermöglichen, wird in CDNs das sog. *Request Routing* realisiert. Weitere Details finden sich in [https://www.researchgate.net/publication/282008087_CDN_-_Content_Delivery_Network].

- Ein wichtiger Trend bei der Weiterentwicklung des Internet ist die Unterstützung der Echtzeitkommunikation und hierfür waren verschiedene Konzepte und Protokolle zur Übermittlung von Audio und Video über IP-Netze notwendig. Bei der im Internet zunehmenden Echtzeitkommunikation handelt es sich um audiovisuelle Kommunikation, und man spricht in diesem Zusammenhang auch von VoIP (*Voice over IP*) bzw. auch von MMoIP (*Multi-Media over IP*). Für den Transport audiovisueller 'Daten' in IP-Netzen wurde RTP (*Real-time Transport Protocol*) entwickelt [Abschnitt 7.3]. Die audiovisuelle Kommunikation stellt eine Art Videotelefonie dar. Hierbei benötigt man ein sog. *Signalisierungsprotokoll*, um u.a. virtuelle Verbindungen, *Sessions*, auf- und abzubauen. SIP (*Session Initiation Protocol*) ist eine derartiges Protokoll [Abschnitt 7.4]. Zur Realisierung der audiovisuellen Kommunikation im Internet kann auch der Webdienst mit dem Protokoll HTTP eingesetzt werden, was unter dem Begriff WebRTC (*Web Real-Time Communication*) geführt wird. Bei WebRTC dienen Webbrowser zusätzlich als Soft-Videotelefone und sie werden entsprechend an Webserver virtuell angebunden, sodass die Webserver als quasi Vermittlungsknoten beim Auf- und Abbau von virtuellen Verbindungen zwischen Soft-Videotelefonen fungieren. Wegen Platzmangel wurde aber hier auf die Darstellung von WebRTC verzichtet; für weitere Informationen über WebRTC sei auf das Wissensportal [Bad14] verwiesen.

Echtzeit-kommunikation: VoIP, MMoIP

Programmierbare MAC-Schicht

- Zur Übertragung der IP-Pakete auf das Medium wird zusätzlich zur physikalischen Anbindung die Datensicherungsschicht [Abb. 1.3-2] benötigt. Im Zuge der TCP/IP-Einführung hatte diese zunächst an Bedeutung verloren, erfährt aber derzeit eine Renaissance bei den *Software Defined Networks* (SDN): *Virtuelle Interfaces* ermöglichen die Steuerung der Kommunikation bei Cloud-Services und bilden somit einen wichtigen Bestandteil des Sicherheitskonzepts, was wir in Kapitel 17 diskutieren.

Internet of Things – als funktionelle Erweiterung des Internet

- Die Integration verschiedener, in der Regel drahtloser Sensor-Aktor-Netze in das herkömmliche Internet führt zur Entstehung des *Internet of Things* (IoT) bzw. auf Deutsch des *Internet der Dinge*. Das IoT ist eine funktionelle Erweiterung des Internet mit dem Ziel, Alltagseinrichtungen (Geräte, Sensoren,...) unterschiedlicher Art und mit unterschiedlichen Fähigkeiten – also verschiedene smarte Dinge – sowohl untereinander als auch mit Rechnern am Internet so zu vernetzen, dass sie alle möglichen Internetdienste nutzen können, um dadurch die Erbringung einer breiten Palette neuer innovativer Services überall und jederzeit zu ermöglichen. Dank des IoT werden bald alle technischen Dinge, insbesondere die unseres alltäglichen Lebens, den Menschen überall und jederzeit zugänglich und somit nutzbar sein.

Weiterentwicklung des Internet

- Die Komplexität des zukünftigen Internet und der Weiterentwicklung seiner Anwendungen kann an dieser Stelle auch nicht annähernd dargestellt werden. Daher greifen wir die Diskussion nach Vorstellung der bestehenden Grundlagen erneut im abschließenden Kapitel 18 auf.

1.8 Verständnisfragen

1. Rekapitulieren Sie den Aufbau des OSI-Referenzmodells.
2. Welche Schicht hat welche Funktion?
3. Wie sieht das TCP/IP-Modell aus?
4. Welche Dienste stellt IP bereit?
5. Was unterscheidet TCP von UDP?
6. Was ist verbindungslose versus verbindungsorientierte Kommunikation?
7. Welche Bedeutung hat die Funktion 'Flusskontrolle' in den Kommunikationsprotokollen und worin besteht ihre Idee?
8. Was ist der Unterschied zwischen *Service Datagram Unit* (SDU) und *Protocal Datagram Unit* (PDU)?
9. Was ist ein Frame und was ein 'Service Access Point' (SAP)?
10. Was ist ein 'Payload'?
11. Welche Aufgabe muss im Schichtenmodell beim Übergang einer Informationseinheit von Schicht $N-1 \rightarrow N$ und umgekehrt von Schicht $N \rightarrow N+1$ realisiert werden?
12. Die rasante und erfolgreiche Entwicklung des Internet ist der Organisation IETF zu verdanken. Wie wird die Entwicklung von Internet-Standards, von sog. RFCs, durch die IETF koordiniert?

2 Sicherheit in 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 Jahren des letzten Jahrhunderts vorherrschende Nutzung des Internet im wissenschaftlichen Betrieb zu sehen. Die Qualifizierung an seiner Teilnahme erfolgte ausschließlich durch die vorhandenen technischen Möglichkeiten seiner Teilnehmer.

Alice, Bob und Eve

Der erste Schritt bestand natürlich darin, das Internet überhaupt für die Anwender bereitzustellen, 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:

Schutzziele bei der IP-Kommunikation

Die Kommunikationspartner müssen gegenseitig bekannt sein, um eine unerwünschte Datenweitergabe (data leakage) zu unterbinden, was sowohl eine *Authentisierung* als auch *Autorisierung* beinhaltet, die Systeme, also die Endsysteme als auch das Netzwerk mit seinen Kommunikationskomponenten, müssen *zuverlässig* sein und sich im Fehlerfalle *robust* verhalten.

Überblick über das Kapitel

Daher haben wir dieses Kapitel wie folgt gestaltet:

Daten im System

- Zunächst wollen wir eine kurze Replik auf die technische Entwicklung der IT-Sicherheit und ihrer Wurzeln vornehmen, die Rolle der Daten und der mit ihnen umgehenden Systeme, d.h. der Akteure, in einem Modell beschreiben, was in der Darstellung der heute genutzten vier Primitiven der IT-Security mündet.

Krypto-Primitive

- Der Sicherung der Datenübertragung und -speicherung ist der folgende Abschnitt gewidmet, wobei hier die Ideen der vier Krypto-Primitiven entwickelt werden, von denen anschließend umfangreich Gebrauch gemacht wird.

Klassische Verschlüsselung und Hashes

- Die klassische symmetrische Verschlüsselung findet sich im nächsten Abschnitt wieder; wobei wir auf Strom- und Blockchiffren sowie auf die Verschränkung der verschlüsselten Datenblöcke im Betriebsmode eingehen. Zudem werden auch Hashfunktionen und ihre heutige Nutzung ausführlich behandelt.

Public-Key-Kryptographie

- Die *Public-Key-Kryptographie* liefert uns die beiden Krypto-Primitive *Schlüsseltausch* und *Signierung*, wobei der Schlüsseltausch speziell auf Grundlage des RSA- als auch der Diffie-Hellman-Algorithmen und die Nutzung digitaler Signaturen mittels X.509-Zertifikaten vorgestellt wird.

Digitale Identitäten

- In den IT-Systemen wird nicht anonym agiert, sondern sowohl die Benutzer als auch die Systeme besitzen eine *digitale Identität* und sind somit authentifizierbar. Wie dies umgesetzt werden kann, wollen wir ebenfalls beleuchten.

2.1 Grundlagen und Entwicklung der IT-Sicherheit

Mit den *Snowden*-Veröffentlichungen und den vermeintlichen 'russischen Hackerangriffen' sind IT-Sicherheit und Cybersecurity in aller Munde. Spätestens mit dem Inkrafttreten der Datenschutzgrundverordnung (DSGV) im Jahr 2018 ist auch Datenschutz aus der Ecke des 'Datenschutzbeauftragten' (DSB) in die öffentliche Wahrnehmung gerückt.

Der Gegenstand und das Zusammenspiel von IT-Sicherheit und Datenschutz wird häufig bis hinauf in kompetente Stellen wie dem Bundesamt für Sicherheit im Informationswesen (BSI) nicht klar differenziert, sondern gemeinsam als *Informationssicherheit* zusammengefasst. In klassischer Lesart möchten wir aber unterscheiden in Bezug auf

- den *Datenschutz* – vor allem in Form digital vorliegender Informationen – was ihre Vertraulichkeit betrifft,
- die *IT-Security* – die sich auf die IT-Systeme bezieht und wie diese die Daten vertraulich und korrekt jederzeit bereitstellen können.

Letzteres umreißt das bekannte 'magische IT-Security-Dreieck' (CIA), das aus der militärischen Nomenklatur entsprungen ist und die *Schutzziele* der IT-Security beschreibt:

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*.

Obwohl das Internet dem ARPANet [Abb. 1.1-1] entsprungen ist, haben bei der Entwicklung der Netzwerkkommunikation diese militärischen Ziele nicht unmittelbar im Vordergrund gestanden und wurden nur da beachtet, wo es unerlässlich ist; wurde doch das Internet als prinzipiell offenes System [Abb. 1.3-4] verstanden.

Im Grunde kann man das Internet als 'Informationsmarkt' verstehen, der über nahezu beliebige und kostengünstige Ressourcen verfügt und zudem – in seiner heutigen Ausprägung – für praktisch jedermann unbeschränkt nutzbar ist. Wie auf jedem offenen, ungeregelten Markt gibt es Akteure mit ganz unterschiedlichen Interessen: Von der Verbreitung von Wissen und Know-how über kostenpflichtige Dienste bis zu den bekannten 'Fake News'-Verbreitern. Es ist wichtig, dieses Zusammenspiel zu verstehen und die daraus erwachsenden Konsequenzen zu berücksichtigen.

2.1.1 Daten und ihre Nutzung

Abb. 2.1-1 zeigt ein einfaches Modell der Datennutzung, was aber bereits die wesentlichen Merkmale in Form einer Grammatik beschreibt:

- *Subjekte* sind IT-Systeme und Netze, aber auch Menschen,
- die *Operationen* (Verarbeiten, Speichern, Transportieren)

- von *Datenobjekten* in verschiedenen Ausprägungen vornehmen.

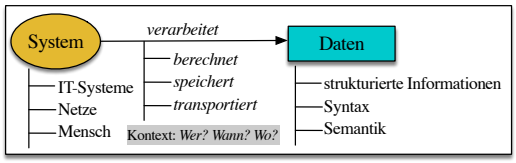


Abb. 2.1-1: Modell der Datenverarbeitung

Daten begreifen wir als strukturierte Informationen, die sowohl *Syntax* als auch *Semantik* und zudem einen *Kontext* aufweisen.

Daten im Netz → Datenpakete

Bei der Datenübertragung über Netze werden die Daten in Nachrichten-Container [Abb. 1.5-1] variabler Größe gesteckt und üblicherweise als *Pakete* aufgefasst, die einen Paket-Header besitzen und in Form von *Frames* ebenso einen Paket-Trailer. Dieses Konzept hat sich als überaus erfolgreich erwiesen und alle anderen Technologien wie z.B. die verbindungsorientierte Kommunikation in Gänze verdrängt.

Daten in Verarbeitung → Datenworte

Bei der Verarbeitung der Daten spielt die Länge eines *Datenworts* ein hervorgehobene Rolle. Daten, Instruktionen und Hauptspeicheradressen werden in dieser Struktur uniform beschrieben. Während die vorige Rechnergeneration das Datenwort auf 32 Bit begrenzte, arbeiten heute Rechnerarchitekturen mit einer internen 64-Bit-Darstellung, sei es in *Least-Significant Bit* (first) LSB oder *Most-Significant Bit* (first) MSB Organisation. Mittels der 64-Bit-Repräsentierung können $2^{64} - 1$ Byte adressiert werden, was einer Datenmenge von mehr als 18,446 Trillionen Byte entspricht. Die Umstellung der Daten in die verschiedenen Formate wird quasi 'en passant' während des entsprechenden Verarbeitungsvorgangs vorgenommen und ist für den Benutzer heute nicht mehr von Bedeutung.

Der Anwender ist ausschließlich an den Dateninhalten interessiert, wobei die Menge der zu verarbeitenden Daten immer noch eine begrenzende Rolle spielt – speziell dann, wenn die Syntax nicht optimal gewählt ist und andauernd Umrechnungen in verschiedene Datenformate erforderlich sind, die sich stark auf die Performance auswirken.

Data in flight

Aufgabe der Technik der Netze ist es natürlich, die Daten zu transportieren, und zwar dergestalt, dass diese *verfügbar*, *unverfälscht* und ggf. *vertraulich* zwischen den Teilnehmern ausgetauscht werden können. Diese Aspekte werden wir in den folgenden Abschnitten im Detail erläutern.

Eine weiteres wichtiges Kriterium besteht darin, die *Authentizität* der Datenquelle (Sender) und der Datensenke (Empfänger) sicherzustellen, sodass Daten mit der notwendigen *Autorisierung* und *Berechtigung* übertragen werden können.

Data at rest

Die langfristige Verfügbarkeit von Daten wird durch ihre persistente Speicherung ermöglicht, die in der Regel auf dem physikalischen Medium blockweise erfolgt. Diese Datenblöcke lassen sich sowohl im Hinblick auf ihre Integrität durch Check-

und Hashsummen sichern, wobei Ersteres auf dem Datenträger selbst und Letzteres vom Dateisystem (wie z.B. ZFS[1]) vorgenommen wird.

Dem Kriterium der Vertraulichkeit kann durch blockweise Verschlüsselung entsprochen werden; die Verfügbarkeit wird durch Redundanzen (z.B. RAID-1[2]) oder mittels Backup/Restore realisiert.

Data in computation

Die Verarbeitung der Daten erfolgt im heutigen Verständnis unverschlüsselt. Daten im Hauptspeicher des Rechners, aber auch in den CPU-Caches werden durch ergänzende Paritätsinformationen gegen Verfälschungen gesichert, sind aber – als transiente Daten – nur solange verfügbar, solange die CPU und der Hauptspeicher mit elektrischer Energie versorgt werden. Die Sicherstellung der Datenintegrität auch in oder nach einem irregulären Betriebszustand muss von der Software-Architektur gewährleistet werden.

Shared IT → Data Leakage

Die Rechnerressourcen, aber auch die Netze werden von mehreren (quasi) zeitgleich laufenden Programmen gemeinsam genutzt, wobei der *Kernel* des Betriebssystems die Aufgabe hat, diese gegeneinander abzugrenzen und somit eine unbeabsichtigte Datenweitergabe (*data leakage*) zu unterbinden. Dass dies nicht so einfach ist, haben die *Spectre* genannten Fehler im Design vor allem der Intel-CPUs gezeigt. Dies gilt im besonderen, da heutige Systeme nicht *single-use*, sondern durch die Virtualisierung [vgl. Kapitel 14] im Rechenzentrum oder in der 'Cloud' [vgl. Kapitel 17] von ganz unterschiedlichen Benutzern und Benutzergruppen geteilt werden.

Erweiterte Datenattribute: Metadaten

Entsprechend Abb. 2.1-1 sind Daten komplexe Objekte mit Syntax und Semantik. Neben diesen inhärenten Eigenschaften besitzen Daten auch *kontextuelle Attribute*, die wir auch als erweiterte Datenattribute bzw. Metadaten bezeichnen und bei der Datenverarbeitung entweder automatisch anfallen oder von prinzipiellem Belang sind:

Gültigkeit
- Wann wurden die Daten erzeugt und wie lange sind die hierin enthaltenen Informationen gültig bzw. von Belang?

Herkunft
- Wer ist der Erzeugende der Daten (die Datenquelle)?

Vertraulichkeit
- Welche Weiterverarbeitungs- bzw. Weitergaberechte sind mit den Daten verknüpft? Im einfachsten Falle lässt sich dies durch die Klassifikation der Daten in 'vertraulich', 'privat' und 'öffentlich' beschreiben.

Metadaten und Datamining

Diese kontextuellen Datenattribute sind nun ihrerseits *Metadaten*, für die im Grunde die gleichen Bedingungen wie für die eigentlichen Daten gelten. Die Datennutzung findet in der Regel *systemisch* statt, d.h. unter Einbeziehung der Metadaten, wie das für *Datamining* heute typisch ist. Das Verständnis der Daten wird dadurch nicht nur von ihrer Semantik, d.h. dem Inhalt bestimmt, sondern ganz wesentlich durch die ergänzenden Metadaten[3].

[1] Solaris Zeta File System ZFS
[2] RAID = Redundant Array of Inexpensive Drives
[3] Dies lässt sich durch folgende Anekdote trefflich illustrieren: https://www.wired.de/article/das-us-militaer-verbietet-fitness-tracking-mit-gps

Die Interpretation von Daten bezieht immer auch den bestehenden Kontext, d.h. die Metadaten, mit ein, durch die die Semantik angereichert wird. Kontextdaten können unterschiedlichere Attribute als die eigentlichen Nutzdaten aufweisen.

IT-Systeme, die naturgemäß diese Daten verarbeiten, müssen aufgrund ihrer Architektur und dem definierten Workflow auf diese Attribute Rücksicht nehmen, was besonders auch der Gesetzgeber im Rahmen der viel diskutierten 'Datenschutzgrundverordnung' (DSGVO) fordert.

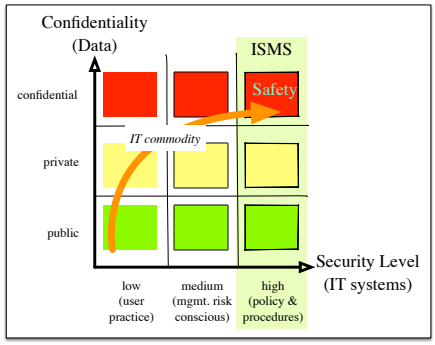


Abb. 2.1-2: Zusammenspiel von Datensicherheit und IT-Security
ISMS: IT Security Management System, mgmt.: Management

Data Safety

Dieser Zusammenhang wird durch den Terminus *Data Safety* verdeutlicht. Abb. 2.1-2 zeigt das Zusammenspiel vom Vertraulichkeitsattribut und der Qualifikation von IT-Systemen: Systeme auf niedriger IT-Security-Stufe, z.B. auch IoT-Devices [vgl. Kapitel 17], sollten nur öffentliche und Kontext-irrelevante Daten verarbeiten, während vertrauliche Daten eines hohen IT-Security-Niveaus bedürfen. Im FinTec-Bereich und speziell in Banken wird dem durch die Einführung eines expliziten *IT Security Management Systems* (ISMS) Rechnung getragen.

IT-Commodity-Systeme

IT-Commodity-Systeme wie IoT-Devices, Tablets und PCs eignen sich nur mit speziellen Werkzeugen für die Handhabung vertraulicher Daten und müssen einem ständigen Monitoring unterworfen werden.

2.1.2 Akteure und Identitäten bei der Datenverarbeitung

Jeglicher Datenaustausch bedingt eine *Datenquelle* und eine *Datensenke*. Ist die Datenquelle die Erzeugende, ist sie somit auch der natürliche Dateneigner, und diese stehen quasi zur freien Verfügung. In allen anderen Fällen müssen die Fragen gestellt werden:

- Dürfen die Daten, die von Dritten stammen, weitergegeben werden?
- Ist es zulässig, die Daten an bestimmte Dritte zu übermitteln?

Unabhängig davon, welches Vertraulichkeitsattribut die Daten aufweisen, so ist doch Herkunft und Ziel der Daten zu bestimmen. Technisch gesehen kann das bei Internet-gestützten Systemen durch die Angabe der IP-Adressen [siehe Kap. 3] von Sender

und Empfänger realisiert werden. Wie aus Abb. 2.1-1 hervorgeht, ist dies aber nicht ausreichend:

- Es ist vielmehr immer ein *Prozess*, der die Datenverarbeitung vornimmt;
- Netze besitzen hier nur eine Vermittlerrolle, sind aber für die vertrauliche und korrekte Weiterreichung der Daten verantwortlich.
- Falls notwendig, muss zusätzlich die Möglichkeit bestehen, den ausführenden *Menschen* zu identifizieren.

Identitäten

Menschen und Prozessen muss somit eine *Identität* zugewiesen werden, der sie unterscheidbar macht. Im einfachsten Fall ist dies eine fortlaufende Nummer, wie dies z.B. bei Prozessen[4] üblich ist. Bei rechnergestützten Prozessen ergänzt man dies um den Hostnamen, z.B. in der Form *pid@host*. Die pid ist nun aber lediglich die Instanz eines Prozesses, die aktuell ausgeführt wird, der Prozess selbst bekommt in der Regel einen 'sprechenden' Namen wie z.B. *daemon*, der für alle Instanzen benutzt wird, also beispielsweise als *daemon@host*.

Name = Öffentliche Identität

Ebenso wird zur Identifikation von Menschen der *Username* verwandt, der aber natürlich nicht eindeutig sind. Um Personen, aber auch Prozesse ansprechen zu können, müssen diese bekannt sein, wie dies beispielsweise bei Webanwendungen über die URL [Abb. 1.1-4] gemacht wird. Namen kennzeichnen somit eine *öffentliche Identität*, wobei der Grad der Öffentlichkeit auch beschränkt sein kann, z.B. auf eine Benutzergruppe, die Teilnehmer nur über deren *Aliasnamen* bzw. *Pseudonyme* kennt.

Authentisierung

Identitäten müssen gegenseitig und gegenüber Dritten *beglaubigt* werden und eindeutig sein, d.h. immer auf die gleiche Person bzw. den gleichen Prozess verweisen. Die Beglaubigung lässt sich organisatorisch (wie z.B. beim Personalausweis) oder aber kryptographisch lösen und verlangt einen eigenen *Authentisierungsprozess*, der zwei Endzustände kennt:

1. Die Person oder der Prozess kann aufgrund der vorliegenden Informationen nachvollziehbar einer Identität – und nur genau dieser – zugewiesen werden
2. Die Informationen widersprechen sich oder sind nicht aussagekräftig genug, um die Identität zweifelsfrei feststellen zu können.

PSK-Verfahren

Das Authentisierungsproblem lässt sich notwendig aber nicht hinreichend lösen, falls zusätzlich zum öffentlichen Teil der Identität ein geheimer, *privater* Teil hinzukommt, der nur den beteiligten Parteien bekannt ist und vertraulich ausgetauscht wurde: das *Passwort*. Daher sprechen wir hier auch von *Pre-shared Key* PSK-Verfahren.

1-Faktor-Authentisierung

Da die Identität öffentlich bekannt ist, ist das lediglich das Passwort der (kryptographisch) entscheidende Teil, und wir bezeichnen dies als *1-Faktor-Authentisierung* (1FA). Neben der Tatsache, dass es anderen nicht bekannt sein darf, ist die Qualität des Passworts im Hinblick auf seine Länge und seine *Entropie*[5] maßgeblich, da es häufig über einen längeren Zeitraum unverändert genutzt wird.

'Sichere' Passwörter sollten aber nur ein einziges Mal Einsatz finden, wofür es eine Reihe von technischen Lösungen und Anwendungen gibt:

Einmal-Passwörter

- Bei der IP-Kommunikation zwischen Prozessen werden insbesondere unter Nutzung der *Transport Layer Security* [Abschnitt 7.2] einmalig generierte Passwörter zur Sicherstellung der Authentizität der (verschlüsselten) Datenpakete für jede TLS-Sitzung benutzt. Diese werden mittels einer *Pseudo-Random Function* PRF oder einer *HMAC-based Key Derivation Function* HDKF generiert.

TLS → PRF oder HKDF

- Bei einigen *One-Time Password* OTP-Verfahren wird das einmalige Passwort von einem statischen Passwort und dem aktuellen Zeitfenster abhängig gemacht. Hierbei müssen die Komponenten *zeitsynchron* arbeiten.

One-Time Password

> Statisch vergebene Passwörter zur Authentisierung der kommunizierenden Instanzen finden nur bei wenigen Protokollen Einsatz, so wie dem in Abschnitt 15.3 beschriebenen RADIUS[6]Verfahren, wo das Passwort zur Verschlüsslung der RADIUS-Nachrichten verwendet wird.

Gegenbeispiel mit statischen Schlüsseln

Eine 2-Faktor- (2FA) oder Mehr-Faktor-Authentisierung (MFA) liegt dann vor, wenn zusätzlich zum Wissen (= des Passworts) ein personalisierter Besitzgegenstand im Authentisierungsprozess vorhanden sein muss, z.B. in Form einer Chipkarte (die auf die Identität ausgestellt sein muss). *Biometrische Informationen* wie z.B. Fingerabdruck- oder Retina-Scan können ebenfalls ergänzend für die Authentisierung von Personen herangezogen werden (Mehr-Faktor-Authentisierung).

2FA und MFA

Biometrie

Autorisierung

In vielen praktischen Anwendungsfällen ist die Identität einer Person oder eines Prozesses von keinem besonderen Belang (und muss auch gar nicht bekannt sein), sondern vielmehr, ob eine Berechtigung vorliegt, beispielsweise einen Service zu nutzen [Abb. 2.1-3].

> Die Ausleihe von Fachartikeln und Büchern in digitaler Form an den Hochschulen ist gestattet, wenn der Student eingeschriebenes Mitglied an einer Hochschule ist. Der Verlag – oder die Datenquelle –, von der aus das Material bezogen wird, nutzt dann den sogenannten *Shibboleth*-Dienst[7] der Hochschule: Der Student muss sich dann gegenüber der *eigenen* Hochschule mittels seiner Kennung (häufig Matrikelnummer) und Passwort authentisieren. Nur das Ergebnis der Überprüfung wird an den Verlag weitergereicht.

Shibboleth an Hochschulen

Kennzeichnend ist hierbei Folgendes:

- Die Autorisierung ist ein *interpersonelles* Attribut, das von einer Gruppe von Nutzern geteilt werden kann.
- Es liegt eine *transitive* Authentisierung vor, wobei ein spezieller Authentisierungsdienst mittels eines *Identity Service Providers* IdP zum Einsatz kommt

Identity Service Provider

Will man Authentisierung unabhängig von einer oder mehreren Benutzergruppen machen – wie sie heute beispielsweise auf Grundlage von privaten Anbietern wie *Facebook*, *LinkedIn* und *Google* vorliegen – ist ein System zur Verwaltung digitaler

Digitale Identitäten

[4] die Process Identification `pid`
[5] wir können an dieser Stelle 'Entropie' mit Zufälligkeit gleichsetzen
[6] RADIUS: Remote Dial-In User Service
[7] siehe: https://www.shibboleth.net

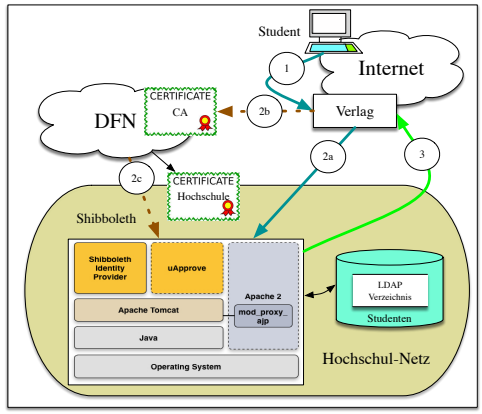


Abb. 2.1-3: Shibboleth als IdP an Hochschulen
(1) Student will über Verlag Fachartikel einsehen, (2a) Anfrage geht an den Shibboleth-Server der HS und Student meldet sich dort an, (3) erfolgreiche Anmeldung geht an Verlag; alternativ: (2b) Anfrage geht über den DFN, der (2c) die Anmeldung an die HS weiterleitet.

Identitäten zu schaffen. Mittels der modernen *Publik Key*-Kryptographie und dem Aufbau von öffentlichen und privaten Zertifizierungsstellen, den *Certificate Autorities* CA oder auch *Trust Center* TC, ist dies technisch seit etwa 1995 möglich: *Public Key Infrastructure* PKI. Der zu betreibende Infrastrukturaufwand ist aber insbesondere seitens staatlicher Seite beachtlich, und so hat es bislang nur Estland als relativ kleine Nation geschafft, dies auch in allen Aspekten, die die öffentliche Verwaltung betreffen, zu realisieren.

Public Key Infrastructure

X.509-Zertifikate

Anders sieht die Situation in Unternehmen aus: Anwendungen werden *X.509-Zertifikate → Digitale Identitäten* ausgestellt, die mit sogenannten *key files* komplettiert werden. Hierdurch erhalten Anwendungen *digitale Identitäten*, die mit bestimmten Rollen und Rechten verknüpft sein können. Auf die technischen Grundlagen hierzu gehen wir in Abschnitt 2.6 noch genauer ein.

2.1.3 Entwicklung der Internet-Kryptographie

Wie bereits Abb. 2.1-1 darstellt, geht es bei der IT-Sicherheit um die Sicherheit, d.h. Vertraulichkeit, Integrität und Verfügbarkeit der von IT-Systemen verarbeiteten Daten. Mit der Entwicklung von IT-Systemen, sprich Computern, seit Endes des 2. Weltkrieges nimmt die IT immer mehr Einzug in das tägliche Leben: Wirtschaft, Kultur und Politik sind ohne Computer nicht mehr vorstellbar. Die Abhängigkeit vom Funktionieren der IT-Systeme ergibt sich insbesondere auch für die IT-gesteuerte Infrastruktur wie Kommunikationssysteme und elektrische Energieversorgung: Die 'analoge' Welt wird durch die digitale abgelöst. Wir benötigen daher nicht nur Datenschutz und IT Security, sondern sowohl die IT-Systeme als auch die Daten müssen zuverlässig zur Verfügung stehen: *IT Safety*.

1.1.1970: Beginn des Internet Zeitalters

Einschneidende Entwicklungen sind seit Beginn der 70er Jahre des letzten Jahrhunderts zu verzeichnen: die Geburt der Computernetze und die Entwicklung leistungsfä-

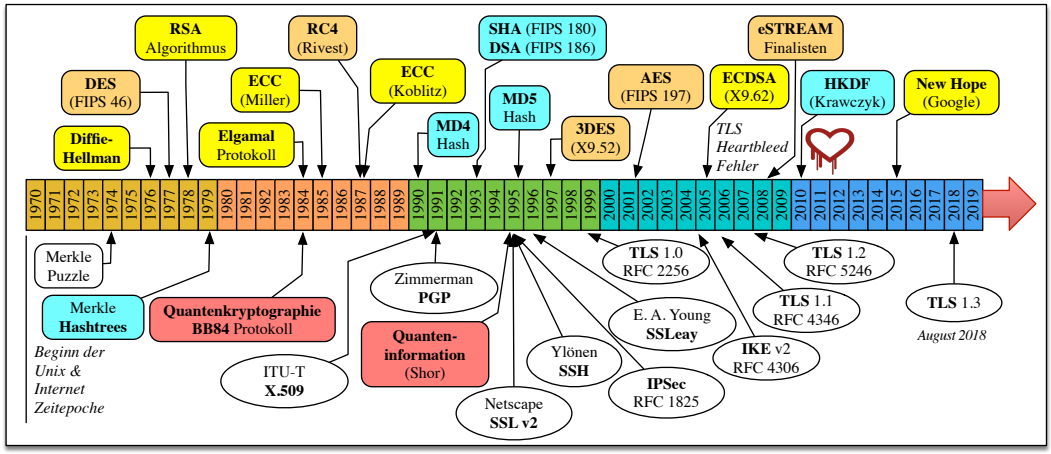


Abb. 2.1-4: Entwicklung und Meilensteine der wichtigsten kryptographischen Errungenschaften der Internet-Zeit.
Oberhalb der Zeitleiste: Wichtige Algorithmen und Standards; unterhalb: Implementierungen; Hashtrees und Quantenkryptographie werden im Buch nicht weiter besprochen; Gelb: Schlüsseltausch- und Signaturprotokolle, orange: symmetrische Verschlüsselung; türkis: Hashverfahren

higer und preisgünstiger Betriebssysteme, insbesondere Unix, das die IP-Technologie von Anfang an beinhaltete. Deshalb kann auch der 1.1.1970 als das Geburtsdatum des Internet-Zeitalters bezeichnet werden. Die Uhren der Betriebssysteme führen dieses Datum als Beginn der 'Neuzeit' und ihrer Zeitrechnung.

1970 - 1980: Lösung des Schlüsseltausch-Problems

Mit der Nutzung der Internet-Technologie [vgl. Abschnitt 1.1] ergab sich sehr früh die Notwendigkeit, die Kommunikation vertraulich ablaufen zu lassen. Die kryptographischen Grundlagen der symmetrischen Verschlüsselung waren hinreichend bekannt, verlangten aber einen effizienten, Computer-gestützten Algorithmus – und stellten unmittelbar die Frage des Schlüsseltausches zwischen den Teilnehmern. Letzteres wurde – nachdem die zentrale Frage durch *Ralph Merkle* [Mer78] einmal gestellt war – durch *Diffie* und *Hellman* [DH76] und später durch den RSA-Algorithmus von *Rivest*, *Shamir* und *Adleman* – durch die bahnbrechende Entdeckung der *Public-Key-Kryptographie* gelöst. Mit der Verfügbarkeit des *Data Encryption Standards* DES als Blockchiffre mit nachprüfbarer Güte waren bis Ende der 70er Jahre die wesentlichen Grundlagen geschaffen worden.

1980 - 1990: Ergänzung der Grundlagen

Kryptographie entstammt aus dem Mathematikgebiet der Zahlentheorie. Die Größenordnung der natürlichen Zahlen, von denen hier Gebrauch gemacht wird, ist schwindelerregend: Quadrillionen ($10^{24} \sim 2^{80}$) sind hier als 'kleine' Zahlen zu verstehen. In der 80er Jahren gab es erste Vorstöße, den Zahlenraum der natürliche Zahlen zu verlassen und sich algebraischen Strukturen wie *Galois-Feldern* und damit den Lösungen polynomialer Gleichungen und hier speziell den *elliptischen Kurven* zuzuwenden [Mil85; KKM85]. Zudem gab es erste Ansätze, quantenmechanische Effekte für die Datenübertragung zu nutzen [BB84; Sho95].

1990 - 1995: Hashes und Public Key Infrastructure

Anfang der 90er Jahre lag der Fokus aber zunächst auf der praktischen Umsetzung kryptographischer Verfahren: Nachdem zusätzlich die *Stromchiffre* RC4 entwickelt wurde, war als nächster Durchbruch die Entwicklung von Hashfunktionen zu verzeichnen. Zwar waren Hashes auf Grundlage von DES durch einen Trick bereits im Einsatz [Abb. 2.3-2], die explizite Verfügbarkeit von Hashfunktionen (MD5 und SHA) gestattete es jedoch, beliebig große Datenmengen in kurzer Zeit mit einer Hashsumme zu versehen. Das war der Türöffner für digitale Signaturen: Das Konzept der *X.509 Zertifikate* und des Aufbaus einer *Public Key Infrastructure* war geboren.

1995 - 2000: Rapide Implementierungen

Damit waren im Grunde alle Bausteine für die moderne Kryptographie vorhanden, und es ergaben sich rapide erste Lösungen: *Phil Zimmermann* entwickelte *Pretty Good Privacy* PGP für den E-Mail- und Datenaustausch. In Ergänzung zu den *Netscape Security Services* NSS auf der Serverseite, wurde für den Webbrowser *Netscape* das Protokoll *Secure Socket Layer* SSL hervorgebracht. Der Finne *Tatu Ylönen* stellte mit seiner *Secure Shell* SSH einen sicheren Ersatz für die unverschlüsselte Datenübertragung mittels TELNET und FTP bereit. Im Zuge der Entwicklung der IPv6-Protokolls wurde auch die Erweiterung IPSec vorgenommen, die auf dem *Internet Key Exchange* IKE aufsetzt.

2000 - 2010: Konsolidierung und umfangreiche Verbreitung

Ab dem Jahr 2000 kann von einer Konsolidierungsphase gesprochen werden. Die diversen Implementierungen, die teilweise unter US-Exportrestriktionen litten (und diese wiederum umschifften), wurden in Internet-RFCs öffentlich gemacht und somit offiziell sanktioniert. Mit dem *Advanced Encryption Standard* AES betrat ein stärkerer und flexibler Nachfolger für DES für die Blockverschlüsselung den Ring, und es entwickelte sich die *Elliptic Curve Cryptography* ECC als Ergänzung zum Diffie-Hellman-Schlüsseltausch mittels diskreter Logarithmen. Im Rahmen des europäischen eSTREAM-Projekts [Bab+08] wurden zudem neue Stromchiffren entwickelt, die die bisherigen ablösen sollen. Auch das auf der INDOCRYPT 2004 [MV04] vorgestellte Verfahren des *Galois Counter Mode* sollte sich als wegweisend herausstellen und beendet die lange geführte Debatte 'MAC-then-crypt' durch einen komplett neuen, einfacheren und sichereren Ansatz. Ergänzend hierzu wurde von *Hugo Krywczyk* aus den IBM Watson Labs ein neuer Ansatz vorgestellt, wie kryptographisch *sichere* Passwörter mit hoher Entropie mittels einer *HMAC-based Key Derivation Function* [Kra10] erzeugt werden können.

2010 - 2020: Ende des Dornröschenschlafs und Post-Quantum-Kryptographie

Mit der Ruhe war es dann etwa ab dem Jahr 2010 vorbei: Im TLS-Protokoll, das für den verschlüsselten Internet-Datenverkehr das Arbeitspferd ist, wurden viele Implementierungslücken aufgedeckt, und mit dem *Heartbleed*-Fehler bei TLS zeigte sich, wie verletzlich die Internet-Kommunikation geworden ist. Zudem wurde offenkundig, dass die Public Key Infrastructure durch nachlässigen Umgang mit der Technik und Root-Zertifikaten sowie durch Geheimdienste kompromittiert war und damit eher ein Problem als eine universelle Lösung darstellt. Neue Ansätze, die auch die vorwiegend theoretische Diskussion um die *Post Quantum Cryptography* PQC[8] mit einschließen, werden z.Zt. entwickelt. Hier sei auf den *New Hope*-Algorithmus verwiesen, den

[8] siehe: http://pqcrypto.org, https://csrc.nist.gov/Projects/Post-Quantum-Cryptography

Google in seinen Chrome-Browser einbaut. Das stark überarbeitete TLS-Protokoll fand im August 2018 in Version 1.3 endgültig Niederschlag in RFC 8446.

2.1.4 Schichtenspezifische IT-Security-Protokolle

PKI

Mit der Erfindung 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 sowie dem Apache Webserver.

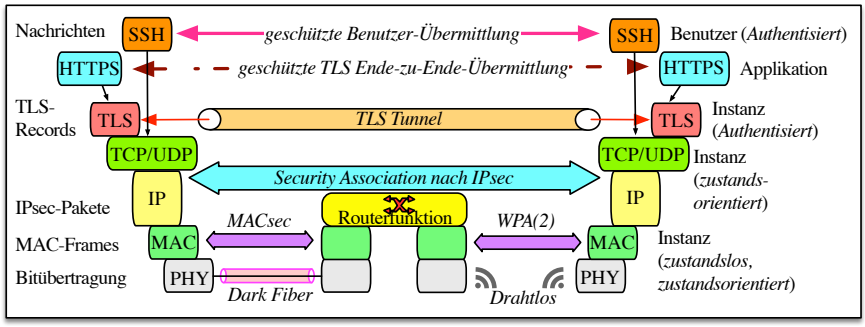


Abb. 2.1-5: Gesicherte IP-Kommunikation auf den verschiedenen Schichten
PHY: Physikalische Schicht, MAC: Media Access Control Schicht, IP: Internet Protocol, IPsec: IP Security, TCP: Transmission Control Protocol, TLS: Transport Layer Security, HTTPS: Hypertext Transport Protocol (Secure), SSH: Secure Shell

Abb. 2.1-5 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 Bits sozusagen 'privat' transportiert[9] werden.

  Die Übertragung per 'Dark Fiber' bedeutet nicht unbedingt, dass die Signale zusätzlich verschlüsselt sind. Wie Abb. 2.1-5 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 Anwendungen hierüber informiert sind.

Link-Level-Verschlüsselung

- Link-Level-Verschlüsselung findet üblicherweise im WLAN statt, wobei hierfür das weit verbreitete Protokoll WPA2 [Abschnitt 13.3] eingesetzt wird. Die Netzwerkschlüssel werden hierbei entweder statisch aufgrund eines *Pre-shared Keys* PSK

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

oder dynamisch, d.h. ephemeral mittels des *Extensible Authentication Protocols* EAP zur Verfügung gestellt [Abschnitt 15.1]. Dieses auf IEEE 801.1X basierende Verfahren wurde auf das Ethernet-LAN übertragen und wird als MACsec geführt [Abschnitt 14.4.4]. Interessant ist MACsec für Internetprovider, die eine Kopplung ihrer Netze (Cross-Connect) vornehmen und dabei die Daten-Frames quasi über fremdes Terrain übertragen.

IPsec

- Auf der Netzwerkschicht kann als Sicherheitsprotokoll *IP Security* (IPsec) [Abschnitt 6.4] 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 vorausgesetzt.

  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 7.2]. 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 *Secure Shell* SSH mit seinen 'Ablegern' *Secure Copy* SCP, *Secure FTP* SFTP und *rsync* genannt. Auch das E-Mail-Verschlüsselungsprotokoll *Pretty Good Privacy* PGP gehört in diese Kategorie, was alledings heutzutage durch die sicheren Messenger-Dienste wie *Threema*, *Jabber* oder *Telegram* weitgehend verdrängt wurde.

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, Autorisierung des *Kommunikationspartners* allerdings nur im Rahmen eines Vertrauensmodells.

Während wir die ersten beiden Aspekte im folgenden Abschnitt diskutieren möchten, greifen wir das Thema der Benutzeridentifikation erst im Kapitel 15 wieder auf.

2.2 Prinzipien und Primitive der IT-Security

Das klassische C-I-A-Dreieck der IT-Security verlangt in Konsequenz, dass die *Availability*, d.h. die Verfügbarkeit von Daten und IT-Systemen, die vorrangige Stelle einnimmt: Ohne Verfügbarkeit spielt die Vertraulichkeit und Integrität von Daten keine Rolle.

Redundanzen

Die Verfügbarkeit von IT-Systemen hängt mit den Eigenschaften der Hard- und Software zusammen, deren Zusammenspiel von einer Zwischenkomponente, der Firmware (manchmal auch als Treiber bezeichnet), geregelt wird. In Bezug auf die Hardware kann die Verfügbarkeit durch Redundanzen, d.h. doppelter Auslegung im Rahmen eines *Cold-* oder *Hot-Standby* sowie *Loadsharing* verbessert werden. Die Robustheit von IT-Systemen ist somit nicht nur eine intrinsische Eigenschaft, sondern auch eine Frage der Architektur.

Software-Architektur

Auch bei Software ist eine an die Aufgabenstellung angepasste Architektur maßgeblich, was in Rahmen des *Software Engineerings* SWE behandelt wird. Die SW-Architektur stellt das Bindeglied zwischen den fachlichen und den nicht-fachlichen Anforderungen[10] dar. Zu den IT-Security-relevanten NFRs zählen die *LSD-Prinzipien*:

- **Least Privileges**: Prozesse, die Daten verarbeiten, dürfen nur mit geringsten Systemrechten ausgestattet sein und den 'Security-Kontext' des Anwenders nicht negativ beeinflussen.
- **Segregation of Duties**: Ein Prozess sollte nur wenige Verarbeitungskompetenzen besitzen, d.h. sollte 'single-use' sein. Weitere Verarbeitungsschritte werden von dedizierten Folgeprozessen vorgenommen[11].
- **Domain Principle**: Daten sollen in der Domain verarbeitet werden, wo sie entstehen.

Diese Grundsätze der Datenverkapselung und -verarbeitung müssen in der Praxis durch eine fachgerechte Implementierung und deren Tests ergänzt werden. Die Tests beinhalten Modul-, Integrations- und Systemprüfungen. Zusätzlich können auch Performance- bzw. Regressionstests vorgenommen werden [Bla09].

IT-Systeme, die diese Qualifikation erfüllen, sind auch zur Verarbeitung vertraulicher Daten geeignet, sofern von den folgenden vier Primitiven der Kryptographie Gebrauch gemacht wird, die zudem die Integrität und Herkunft der Daten verifizierbar machen.

2.2.1 Verschlüsselungs-Primitiv $\mathbb{C}$

Die Sicherung der Vertraulichkeit (engl. *confidentiality*) von Daten und Informationen ist eine Notwendigkeit, die im zivilen Leben die Privatsphäre sichert, im militärischen aber dazu dient, den 'Feind' über die eigenen Mitteln und Vorgehensweisen im Unklaren zu lassen.

Secret-Key-Kryptographie

Bei der klassischen Verschlüsselung von Daten wird ein Schlüssel genutzt, der den Anwendern bekannt sein muss, aber ansonsten von keinem Dritten; der Schlüssel ist also geheim: *Secret Key Cryptography*. Bezeichnen wir die Klartextdaten mit x und den Schlüssel mit k, ergeben sich zwei Operationen:

- *Verschlüsselungsoperation*: $x' = \mathbb{C}(x, k)$
- *Entschlüsselungsoperation*: $x = \mathbb{C}^{-1}(x', k)$

Die verschlüsselten Daten wurden als x' eingeführt und die Schlüsseloperation bzw. Chiffrefunktion mit $\mathbb{C}$. Der geheime Schlüssel k dient sowohl zum Ver- als auch zum Entschlüsseln, was auch die Wortwahl 'symmetrische Verschlüsselung' begründet. Bei aktuell implementierten Verschlüsselungsfunktionen wie AES gilt zudem: $\mathbb{C} = \mathbb{C}^{-1}$, d.h. Ver- und Entschlüsselung findet über den gleichen Algorithmus statt, wodurch die Implementierung deutlich vereinfacht wird.

Zustandslose und zustandsbehaftete Schlüssel

Somit können Operationen wie Festplattenverschlüsselung oder das Anlegen und Auslesen des Passwort-Safes trivial gelöst werden. Hierbei bleibt der geheime Schlüssel über einen längeren Zeitraum unverändert; er ist *zustandslos*. Sind die Daten aber mit einem Partner, z.B. über das Netzwerk auszutauschen, muss dem Gegenüber der Schlüssel (und natürlich auch der Algorithmus) bekannt sein, damit die empfangenen, verschlüsselten Daten wieder entschlüsselt und somit im Klartext gelesen werden können. Hierbei empfiehlt es sich nicht, immer den gleichen Schlüssel zu verwenden, sondern es sollte bei jeder Transaktion der Schlüssel gewechselt werden. Die Transaktions- bzw. Sitzungsschlüssel sind daher nur temporär gültig bzw. ephemeral und somit *zustandsbehaftet*.

Bevor wir in Abschnitt 2.4 die technische Implementierungen von $\mathbb{C}$ besprechen, müssen wir zunächst klären, wie unser Gegenüber zu seinem Sitzungsschlüssel kommt.

2.2.2 Schlüsseltausch-Primitiv κ

Das Schlüsseltauschproblem ist eine asymmetrische kryptographische Operation, daher auch häufig die Bezeichnung 'asymmetrische Verschlüsselung'. Allerdings ist der Terminus 'Verschlüsselung' unzutreffend; Algorithmen wie Diffie-Hellman dienen dazu, sich auf einen gemeinsamen Schlüssel zu verständigen. Folgendes Beispiel beschreibt zunächst den Grundgedanken beim RSA-Schlüsseltausch κ, mit dessen Hilfe *Alice* und *Bob* zu einem geheimen Schlüssel zu gelangen, um hiermit ihre vertraulichen Daten gegen Dritte mittels des Verschlüsselungs-Primitivs $\mathbb{C}$ schützen zu können.

Schlüsseltausch von Alice und Bob

> Bob besitzt eine hinreichend sichere Schachtel mit einem Schnappschloss, das nur er mit seinem privaten Schlüssel öffnen kann. Die Schachtel wird nun von Alice angefragt und Bob händigt Alice diese in geöffnetem Zustand aus. Die Schachtel an sich ist wertlos und enthält ... nichts weiter, muss aber Bob zugewiesen sein. Nun generiert Alice einen Sitzungsschlüssel, legt ihn in die Schachtel und lässt diese zuschnappen. Alice bittet nun Charlie, die Schachtel mit ihrem Schlüssel an Bob zu übergeben. Da die Schachtel verschlossen ist und nur Bob sie öffnen kann, bleibt der Sitzungsschlüssel für Charlie im

[10] engl.: Functional and None-functional Requirements FR/NFR
[11] siehe: https://cr.yp.to/qmail/qmailsec-20071101.pdf

> Verborgenen. Bob öffnet nun mit seinem privaten Schlüssel die Schachtel und entnimmt den Sitzungsschlüssel.

Public-Key-Kryptographie

Damit der Schlüsseltausch erfolgen kann, wird ein öffentlicher und privater Schlüssel von demjenigen benötigt, mit dem eine verschlüsselte Konversation vorgenommen werden soll: der Empfänger bzw. Receiver r. Seinen *public key* wollen wir als p_r bezeichnen und seinen *private key* mit $\bar{p}_r$.

Die Schlüsseltauschoperation für den gemeinsamen Schlüssel s läuft nun in zwei Schritten ab:

- Initiierung des Schlüsseltausches (vom Sender aus): $s' = \kappa(s, p_r)$
- Vollendung des Schlüsseltausches durch den Receiver: $s = \bar{\kappa}(s', \bar{p}_r)$

Hierbei muss der *Sender* über den *public key* p_r von *Receiver* verfügen, und natürlich nutzen auch beide Partner den gleichen Algorithmus für den Schlüsseltausch, damit der 'verschlüsselte' Sitzungsschlüssel aus s' korrekt entnommen werden kann. Die Schlüsseltauschfunktionen (*key exchange*) κ und $\bar{\kappa}$ sind hierbei unterschiedlich – und auch unterschiedlich schnell! Wichtig ist auch, dass es eine ein-eindeutige Zuordnung von *private key* und *public key* gibt: $p_r \Leftrightarrow \bar{p}_r$.

Der Schlüsseltausch erfolgt hierbei *anonym*, was eine fatale Konsequenz aufweisen kann:

Man-in-the-Middle-Angriff

> Wie gezeigt, hängt der Schlüsselaustausch von der Qualität der Schachtel und des Schlosses ab. Charlie muss sich anstrengen, diese unbemerkt zu öffnen, um den Sitzungsschlüssel zu entnehmen und um damit die von ihm abgefangenen vertraulichen Nachrichten zu entschlüsseln. Wird Charlie auch zur Übergabe der leeren Schachtel bemüht, kann er aber cleverer vorgehen: Er nimmt die Schachtel von Bob entgegen, tauscht sie durch seine eigene Schachtel aus und reicht diese an Alice weiter. Auf dem Rückweg öffnet er nun (seine) geschlossene Schachtel, entnimmt den Sitzungsschlüssel und legt einen neuen (seinen) eigenen in Bobs Schachtel, um diese an Bob zu übergeben. Somit kann Charlie nun – als Beteiligter im Datenaustausch – komplett unbemerkt und ohne Aufwand die verschlüsselte Konversation zwischen Bob und Alice verfolgen: ein *Man-in-the-Middle*-Angriff (MitM) wurde durchgeführt!

Anonymer Schlüsseltausch

Ein anonymer Schlüsseltausch ist nur für Szenarien ratsam, wo ergänzende Nachrichtenauthentisierung vorgenommen werden kann. In allen anderen Fällen ist die Authentizität zumindest des Empfängers (Receivers) sicherzustellen, was glücklicherweise auch über die 'asymmetrische' Kryptographie möglich ist; nun aber mit umgekehrten Rollen für den *private* und *public key*. Zuvor benötigen wir aber noch eine weitere wichtige Kryptofunktion, das *Hash-Primitiv*.

2.2.3 Hash-Primitiv h

Ein zentrales Element der IT-Security ist die Möglichkeit, eine Nachricht so zu kennzeichnen, dass diese vom Empfänger als *nicht-verändert* (integer) verifiziert werden kann. Hierbei gehen wir davon aus, dass sowohl die Speicherung, als auch die Übertragung von Nachrichten immer mit Fehlern verknüpft ist, weil externe Einflüsse die Nutzdaten überlagern und modifizieren können.

Hashsumme ≠ Checksumme

Technische Lösungen hierfür bieten Paritäts-Bits und Checksummen, also Informationen, die der Nachricht hinzugefügt werden. Dies kann entweder blockweise oder aber nachrichtenweise, d.h. mit variabler Datenlänge, durchgeführt werden. Interessanterweise waren Hashfunktionen schon früh bekannt und wurden beispielsweise bei der Sicherung von Unix-Passwörtern genutzt; es brauchte aber bis Anfang der 90er Jahre, ehe *Ron Rivest* [Abb. 2.1-4] mit MD4 (*Message Digest 4*) die mathematischen Grundlagen für die nicht-injektiven Hashfunktionen legte.

Merkmale von Hashfunktionen

Hashfunktionen als *Einwegfunktionen* besitzen folgende Eigenschaften [Abb. 2.2-1]:

- Eine Hashfunktion liefert für eine beliebige Nachricht x einen Hashwert h mit *konstanter Wertelänge* L: $h = \mathcal{h}(x)$.
- Es gibt *keine mathematische Umkehrfunktion* für $\mathcal{h}$: $x \neq \mathcal{h}^{-1}(h) \Rightarrow \cancel{\mathcal{h}^{-1}}$.
- Hashfunktionen nutzen den *Avalanche-Effekt*; d.h. wenn der Input-Wert x sich auch nur um einen winzigen Betrag δ ändert, ist der Hashwert ein komplett anderer und lässt sich nicht aus beiden Teilen ableiten: $x + \delta => \mathcal{h}(x + \delta) \neq \mathcal{h}(x) + \mathcal{h}(\delta)$.
- Hashfunktionen müssen *kollisionsresistent* sein, d.h. es darf nicht möglich sein, einen Wert x' mathematisch zu bestimmen, für den gilt: $\mathcal{h}(x) = \mathcal{h}(x')$. Dies hängt natürlich von der Größe der Bildmenge ab; wobei die Urbildmenge als prinzipiell unbeschränkt gilt.

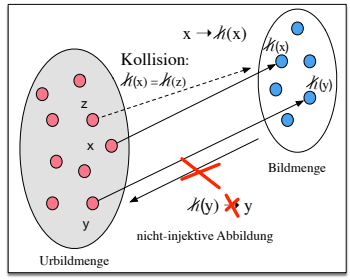


Abb. 2.2-1: Zur Definition von Hashfunktionen;
Die Urbildmenge ist unbeschränkt, während die Anzahl der Elemente in der Bildmenge durch die Forderungen einer konstanten Hashlänge beschränkt ist; bei SHA-256 besitzt die Bildmenge also $2^{256} \approx 10^{77}$ Elemente.

Der Wert h einer Hashfunktion weist einige interessante Eigenschaften auf, die natürlich von der konkreten Hashfunktion abhängig sind:

- Der Hashwert h ist *zustandslos*; d.h. für eine gegebene Hashfunktion $\mathcal{h}$ und bei gleichem Eingabewert x ist er immer identisch.
- Der Hashwert h besitzt eine *maximale Entropie* und verhält sich wie ein Zufallswert, und zwar unabhängig von x. Diese Eigenschaft wird auch häufig als *Pseudo-Zufallswert* beschrieben.

Rainbow-Tabellen

In der Praxis spielt die schnelle Berechenbarkeit von Hashwerten eine große Rolle. Übliche Hashfunktionen wie MD5 und SHA arbeiten schnell, auch bei bei großen Datenmengen für den Eingangswert x. Dies ist dann wichtig, wenn Hashes zur Sicherstellung der Datenintegrität genutzt werden. Die schnelle Berechenbarkeit führt aber

zu dem Phänomen, dass Hashwerte für lexikalische Ausdrücke generiert und 'online' gestellt werden können: *Rainbow-Tabellen*.

Werden Hashes zur Verschleierung von Passwörtern genutzt, sind Hashfunktionen sinnvoll, die viele Systemressourcen (z.B. Hauptspeicher) 'verbrauchen'. Hierzu zählen vor allem *bcrypt* und *scrypt*, die gerne auch als Grundlage für *Password-Based Key Derivation Functions* genutzt werden.

Hashwert = Index

Der große Nutzen von Hashwerten besteht aber darin, dass man Daten hiermit *indizieren* kann, wobei der Index eine konstante Länge aufweist. Der Index ist somit eindeutiger Repräsentant der Daten und mit diesem über die Hashfunktion h verknüpft. Dies wird speziell für *Digitale Signaturen* ausgenutzt, denen wir uns im weiteren zuwenden wollen.

Eine weitere populäre Nutzung von Hashfunktionen liegt in Form der *Blockchain* und hier speziell der Erzeugung von Krypotwährungen wie Bitcoin vor, wo deren Eigenschaft das mathematische Fundament des *Mining* darstellt:

Hashwerte bei Bitcoins

> Die Nicht-Umkehrbarkeit von Hashfunktionen spielt aktuell bei der Berechnung von Bitcoins ₿ eine zentrale Rolle: Bitcoins werden 'gemined', indem gefordert wird, dass der SHA-256-Hashwert immer weiter gegen Null tendiert: '`000000000019d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f`'. Der hier hexadezimal dargestellte Wert ist im Genesis-Block der Bitcoin-Blockchain vorhanden. Wenn alle 256 Bit des Hashwerts auf Null gesetzt sind bzw. wenn der Inputwert y bekannt ist, für den diese Bedingung gilt, also SHA-256(y) = '`0000 ... 0000`', sind alle Bitcoins erschöpft. Die 'einfache' Berechenbarkeit von SHA-256 macht Bitcoin-Mining attraktiv, speziell auf geeigneter Hardware; allerdings mit dem negativen Seiteneffekt, dass enorm viel elektrische Energie für sinnlos generierte Hashwerte benötigt wird.

2.2.4 Signatur-Primitiv σ

Beim Signatur-Primitiv σ drehen wir den Verwendungszweck des *public key* p und des *private key* $\bar{p}$ um und gehen davon aus, dass sie dem Unterzeichner (*Originator*) zugewiesen sind – also p_o und $\bar{p}_o$ –, der ein Dokument x signieren (ausgedrückt durch den Wert sig) möchte:

- *Signaturoperation* des Unterzeichners: $sig = \sigma(x, \bar{p}_o)$
- *Verifikationsoperation* beim Empfänger: $x = \bar{\sigma}(sig, p_o)$

Message Digest

Der *public key* des Unterzeichners p_o ist öffentlich bekannt bzw. wird zusammen mit dem Dokument x und der Signatur sig dem Empfänger zur Verfügung gestellt. Aus praktischen Gründen ersetzen wir allerdings das Dokument x durch dessen Hashwert: $h = h(x)$, was als *Message Digest* bezeichnet wird. Die Signatur ist wiederum *zustandslos*.

Üblicherweise wird daher ein Zeitstempel t zur Signatur hinzugefügt [Abb. 2.2-2]: $sig|_t = \sigma(h(x), t, \bar{p}_o)$. Hierdurch wird die Signatur auch transaktionssicher.

Ziele digitaler Signaturen

Laut *Bruce Schneier* [Sch96] lassen sich mit digitalen Signaturen folgende Ziele realisieren:

1. *Verifiable*: Der Empfänger kann einfach überprüfen, dass die Nachricht tatsächlich vom Unterzeichner stammt.
2. *Un-Forgeable*: Nur der Unterzeichner kann die Signatur dem Dokument beifügen; diese kann somit nicht gefälscht werden.
3. *None-Reusable*: Die digitale Signatur ist für dieses Dokument einmalig und kann nicht für andere benutzt werden.
4. *Un-Alterable*: Eine Änderung des Dokuments nach Berechnung der Signatur zerstört die Bindung zwischen beiden: Fälschungssicherheit.
5. *None-Deniable*: Der Unterzeichner kann nicht bestreiten, dass er das Dokument signiert hat.

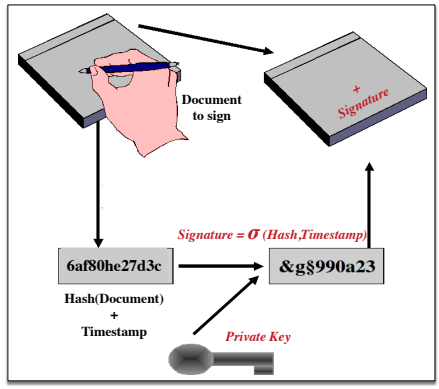


Abb. 2.2-2: Erzeugung einer digitalen Signatur

2.2.5 Zusammenspiel der Krypto-Primitive

Die gesamte moderne IT-Security macht sich die Krypto-Primitive

- Verschlüsselung $\mathbb{C}$,
- Schlüsseltausch κ,
- Signatur σ sowie
- Hashes h

zunutze und baut mit den unterschiedlichen Implementierungen entsprechende Krypto-Lösungen auf, wie Abb. 2.2-3 in Anlehnung an XKCD[12] zeigt.

Verschlüsselungs-Primitiv

Die Verschlüsselungs-Primitive $\mathbb{C}$ können wir grob in Block- und Stromschlüssel einordnen, worauf wir später noch im Detail eingehen werden. Allerdings werden bis dato bei starker Verschlüsselung nahezu ausschließlich Blockchiffren – wie AES – in einem verschränkten Betriebsmodus eingesetzt. Eine Mischform von Block- und Stromchiffren gilt bei richtiger Wahl der Komponenten als besonders sicher. Allerdings findet bei TLS [Abschnitt 7.2] für den Datentransport das Stromschlüssel-basierte Protokoll *ChaCha* [RFC 7905] zunehmend Einsatz.

[12] *A webcomic of romance, sarcasm, math, and language* von *Randall Munroe*, siehe: http://xkcd.com/

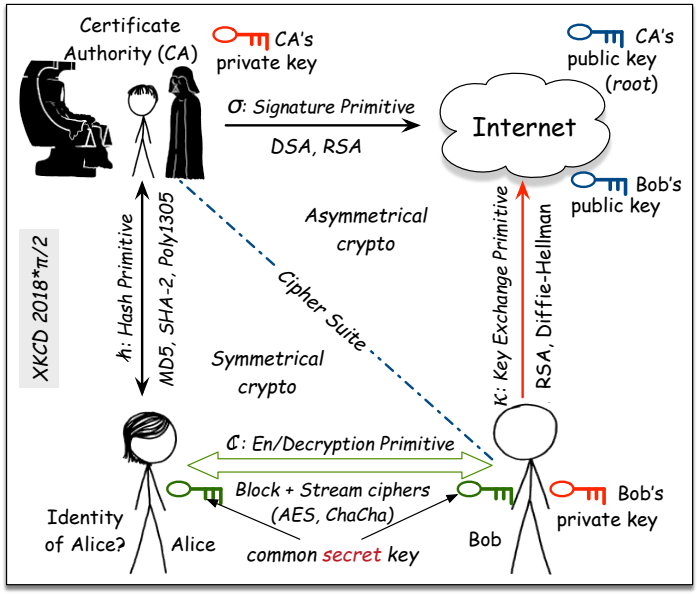


Abb. 2.2-3: Kryptographische Primitive und ihre Implementierung
AES: Advanced Encryption Standard, MD5: Message Digest 5,DSA: Digital Signature Algorithm, DSS: Digital Signatur Standard, RSA: Rivest/Shamir/Adleman Algorithmus, SHA: Secure Hash

Es besteht die Meinung, dass vor allem die Länge des Schlüssels die Qualität der Verschlüsselung bestimmt. Bei Blockchiffren sind die einzelnen Nachrichtenblöcke immer gleich lang wie der Schlüssel, was dazu führt, dass wir in einer Verschlüsselungsoperation immer genau n Bits der Blocklänge als auswertbare Nutzinformation mitgeben, d.h. die Verschlüsselung wird 'stärker', aber um den Preis von mehr Informationen, die automatisch mitgeliefert werden.

Schlüsseltausch-Primitiv

Die Schlüsseltausch-Operation κ basiert wie auch die Signatur-Operation σ auf den bekannten RSA- bzw. Diffie-Hellman-Algorithmen. Beide Verfahren arbeiten deterministisch, d.h. mit der Vollendung des Schlüsseltauschs $\bar{\kappa}$ ist klar, dass beide Partner über die gleichen Parameter verfügen. Zukünftige, z.B. Gitter-basierte Verfahren, operieren dagegen *probabilistisch* bzw. *optimistisch*:

> In den meisten Fällen stimmen die ausgehandelten Parameter überein, der Algorithmus garantiert dies aber nicht. Sollte die anschließende Verschlüsselung nicht klappen, muss der Schlüsseltausch erneut angestoßen werden.

Signatur-Primitiv

Im praktischen Einsatz des Signatur-Primitivs σ ist man auf eine tragende Infrastruktur angewiesen, wo von einem 'Trust-Anker' aus Vertrauen bezogen werden kann. Für die *Public Key Infrastructure* PKI und den *Certificate Authorities* ist dieses Vertrauen verloren gegangen, und so stehen Alternativen wie das dezentral organisierte *Domain Name System* DNS derzeit im Zentrum einer Neuausrichtung.

Hash-Primitiv

Das Hash-Primitiv h hat sich als 'Schweizer-Messer' der Kryptographie entwickelt, wird es doch sowohl zur Integritätsprüfung lang- und kurzfristiger gültiger Datenblöcke benutzt, wobei hier auf den Datenkontext im vorigen Abschnitt verwiesen werden kann. Der Hashwert liefert somit einen technischen Kontext zur Nutzinformation, der in manchen Fällen – wie unter Einsatz von TLS Abb. 2.3-1 – zwar 'Integrität' sichert, aber auch 'Data Leakage' mit sich bringen kann.

Hashfunktionen und Verschlüsselungsfunktionen zählen zu den 'klassischen/symmetrischen' Krypto-Verfahren. Deren Implementierung ist auch erstaunlich ähnlich und basiert auf den mathematischen Ideen der *Permutation*, *Transposition* und – wichtig – *Substitution*. Die letztere Operation macht das Ergebnis quasi 'unvorhersehbar' und abhängig von den Eingangswerten. Schlüsseltausch- und Signaturoperation fallen hingegen in den Bereich der 'asymmetrischen' Kryptographie [Abb. 2.2-3] und werden heute technisch vor allem mittels *elliptischer Kurven* unter Einsatz des *Diffie-Hellman-Protokolls* vorgenommen.

Keine Authentisierung von Alice

Bei den derzeitigen Krypto-Lösungen wird im Rahmen der Transaktion nur die Identität des Empfängers (also Bob; vgl. Abb. 2.2-3) verlangt; Alice handelt hingegen anonym, und die Berechtigungen, über die Alice verfügt, müssen in der Applikation geklärt werden.

> Will man das ändern, muss Alice ebenfalls über einen *public* und *private key* verfügen, was technisch möglich, praktisch jedoch mit einigem Aufwand verbunden ist. In den meisten Fällen begnügt man sich damit, Alice eine *lokale Identität* zuzuweisen. Globale Identitäten, die z.B. an Zertifikate im Personalausweis geknüpft sind, wurden zwar in den letzten Jahrzehnten entwickelt, haben sich aber immer noch nicht auf breiter Front durchgesetzt.

2.3 Hashfunktionen und ihr Einsatz

Die zentralen Eigenschaften von Hashfunktionen und Hashwerten wurden bereits bei der Diskussion des Hash-Primitivs h vorgestellt. Hierbei wurde eine Hashfunktion als Rechenvorschrift eingeführt, mit der eine 'Eingangs'-Zeichenfolge beliebiger Länge in eine 'Ausgangs'-Zeichenfolge konstanter (im Allgemeinen kürzerer) Länge umgewandelt wird: der *Hashwert* oder die *Hashsumme*. Diese 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, was eine 'Hashsumme' fundamental von einer '*Checksumme*' unterscheidet.

Ein weiteres wichtiges Merkmal, was Hashsummen von Checksummen unterscheidet, ist ihre Nicht-Additivität: Wenn wir zu einem bekannten Wert x, für den $h(x)$ gebildet werden soll, einen weiteren Wert k hinzufügen, sagen wir in Form einer Konkatenierung $h = h(k \parallel x)$, benötigt man das Wissen sowohl von x als auch von k, um den Hashwert h zu berechnen.

4 Transportprotokolle TCP, UDP und SCTP

Notwendigkeit der Transportprotokolle

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

Unterschiede zwischen UDP, TCP und SCTP

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

Überblick über das Kapitel

Nach einer kurzen Erläuterung der Aufgaben der Transportschicht in Abschnitt 4.1 beschreibt Abschnitt 4.2 das Konzept und den Einsatz von UDP. Die an die Anforderungen der Echtzeitkommunikation angepasste und neue Version von UDP wird als *UDP-Lite* bezeichnet, worauf wir ebenfalls eingehen. Die Funktionen von TCP, insbesondere die Fehlerkontrolle und Flusskontrolle, erläutert Abschnitt 4.3 und die Implementierungsaspekte von TCP präsentiert Abschnitt 4.4. Die Überlastkontrolle bei TCP nach dem Prinzip ECN (*Explicit Congestion Notification*) stellt Abschnitt 4.5 dar. Auf das Konzept und den Einsatz von SCTP geht Abschnitt 4.6 ein.

Ziel dieses Kapitels

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

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

4.1 Grundlagen der Transportprotokolle

Wozu Transportschicht?

Wir greifen die bereits in Abschnitt 1.4.2 gezeigte Darstellung des Transports von IP-Paketen auf und betrachten die ersten drei Schichten in einem IP-Netz als *IP-Übermittlungsdienst*. Da dieser Dienst über keine Mechanismen verfügt, um u.a. Verluste der übertragenen Daten zu entdecken und zu veranlassen, dass der Quellrechner die Übermittlung wiederholt, ist die Transportschicht nötig. Damit können bestimmte Mechanismen für die Garantie der zuverlässigen Datenübermittlung zur Verfügung gestellt werden. Abb. 4.1-1 illustriert die Bedeutung der Transportschicht in IP-Netzen und die Aufgabe ihrer Protokolle.

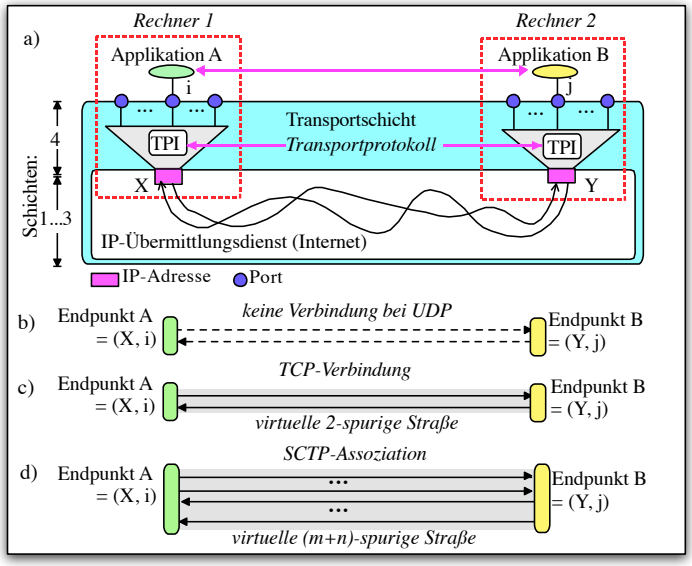


Abb. 4.1-1: Bedeutung der Transportschicht in IP-Netzen und ihre Protokolle: a) Logische Interpretation der Transportschicht, b) Kommunikation mit UDP, c) Interpretation einer TCP-Verbindung, d) Interpretation einer SCTP-Assoziation
TPI: Transportprotokollinstanz; X, Y: IP-Adressen; i, j: Ports

Sockets

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

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

Transportprotokolle

Die wichtigsten Transportprotokolle in IP-Netzen sind:

- **UDP** (*User Datagram Protocol*),

- **TCP** (*Transmission Control Protocol*) und
- **SCTP** (*Stream Control Transmission Protocol*).

UDP

UDP ist ein verbindungsloses und unzuverlässiges Transportprotokoll, nach dem hauptsächlich voneinander unabhängige Nachrichten zwischen den kommunizierenden Rechnern ausgetauscht werden, ohne dass eine explizite Vereinbarung hinsichtlich des Ablaufs der Übertragung vorgenommen wird. Es wird daher keine virtuelle (logische) Verbindung zwischen den Kommunikationsinstanzen aufgebaut [Abb. 4.1-1b]. UDP wird in Abschnitt 4.2 näher dargestellt.

TCP

TCP ist ein verbindungsorientiertes und zuverlässiges Transportprotokoll. Vor dem eigentlichen Datenaustausch werden Vereinbarungen hinsichtlich des Verlaufs der Kommunikation zwischen den Rechnern getroffen. Zentraler Bestandteil der Vereinbarung sind der Aufbau, die Überwachung der Korrektheit der übertragenen Daten (Byte) und der Abbau einer virtuellen Verbindung, d.h. einer *TCP-Verbindung* [Abb. 4.1-1c], die unabhängig (d.h. vollduplex) für beide Kommunikationsrichtungen erfolgt.

SCTP

SCTP ist ein neues verbindungsorientiertes Transportprotokoll. Im Gegensatz zu TCP kann sich eine *SCTP-Verbindung*, die auch *SCTP-Assoziation* genannt wird, aus einer Vielzahl von entgegen gerichteten Kommunikationspfaden zusammensetzen [Abb. 4.1-1d]. SCTP wird in Abschnitt 4.5 detaillierter dargestellt.

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

Mehrere IP-Adressen

Ferner ist zu bedenken, dass alle aktuellen TCP/IP-Implementierungen mehrere IP-Adressen bereitstellen, und zwar die IPv4-Adressen `0.0.0.0` für die IP-Instanz (d.h. den Rechner) selbst [vgl. Tab. 3.3-2], `127.0.0.1`] für das erste Netzwerk-Interface und die jeweils zugewiesenen privaten oder offiziellen IP-Adressen. Unterstützt der Rechner gleichzeitig IPv6, kommen die entsprechenden IPv6-Adressen hinzu.

Da die Portnummer als Angabe im TCP-, im UDP- und im SCTP-Header 16 Bit lang ist [Abb. 4.2-1, Abb. 4.3-2 und Abb. 4.6-3], kann ein Rechner pro verfügbare IP-Adresse gleichzeitig bis zu 65535 Ports organisieren. Die Ports mit den Nummern 0 bis 1023 können in der Regel nur von privilegierten, systemnahen Anwendungen benutzt werden.

Well-known Ports

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

Freie Ports

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

Registered Ports und IANA

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

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

4.2 Konzept und Einsatz von UDP

Besonderheiten von UDP

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

Einsatz von UDP

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

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

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

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

Neben dem Protokoll RADIUS [Abschnitt 15.2] nutzen ferner die Anwendungen TFTP (*Trivial File Transfer Protocol*) und RPC/NFS (*Remote Procedure Call* beim *Network File System*) UDP als Transportprotokoll.

4.2.1 Aufbau von UDP-Paketen

Den Aufbau von UDP-Paketen zeigt Abb. 4.2-1 mit folgenden Angaben im UDP-Header:

- `Source Port` (Quellport)
  Angabe der Nummer des Ports als Kommunikationspuffer der Applikation im Quellrechner [Abb. 4.1-1a].
- `Destination Port` (Zielport)
  Die Nummer des Ports als Identifikation der Applikation im Zielrechner.

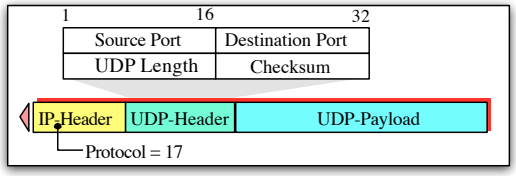


Abb. 4.2-1: Aufbau von UDP-Paketen

- `UDP Length` (UDP-Paketlänge)
  Hier wird die Länge des UDP-Pakets angegeben.
- `Checksum` (Prüfsumme)
  Diese Prüfsumme ermöglicht, sowohl im UDP-Paket als auch in einigen Angaben im IP-Header, die den *IP-Pseudo-Header* bilden, Übertragungsfehler zu entdecken [Abb. 4.2-2].

Mit der Angabe `Source Port` und `Destination Port` im UDP-Header und der Angabe von `Source IP Address` und `Destination IP Address` im IP-Header werden die beiden Endpunkte (d.h. die Sockets) festgelegt, zwischen denen der Datenaustausch stattfindet [Abb. 4.1-1b].

UDP-Prüfsumme

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

IP-Pseudo-Header

Die Prüfsumme im UDP-Header deckt auch einige Angaben im IP-Header ab, die den *IP-Pseudo-Header* bilden. Abb. 4.2-2 zeigt dies.

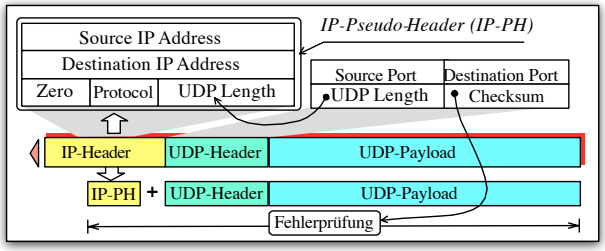


Abb. 4.2-2: Prüfsumme im UDP-Header und Angaben im IP-Pseudo-Header (IP-PH)

Die Angaben im IP-Pseudo-Header sind:

- `Source IP Address` (*IP-Quelladresse*), d.h. Quelladresse aus dem IP-Header,
- `Destination IP Address` (*IP-Zieladresse*), d.h. Zieladresse aus dem IP-Header,
- `Zero`: Der Wert `0` wird hier eingetragen.
- `Protocol`: Es wird die Protokollnummer 17 von UDP angegeben [Tab. 3.1-1].
- `UDP Length` (UDP-Paketlänge): Die Länge des UDP-Pakets wird mitgeteilt. Dieser Wert ist aus dem Feld `UDP Length` im UDP-Header übernommen.

Die Konstruktion des IP-Pseudo-Header kann nur als historisch betrachtet werden. Einerseits verstößt sie elementar gegen das Schichtenprinzip für den Kommunikationsablauf, andererseits kann sie keines der angestrebten Ziele wie beispielsweise

Identifikation von 'fremden' UDP-Paketen in einer Kommunikationsbeziehung garantieren. Daher wird dieses Konstrukt bereits bei UDP-Lite aufgegeben.

UDP-Paketgröße 512 Byte

Eine zentrale Einschränkung bei der Kommunikation über UDP besteht in der Notwendigkeit, die maximale Größe der UDP-Pakete auf 512 Byte zu beschränken. Diese Anforderung entstammt nicht der Definition von UDP, sondern hat einen praktischen Hintergrund: RFC 1122 verlangt bei der IP-Kommunikation als minimale Größe des Datenpuffers 576 Byte. Zieht man den (minimal) 20 Byte großen IP-Header ab, bleiben für das UDP-Paket 556 Byte und abzüglich des UDP-Header (8 Byte) noch maximal 548 Byte für Nutzdaten. Diese historisch festgelegte Größe, die Unabhängigkeit einzelner UDP-Pakete voneinander sowie das Fehlen einer Flusskontrolle (wie bei TCP) führen dazu, dass UDP-Pakete in der Regel auf eine Größe von 512 Byte beschränkt sind.

4.2.2 Protokoll UDP-Lite

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

Wozu UDP-Lite?

Um die empfangenen UDP-Pakete mit einzelnen Bitfehlern in Audio- bzw. in Videodaten nicht verwerfen zu müssen und damit die Häufigkeit der Verluste von UDP-Paketen mit Audio- und Videodaten zu reduzieren, war eine Modifikation von UDP nötig. Die UDP-Fehlerkontrolle wurde so verändert, dass nur die Bereiche in UDP-Paketen überprüft werden, in denen Übertragungsbitfehler eine große Auswirkung auf die Qualität der Kommunikation haben. Dies hat zur Entstehung von UDP-Lite geführt [RFC 3828], dessen Paketaufbau in Abb. 4.2-3 gezeigt ist.

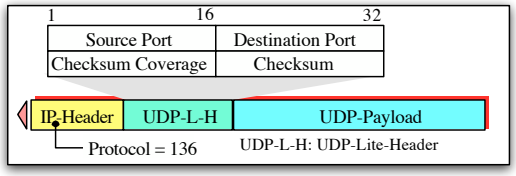


Abb. 4.2-3: Aufbau von UDP-Lite-Paketen

Der Header von UDP-Lite hat fast die gleiche Struktur wie der von UDP [Abb. 4.2-1]. Im Unterschied zu UDP enthält der Header von UDP-Lite `Checksum Coverage` anstelle der Angabe `UDP Length`. Die Nummern der *Well-known Ports* sind bei UDP-Lite die gleichen wie bei UDP. Damit können sowohl UDP als auch UDP-Lite die gleichen Applikationen nutzen.

Idee von UDP-Lite

Die Idee von UDP-Lite besteht darin, dass im Header mit `Checksum Coverage` angegeben werden kann, welcher Teil der UDP-Payload bei der Fehlerprüfung berücksichtigt wird. Abb. 4.2-4 illustriert die Idee von UDP-Lite.

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

Falls als UDP-Payload ein RTP-Paket übermittelt wird, kann mit `Checksum Coverage` angegeben werden, dass die Fehlerprüfung beispielsweise nur im RTP-Header stattfinden soll. Daher deckt die Prüfsumme in diesem Fall nur den IP-Pseudo-Header, den UDP-Header und den RTP-Header ab.

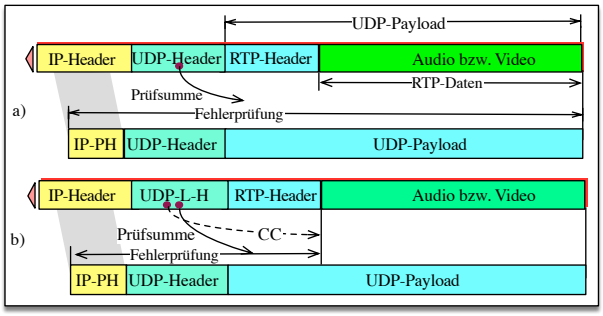


Abb. 4.2-4: Überprüfung von Übertragungsbitfehlern: a) bei UDP, b) bei UDP-Lite
CC: Checksum Coverage, IP-PH: IP-Pseudo-Header, UDP-L-H: UDPLite Header, RTP: Real-time Transport Protocol

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

Checksum Coverage

Vergleicht man Abb. 4.2-2 und Abb. 4.2-4, so ist ersichtlich, dass die Angabe `UDP Length` im UDP-Header durch `Checksum Coverage` bei UDP-Lite ersetzt wurde. Dies ist möglich, da die Angabe `UDP Length` im UDP-Header redundant ist und die Länge des UDP-Pakets aus den Angaben im IP-Header (d.h. `UDP Length = Total Length minus Header Length`) berechnet werden kann.

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

4.3 Funktion des Protokolls TCP

TCP ist ein verbindungsorientiertes, zuverlässiges Transportprotokoll, das bereits in RFC 793 spezifiziert wurde und seither ständig weiterentwickelt wird. Seine Bedeutung kommt in Abb. 4.3-1 zum Ausdruck. Überdies wird hier verdeutlicht, dass

mittels TCP mehrere Applikationen auf die Dienste von IP gleichzeitig zugreifen können. Daher kann TCP auch als *logischer Multiplexer* von Applikationen angesehen werden.

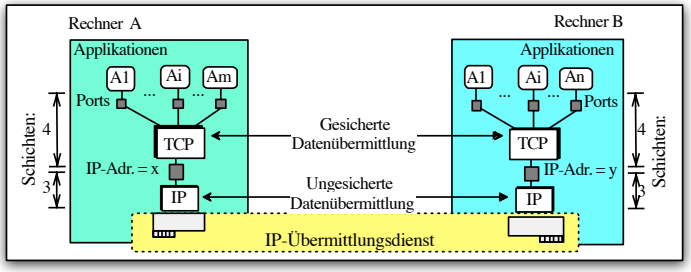


Abb. 4.3-1: TCP als Sicherungsprotokoll zwischen zwei entfernten Rechnern

TCP als Kontrollprotokoll

Im Gegensatz zum UDP-Paket – das als reiner Datencontainer aufgefasst werden kann – beinhalten TCP-Pakete umfangreiche Steuerungsinformationen, die zum geregelten Ablauf der Verbindung eingesetzt werden, aber teilweise auch für die auf TCP aufbauenden Applikationen genutzt werden können. Die Steuerungsinformationen werden bei Bedarf im TCP-Header im Optionsfeld (`Options`) untergebracht. TCP hat sich im Laufe seiner Entwicklung durch den Einsatz neuer sowie die Umdefinition bestehender Optionen wie ein Chamäleon an veränderte Gegebenheiten angepasst.

TCP-Verbindungen

Die Zuverlässigkeit von TCP wird dadurch gewährleistet, indem sich die TCP-Instanzen gegenseitig über ihren jeweiligen Zustand (*Status*) informieren, also eine *virtuelle Verbindung* unterhalten. Um die Datenübermittlung nach TCP zu gewährleisten, muss daher zunächst eine *TCP-Verbindung* aufgebaut werden. Wenn der Nutzdatenfluss ausschließlich von *A* nach *B* geht, existiert immer auch eine Kontrollverbindung von *B* nach *A*. Typischerweise hat aber auch die Gegenseite etwas mitzuteilen, wodurch auch Nutzdaten von *B* nach *A* übertragen werden. In diesem Fall werden die Kontrollinformationen in den (beidseitigen) Nutzpaketen untergebracht. TCP realisiert eine *Vollduplex-Verbindung* zwischen den Kommunikationspartnern mit einem ausgeklügelten und effizienten *In-Band-Steuerungsverfahren* [Abb. 4.1-1]. Im Vergleich hierzu stellt beispielsweise ICMP für IPv4 [Abschnitt 3.7] einen Out-of-Band-Kontroll- und Fehlermechanismus bereit; bei UDP existiert hingegen gar keiner.

4.3.1 Aufbau von TCP-Paketen

Arten von TCP-Paketen

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

- TCP-Pakete mit Nutzdaten – als *TCP-Nutzlast* – und mit Steuerungsinformationen im TCP-Header [Abb. 1.4-2]. Zusätzlich zur eigentlichen Übermittlung von Nutzdaten sind Angaben zur Realisierung der Flusskontrolle nach dem *Sliding-Window-Prinzip* [Abb. 4.4-2] im TCP-Header dieser Pakete enthalten.

- TCP-Pakete ausschließlich mit Steuerungsinformationen. Deren Aufgabe ist das Regeln des Auf- und Abbaus von TCP-Verbindungen sowie die Überprüfung ihrer Fortdauer in Form von *Keep-Alives*, falls längere Zeit keine Datenübermittlung erfolgte.

Nummerierung von Byte

Ist eine TCP-Verbindung aufgebaut, besteht die zentrale Aufgabe von TCP darin, die zu übertragenden Daten bzw. Nachrichten vom Quellrechner *byteweise zu nummerieren* und diese in Form von *Datensegmenten* zu übermitteln. Die Nummer des ersten Nutzbyte im TCP-Datencontainer ist die laufende *Sequenznummer* und bezeichnet somit die Anzahl aller gesendeten Nutzdatenbyte der vorigen Segmente (ohne Wiederholungen); eine Methode, die als *Byte-stream* bezeichnet wird [Abb. 4.3-8].

Flusskontrolle

Die Partnerinstanz quittiert die erhaltenen Daten, indem sie als *Quittung* (*Acknowledgement*) den letzten Wert der *Sequenznummer* (`SEQ`) plus 1 – also `SEQ + 1` – zurückgibt [Abb.4.3-8]. Damit dieser Mechanismus funktioniert, wird vor Beginn einer Übermittlung zwischen den TCP-Instanzen im Quell- und im Zielrechner entsprechend der anliegenden Datenmenge die maximale Segmentgröße vereinbart (z.B. 1000 Byte). Diese und weitere Angaben werden im *TCP-Header* eingetragen.

Garantie der Reihenfolge

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

TCP-Header

Abb. 4.3-2 zeigt den Aufbau des TCP-Header.

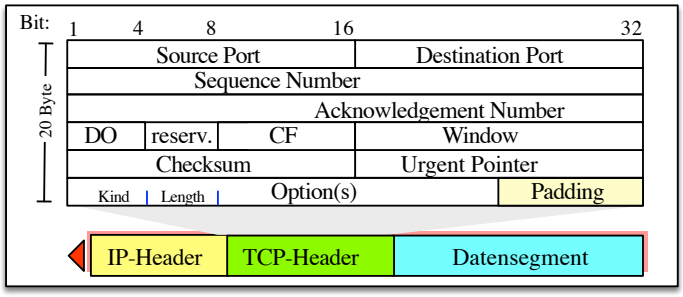


Abb. 4.3-2: Aufbau des TCP-Header
CF: Control Flags, DO: Data Offset

Die einzelnen Angaben im TCP-Header haben folgende Bedeutung:

- `Source Port` (Quellport)
  Hier wird die Nummer des Ports der Applikation im Quellrechner angegeben, die die TCP-Verbindung initialisiert hat.
- `Destination Port` (Zielport)
  Hier wird die Nummer des Ports dieser Applikation im Zielrechner angegeben, an die die Daten adressiert sind.
- `Sequence Number` (Sequenznummer)
  Die Sequenznummer bezieht sich auf den Sender und dient diesem zur Byte-weisen Numme-

Bedeutung von ISN

rierung der gesendeten Daten. Beim Aufbau der TCP-Verbindung generiert jede TCP-Instanz eine Anfangssequenznummer `ISN` (*Initial Sequence Number*). Diese Nummern werden ausgetauscht und gegenseitig bestätigt [Abb. 4.3-4]. Um die gesendeten TCP-Pakete eindeutig am Zielrechner zu identifizieren, muss der Fall ausgeschlossen werden, dass sich zu einem Zeitpunkt im Netz mehrere Segmente mit der gleichen Sequenznummer befinden. Aus diesem Grund darf sich die Sequenznummer innerhalb der festgelegten *Lebenszeit* (*Time to Live*) für die IP-Pakete nicht wiederholen [Abb. 2.2-1]. Im Quellrechner wird die Sequenznummer immer jeweils um die Anzahl bereits gesendeter Byte erhöht.
Die `ISN` wird in der Regel nicht von 0 hochgezählt, sondern bei aktuellen TCP-Implementierungen zufällig erzeugt. Hierdurch kann in gewissem Umfang gewährleistet werden, dass ein Kommunikationsendpunkt zum Zeitpunkt der Initialisierung über eine weitgehend einmalige ISN verfügt, die somit neben dem Socket (*IP-Adresse, Portnummer*) ein zusätzliches Verbindungsmerkmal darstellt.

- `Acknowledgement Number` (Quittungsnummer)
  Die Quittungsnummer wird vom Empfänger vergeben und dient zur Bestätigung der empfangenen Daten, indem dem Quellrechner mitgeteilt wird, bis zu welchem Byte (*Sequenznummer*) die Daten korrekt empfangen wurden.
- `Data Offset` (Datenabstand)
  Das vier Bit große Feld gibt die Länge des TCP-Header in 32-Bit-Worten an und damit die Stelle, ab der die Daten beginnen. Dieses Feld kann auch als `Header Length` interpretiert werden.
- Reserved (Reserviert)
  Diese drei Bit sind reserviert und üblicherweise auf `0` gesetzt.
- `Control Flags` (Kontrollflags)
  Die Kontrollflags legen fest, welche Felder im Header gültig sind und dienen zur Verbindungssteuerung. Zurzeit sind 9 Bit vorgesehen. Ist das entsprechende Bit gesetzt, gilt Folgendes:
  - `NS` (*ECN-nonce Concealment Protection*) [RFC 3540]: Unterstützung des ECN-Verfahrens [Abschnitt 4.5] mit einem Nonce-Bit.
  - `CWR` (*Congestion Window Reduced*): Einsatz bei der Überlastkontrolle nach `ECN` (*Explicit Congestion Notification*) [RFC 3168].
  - `ECE` (*ECN-Echo*): Einsatz bei der Überlastkontrolle nach ECN [RFC 3168].
  - `URG`: Der *Urgent Pointer* (Zeiger im Urgent-Feld) ist vorhanden und gültig.
  - `ACK`: Die *Quittungsnummer* ist vorhanden und gültig.
  - `PSH` (*Push-Funktion*): Die Daten sollen sofort an die nächsthöhere Schicht weitergegeben werden, ohne nächste Segmente abzuwarten. UNIX-Implementierungen senden `PSH` immer dann, wenn sie mit dem aktuellen Segment gleichzeitig alle Daten im Sendepuffer übergeben.
  - `RST` (*Reset*): Die TCP-Verbindung soll zurückgesetzt werden.
  - `SYN`: Aufbau einer TCP-Verbindung und ist mit einem `ACK` zu quittieren [Abb. 4.3-4].
  - `FIN`: Einseitiger Abbau einer TCP-Verbindung und das Ende des Datenstroms aus dieser Richtung, das mit einem `ACK` quittiert werden muss [Abb. 4.3-5].

Window für Flusskontrolle

- `Window` (Fenstergröße)
  Diese Angabe dient der Flusskontrolle nach dem Fenstermechanismus [Abb.4.3-6]. Das Feld `Window` gibt die Fenstergröße an, d.h. wie viele Byte – beginnend ab der Quittungsnummer – der Zielrechner in seinem Aufnahmepuffer noch aufnehmen kann. Empfängt der Quellrech-

ner eine TCP-PDU mit der Fenstergröße gleich 0, muss der Sendevorgang gestoppt werden. Wie die Fenstergröße die Effizienz der Übermittlung beeinflussen kann, ist aus Abb. 4.4-1b ersichtlich (Senderblockade).

- `Checksum` (Prüfsumme)
  Diese Prüfsumme erlaubt es, den TCP-Header, die Daten und einen Auszug aus dem IP-Header (u.a. Quell- und IP-Zieladresse), der an die TCP-Instanz zusammen mit den Daten übergeben wird, auf das Vorhandensein von Bitfehlern zu überprüfen. Bei Berechnung der Prüfsumme wird dieses Feld selbst als null angenommen.
- `Urgent Pointer` (Urgent-Zeiger)
  TCP ermöglicht es, wichtige (dringliche) und meist kurze Nachrichten (z.B. Interrupts) den gesendeten normalen Daten hinzuzufügen und an Kommunikationspartner direkt zu übertragen. Damit können außergewöhnliche Zustände signalisiert werden. Derartige Daten werden als *Urgent-Daten* bezeichnet. Ist der Urgent-Zeiger gültig, d.h. `URG = 1`, so zeigt er auf das Ende von Urgent-Daten im Segment. Diese werden immer direkt im Anschluss an den TCP-Header übertragen. Erst danach folgen normale Daten.
- `Options`
  TCP erlaubt es, `Service-Optionen` anzugeben. Das erste Byte in jeder `Option` legt den *Optionstyp* (*Kind*) fest, das zweite Byte die Länge (*Length*) der entsprechenden Option. Hierdurch können mehrere Optionen zugleich angegeben werden. Der experimentelle RFC 6994 geht noch einen Schritt weiter und erlaubt in den nächsten 2 Byte die Angabe einer *Experimental-ID* (`ExID`).

  Das `Options`-Feld ist entsprechend Tab. 4.3-1 zu interpretieren. In den RFCs 1323 und 2018 wurde die Bedeutung des `Options`-Feldes folgendermaßen festgelegt:

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

Tab. 4.3-1: Verwendung möglicher Optionen im `Options`-Feld des TCP-Headers `http://www.iana.org/assignments/tcp-parameters`; *) sind *historische* Optionen, die nicht mehr genutzt werden

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

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

- ▷ `Window Scale` (`WSopt`): Mit `WSopt` können die Kommunikationspartner während des Aufbaus einer TCP-Verbindung (also im `SYN`-Paket) festlegen, ob die Größe des 16 Bit `Window` um einen konstanten Skalenfaktor multipliziert wird. Dieser Wert kann unab-

hängig für den Empfang und das Versenden von Daten ausgehandelt werden. Als Folge dessen wird nun die Fenstergröße von der TCP-Instanz nicht mehr als 16 Bit-, sondern als 32 Bit-Wert aufgefasst. Der maximale Wert für den Skalenfaktor von `WSopt` beträgt 14, was einer neuen oberen Grenze für `Window` von 1 GByte entspricht.

- ▷ `Timestamps Option` (`TSopt`):
  Dieses Feld besteht aus den Teilen `Timestamp Wert` (`TSval`) und `Timestamp Echo Reply` (`TSecr`), die jeweils eine Länge von 4 Byte aufweisen. Eingetragen wird hier der *Tickmark*, ein interner Zähler, der pro Millisekunde um eins erhöht wird. `Timestamps Option` kann nur im ersten `SYN`-Paket angezeigt werden, das Feld ist nur bei `ACK`-PDUs erlaubt. Genutzt wird `TSopt` zur besseren Abschätzung der `RTT` (*Round Trip Time*) und zur Realisierung von `PAWS` (*Protect Against Wrapped Sequences*) [RFC 1323].
- ▷ Mit RFC 2018 wurde das Verfahren *Selective Acknowledgement* (`SACK`) eingeführt, das sich wesentlich auf das Optionsfeld `SACK` stützt und es zulässt, dieses Feld variabel zu erweitern. Auf dieses Verfahren wird später näher eingegangen.
- ▷ Ergänzt werden die Optionen um `Connection Count` (`CC`) sowie `CC.NEW` und `CC.ECHO`, die bei der Implementierung von *T/TCP* anzutreffen sind [Abschnitt 4.4.4].
- ▷ Bei Verwendung von *Multipath TCP* [Abschnitt 7.5] werden noch weitere TCP-`Options` genutzt [Abb. 7.5-6].

- `Padding` (Füllzeichen)
  Die Füllzeichen ergänzen die Optionsangaben auf die Länge von 32 Bit.

TCP Timeouts

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

TCP-Window

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

4.3.2 Konzept der TCP-Verbindungen

Three-Way Handshake

Eine TCP-Verbindung wird mit dem Ziel aufgebaut, einen zuverlässigen Datenaustausch zwischen den kommunizierenden Anwendungsprozessen in entfernten Rechnern zu gewährleisten. Die TCP-Verbindungen sind vollduplex. Man kann eine TCP-Verbindung als ein Paar von gegenseitig gerichteten unidirektionalen Verbindungen zwischen zwei Sockets interpretieren [Abb. 4.1-1c]. Der Aufbau einer TCP-Verbindung erfolgt immer mittels des *Three-Way Handshake* (*3WHS*)-Verfahrens, das

für eine Synchronisation der Kommunikationspartner sorgt und gewährleistet, dass die TCP-Verbindung in jede Richtung korrekt initialisiert wird. An dieser Stelle ist hervorzuheben, dass die Applikation im Quellrechner mit TCP über einen wahlfreien Port kommuniziert, der dynamisch zugewiesen wird.

TCP-Zustandsmodell

Das TCP-Modell geht von einer *Zustandsmaschine* aus. Eine TCP-Instanz befindet sich immer in einem wohldefinierten Zustand. Die Hauptzustände sind `Listen` und (Verbindung-) `Established`. Zwischen diesen stabilen Zuständen gibt es gemäß Abb.4.3-3 eine Vielzahl zeitlich befristeter (Zwischen-)Zustände. Mittels der Kontroll-Flags `ACK`, `FIN`, `SYN` und ggf. auch `RST` wird zwischen den Kommunikationspartnern der Wechsel zu bzw. der Verbleib in einem Zustand signalisiert.

Fenstermechanismus

Um die Menge der auf einer TCP-Verbindung übertragenen Daten zwischen den kommunizierenden Rechnern entsprechend abzustimmen, was man als Flusskontrolle bezeichnet, kommt der *Fenstermechanismus* (Window) zum Einsatz. Zur effizienten Nutzung des Fenstermechanismus stehen zwei Parameter zur Verfügung, die zwischen den TCP-Instanzen im Verlauf der Kommunikation dynamisch angepasst werden. Es handelt sich hierbei um: `Windowsize` und `Maximum Segment Size`.

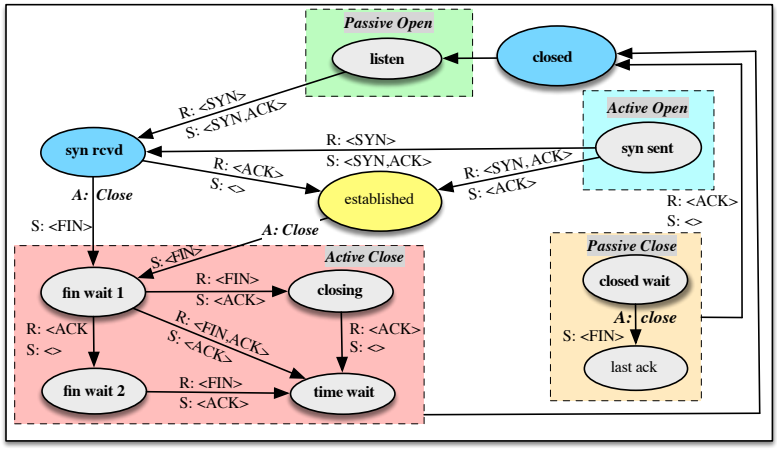


Abb. 4.3-3: TCP-Zustandsdiagramm und gestattete Übergänge zwischen den Zuständen
E: Empfänger, S: Sender, A: Applikation

Interpretation von Windowsize

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

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

Interpretation von Maximum Segment Size

`Maximum Segment Size` (`MSS`) stellt das Gegenstück zu `WSIZE` dar, ist also die Größe des TCP-Sendepuffers für die zu übertragenden Daten. Für übliche TCP-

Implementierungen gilt die Ungleichung `MSS < WSIZE`. Die dynamische Aushandlung dieser Parameter zusammen mit der Methode der Bestimmung der *Round Trip Time* (`RTT`) begründet ursächlich das gute Übertragungsverhalten von TCP in sehr unterschiedlichen Übermittlungsnetzen (siehe Abschnitt 4.4.2).

4.3.3 Auf- und Abbau von TCP-Verbindungen

Den Aufbau einer TCP-Verbindung zeigt Abb. 4.3-4. Hier soll insbesondere zum Ausdruck gebracht werden, dass eine TCP-Verbindung vollduplex ist und als ein Paar von zwei unidirektionalen logischen Verbindungen gesehen werden kann [Abb. 4.3-1c]. Die Kommunikationspartner befinden sich zum Anfang der Übertragung immer in folgenden Zuständen [Abb. 4.3-3]:

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

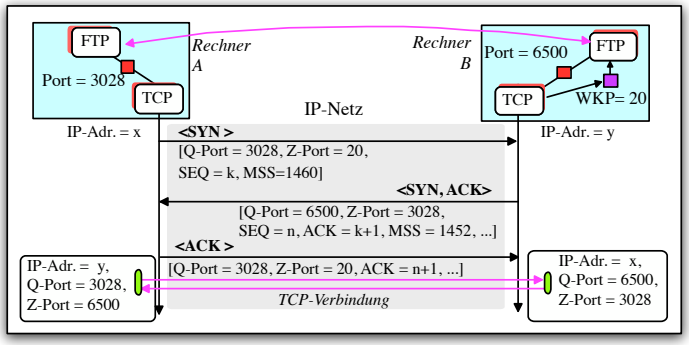


Abb. 4.3-4: Beispiel für den Aufbau einer TCP-Verbindung
Q: Quell, Z: Ziel, WKP: Well-known Port

Beispiel: FTP

In Abb. 4.3-4 wird die TCP-Verbindung im Rechner *A* mit der IP-Adresse `x` durch die Applikation FTP initiiert. Hierbei wird ihr für die Zwecke der Kommunikation beispielsweise die Portnummer 3028 zugewiesen. Die TCP-Instanz im Rechner *A* generiert ein TCP-Paket, in dem das Flag `SYN` gesetzt ist. Somit wird dieses Paket hier als `<SYN>`-Paket bezeichnet. Der Verbindungsaufbau beginnt damit, dass jeder der beiden Kommunikationspartner zunächst einen Anfangswert für die jeweilige Sequenznummer wählt. Dieser Anfangswert für eine Verbindung wird als *Initial Sequence Number* (`ISN`) bezeichnet.

<SYN>-Paket

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

- `SYN`-Flag im TCP-Header wird gesetzt. Damit ist der Name `<SYN>`-Paket zu begründen.

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

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

Reaktion des Zielrechners auf <SYN>-Paket

Die Ziel-TCP-Instanz befindet sich im *Listenmodus*, sodass sie auf ankommende `<SYN>`-Pakete wartet. Nach dem Empfang eines `<SYN>`-Pakets leitet die Ziel-TCP-Instanz ihrerseits den Verbindungswunsch an die Ziel-Applikation (hier FTP) gemäß der empfangenen Nummer des Zielports weiter. Falls die Applikation im Zielrechner die ankommende TCP-Verbindung akzeptiert, wird ihr ein Port als Puffer für zu sendende und zu empfangende Daten eingerichtet. Diesem Port wird eine große Nummer (z.B. 6500) zugeteilt. Danach wird eine eigene `ISN` in Richtung Rechner *A* generiert.

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

<SYN,ACK>-Paket

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

Reaktion auf <SYN,ACK>-Paket

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

`MSS` in <SYN>-Paketen

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

Aus Abb. 4.3-4 geht außerdem hervor, dass sich eine TCP-Verbindung aus zwei *unidirektionalen Verbindungen* zusammensetzt. Jede dieser gerichteten Verbindungen wird im Quellrechner durch die Angabe der IP-Zieladresse und der Quell- und Zielports eindeutig identifiziert.

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

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

Abbau einer TCP-Verbindung

Den Abbau einer TCP-Verbindung illustriert Abb. 4.3-5. Im Normalfall kann der Abbau von einer der beiden kommunizierenden Applikationen initiiert werden. Da jede TCP-Verbindung sich aus zwei gerichteten Verbindungen zusammensetzt, werden diese gerichteten Verbindungen quasi nacheinander abgebaut. Jede TCP-Instanz koordiniert den Abbau ihrer gerichteten Verbindung zu ihrer Partner-TCP-Instanz und verhindert hierbei den Verlust von übertragenen, aber noch nicht quittierten Daten.

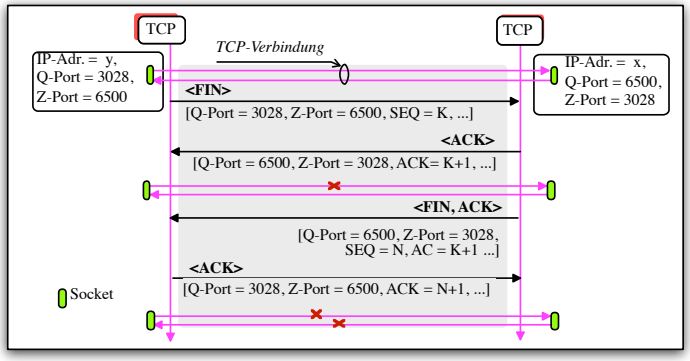


Abb. 4.3-5: Beispiel für den Abbau einer TCP-Verbindung

Maximum Segment Lifetime

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

Beim Abbau einer Verbindung tritt u.U. ein zusätzlicher interner *Time-out-Mechanismus* in Kraft. Die TCP-Instanz geht in den Zustand *Active-Close*, versendet ein abschließendes `<ACK>`-Paket und befindet sich dann im Status *Time-Wait* [Abb.4.3-3]. Dessen Zeitdauer beträgt `2 * MSL` (*Maximum Segment Lifetime*), bevor die TCP-Verbindung letztlich geschlossen wird. TCP-Pakete, die länger als die MSL-Dauer im Netz unterwegs sind, werden verworfen. Der Wert von *MSL* beträgt bei heutigen TCP-Implementierungen in der Regel 120 Sekunden. Anschließend wird der Port freigegeben und steht (mit einer neuen `ISN`) für spätere Verbindungen wieder zur Verfügung.

4.3.4 Flusskontrolle bei TCP

Sliding-Window-Prinzip

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

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

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

Sequenznummer

Mit der Sequenznummer `SEQ` werden die zu sendenden Daten fortlaufend byteweise nummeriert. Sie besagt, mit welcher Nummer die Nummerierung der im TCP-Paket gesendeten Byte beginnen soll [Abb. 4.3-8].

Quittungsnummer und `Window`

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

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

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

Flusskontrolle

Abb. 4.3-6 veranschaulicht die Flusskontrolle nach dem Sliding-Window-Prinzip mit der `Window-Größe = 4`. Wie hier ersichtlich ist, lässt sich das Window als Sendefenster interpretieren.

Sendeblockade

> Betrachten wir zunächst das Beispiel in Abb. 4.3-6a. Da die Window-Größe 4 beträgt, darf der Sender nur 4 Byte absenden, ohne auf eine Quittung warten zu müssen. Dies bedeutet, dass er die Byte mit den Nummern 1, 2, 3 und 4 absenden darf. Nach dem Absenden der ersten drei Byte ist eine Quittung eingetroffen, mit der das erste Byte quittiert wird. Dadurch verschiebt sich das Fenster (*Window*) mit den zulässigen Sequenznummern um eine Position nach rechts. Da maximal 4 Byte unterwegs sein dürfen, kann der Sender nun die nächsten Byte mit den Nummern 4 und 5 senden. Nach dem Absenden des Byte mit Sequenznummer 5 wird das Byte mit der Sequenznummer 2 durch den

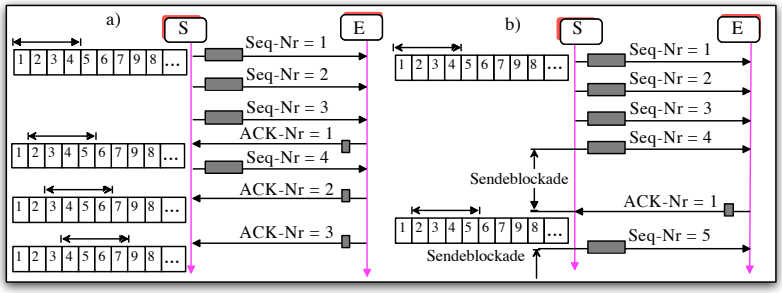


Abb. 4.3-6: Sliding-Window-Prinzip bei Window-Größe = 4:
a) fehlerfreie Übertragung, b) Bedeutung der Sende-Blockade
ACK: Quittungsnummer (Acknowledgement Number), Seq-Nr.: Sequenznummern;
E: Empfänger, S: Sender

Empfänger positiv quittiert. Dadurch verschiebt sich das Fenster nach rechts um eine Position weiter.

Abb. 4.3-6b zeigt die Situation, in der der Sendeprozess blockiert werden muss (*Sendeblockade*). Sie kommt oft dann vor, wenn einerseits die Verzögerungszeit im Netz groß und andererseits die Window-Größe zu klein ist. Mit großen Verzögerungszeiten ist immer zu rechnen, wenn eine Satellitenstrecke als ein Übertragungsabschnitt eingesetzt wird. Wie hier ersichtlich ist, muss der Sender nach dem Absenden von Byte mit den Sequenznummern 1, 2, 3 und 4 auf eine Quittung warten. Hier wurden die Daten aus dem Sendefenster abgesendet, und deren Empfang wurde noch nicht bestätigt. Bevor einige Byte quittiert werden, darf der Sender keine weiteren Byte senden. Nach dem Eintreffen der Quittung für das Byte mit Sequenznummer 1 verschiebt sich das Sendefenster um eine Position nach rechts. Das Byte mit der Sequenznummer 5 darf nun gesendet werden. Anschließend muss der Sendevorgang wiederum bis zum Eintreffen der nächsten Quittung blockiert werden.

4.3.5 TCP Sliding-Window-Prinzip

Bei TCP erfolgt die Flusskontrolle nach dem *Sliding-Window-Prinzip*. Hierbei legt die Window-Größe die maximale Anzahl von Byte fest, die der Quellrechner absenden darf, ohne auf eine Quittung vom Zielrechner warten zu müssen. Abb. 4.3-7 illustriert die Interpretation von Window bei TCP.

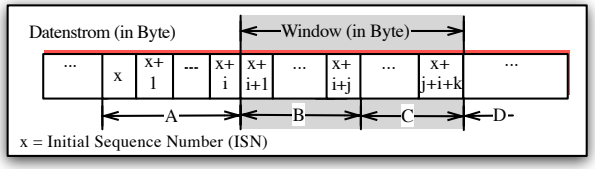


Abb. 4.3-7: Interpretation von Window bei TCP

Byte in Flight

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

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

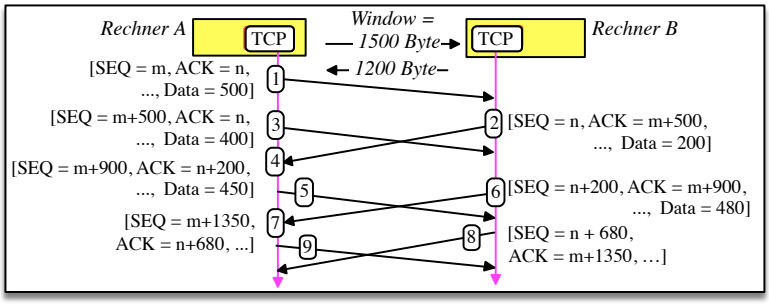


Abb. 4.3-8: Beispiel für den TCP-Ablauf bei fehlerfreier Datenübermittlung
SEQ: Sequenznummer, ACK: Quittungsnummer

Sliding Window mit fehlerfreier Datenübermittlung

Im Folgenden wird der Datenaustausch nach TCP verdeutlicht und damit auch das Sliding-Window-Prinzip näher erläutert. Abb. 4.3-8 zeigt ein Beispiel für eine fehlerfreie Datenübermittlung.

Fehlerfreier Datenaustausch

Die hier dargestellten einzelnen Ereignisse sind wie folgt zu interpretieren:

1. Das erste TCP-Paket von Rechner *A* zu Rechner *B* besitzt die Sequenznummer `SEQ = m`. Dieses Paket enthält die ersten 500 Datenbyte. `SEQ = m` verweist darauf, dass den einzelnen übertragenen Datenbyte die Nummern `m, m+1, ..., m+499` zugeordnet sind.
2. Von Rechner *B* wird mit einem TCP-Paket, in dem 200 Datenbyte enthalten sind und bei dem das `ACK`-Flag gesetzt wurde, bestätigt, dass das erste TCP-Paket von Rechner *A* fehlerfrei aufgenommen wurde und das nächste Datenbyte mit der Nummer `m+500` erwartet wird. `SEQ = n` besagt, dass die Nummern `n, n+1, ..., n+199` den hier übertragenen Datenbyte zuzuordnen sind.
3. Das zweite TCP-Paket von Rechner *A* zu Rechner *B* mit den nächsten 400 Datenbyte und mit `SEQ = m+500`. Somit enthält dieses TCP-Paket die Datenbyte mit den Nummern `m+500, m+501, ..., m+899`. Damit wurden bereits 900 Datenbyte von Rechner *A* abgeschickt und vom Zielrechner *B* noch nicht quittiert. Da die Window-Größe in Richtung zum Rechner *B* 1500 Byte beträgt, können vorerst nur noch `1500 - 900 = 600` Datenbyte abgesendet werden.
4. Nach dem Empfang dieses TCP-Pakets werden 500 Datenbyte vom Zielrechner *B* positiv quittiert. Somit verschiebt sich das Sendefenster im Rechner *A* um 500. Da der Empfang von 400 Byte (das zweite TCP-Paket) noch nicht bestätigt wurde, können noch `1500-400 = 1100` Datenbyte gesendet werden.
5. Das dritte TCP-Paket von Rechner *A* zu Rechner *B* mit `SEQ = m+900` und mit 450 Datenbyte. Dieses Paket enthält die Datenbyte mit den Nummern von `m+900` bis `m+1349`. Somit sind bereits 850 Datenbyte abgeschickt, die noch nicht quittiert

wurden. Darüber hinaus können nur noch weitere 650 (d.h. 1500 - 850) Datenbyte abgesendet werden. Mit diesem TCP-Paket wird dem Rechner *B* auch mitgeteilt, dass die nächsten Datenbyte ab Nummer n+200 erwartet werden.

6. Rechner *B* quittiert mit einem TCP-Paket, in dem das ACK-Flag gesetzt wird, das dritte TCP-Paket von Rechner A und sendet zu Rechner *A* die nächsten 480 Datenbyte. Diesen Datenbyte sind die Nummern n+200, ..., n+679 zuzuordnen.
7. Nach dem Empfang dieses TCP-Pakets werden die an Rechner *A* abgeschickten Datenbyte mit den Nummern m+900-1 positiv quittiert. Damit verschiebt sich in Rechner *A* das Sendefenster entsprechend.
8. Rechner *B* bestätigt mit einem TCP-Paket, in dem das ACK-Flag gesetzt wird, die Datenbyte einschließlich bis zur Nummer m+1350-1. Auf diese Weise wurden alle zu Rechner *B* abgeschickten Daten quittiert. Rechner *A* kann nun an Rechner *B* die durch die Window-Größe festgelegte Datenmenge (d.h. 1500 Byte) unmittelbar weitersenden, ohne vorher auf eine positive Quittung von Rechner *B* warten zu müssen.
9. Rechner *A* quittiert mit ACK = n+680 positiv Rechner *B*, alle Datenbyte bis zur Nummer n+680-1 empfangen zu haben. Zugleich wird mit SEQ = m+1350 signalisiert, dass bereits m+1350 Datenbyte geschickt wurden.

Sendeblockade bei der Datenübermittlung

Den Ablauf des Protokolls TCP bei einer fehlerbehafteten Datenübermittlung illustriert Abb. 4.3-9.

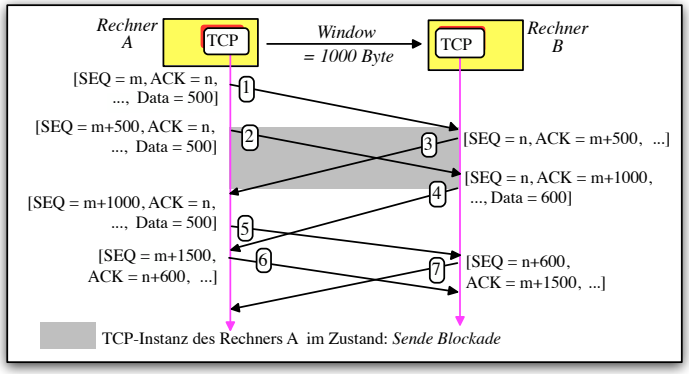


Abb. 4.3-9: Beispiel für das TCP-Verhalten bei einer fehlerbehafteten Datenübermittlung

Fehlerhaftete Datenübertragung

Die einzelnen Ereignisse sind hier folgendermaßen zu interpretieren:

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

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

Fehlerbehaftete Datenübermittlung

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

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

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

MSL Timer

Das Funktionsweise des MSL-Timer bei TCP zeigt Abb. 4.3-10. Um die Darstellung zu vereinfachen, werden hier nur jene Angaben gezeigt, die nötig sind, um dieses Prinzip zu erläutern.

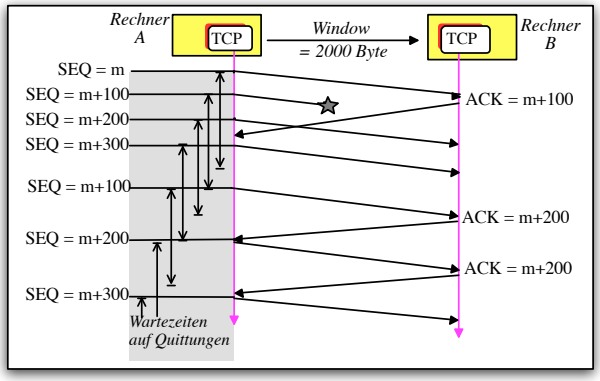


Abb. 4.3-10: Gesteuerte Segmentwiederholung über den TCP MLS Timer
ACK: Quittungsnummer, SEQ: Sequenznummer

Nutzung des MSL-Timers

Wie in Abb. 4.3-10 dargestellt, wird der MSL-Timer nach dem Absenden jedes TCP-Segments neu gestartet. Mit diesem *Timer* wird eine maximale Wartezeit (*Timeout*)

auf die Quittung angegeben. Kommt innerhalb dieser festgelegten maximalen Wartezeit keine Quittung an, wird die Übertragung des betreffenden TCP-Pakets wiederholt. Darin besteht das eigentliche Prinzip der Fehlerkontrolle bei TCP.

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

Bestätigung

Der Empfang der TCP-Pakete wird nur dann bestätigt, wenn ihre Reihenfolge vollständig ist. Somit kann die Situation eintreten, dass eine Reihe von TCP-Paketen (soweit die Window-Größe dies zulässt) sogar dann wiederholt übertragen werden muss, wenn sie bereits fehlerfrei beim Zielrechner ankamen. Wie dem Beispiel in Abb. 4.3-10 zu entnehmen ist, betrifft dies in unserem Fall die TCP-Pakete mit den Sequenznummern `m+200` und `m+300`. Diese müssen nochmals übertragen werden.

Round Trip Time

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

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

4.4 Implementierungsaspekte von TCP

TCP wurde in der Vergangenheit den sich ändernden Gegebenheiten der Netze (LANs und WANs) angepasst. Dies betrifft nicht die Protokollparameter, die über die Jahre unverändert geblieben sind, sondern vielmehr die Implementierung der Algorithmen in den TCP-Instanzen, d.h. den *TCP-Stack* als Bestandteil der Kommunikationssoftware in Betriebssystemen und Routern. Der TCP-Stack ist so zu optimieren, dass er unter den heute gegebenen Netzen und Anwendungen eine maximale Performance und eine hohe Übertragungssicherheit gewährleistet [RFC 1323, 2001 und 2018].

Die Einsatzgebiete von TCP haben sich durch die Popularität des Internet und den Entwicklungen der lokalen Netze stark erweitert, und es sind insbesondere folgende Netze zu unterstützen:

17 Internet of Things – Technische Grundlagen und Protokolle

IoT: Eine besondere Internet-Erweiterung

Der Wunsch und das Streben nach mehr Lebensqualität, mehr Energieeffizienz und bessere Umweltschutz hat zur Folge, dass verschiedene Sensoren für Überwachung und Meldung unterschiedlicher Ereignisse wie auch diverse, auf Basis von Mikroprozessoren aufgebaute Aktuatoren zur Ansteuerung wichtiger technischer, in unserem Alltag allgegenwärtiger 'smarter Dinge' ständig und immer mehr an Bedeutung und Verbreitung gewinnen. Damit man diese 'Dinge' überall und jederzeit ansteuern und effektiv nutzen kann, werden sie in das herkömmliche Internet integriert, und dies führt zu einer besonderen Internet-Erweiterung. Diese wird als *Internet of Things* (IoT) bzw. als *Internet der Dinge* bezeichnet.

WSANs, LR-PWANs, LLNs und 6LoWPAN

Die Kommunikation zwischen Things im IoT erfolgt meistens über sogenannte WSANs (*Wireless Sensor Actuator Networks*), die häufig als LR-WPANs (*Low-Rate Wireless Personal Area Networks*) nach dem IEEE-Standard 802.15.4 eingerichtet werden. Da WSANs sehr stark ressourcenbeschränkt, besonders energiearm und verlustbehaftet sind, nennt man sie auch *Constrained Networks* bzw. LLNs (*Low-Power and Lossy Networks*). Eine Adaption des IPv6 für den Einsatz im IoT ist unabdingbar, und sie wird als 6LoWPAN (*IPv6 over Low-power Wireless Personal Area Networks*) bezeichnet.

RPL und CoAP

Das IoT bilden de facto die in das Internet integrierten LLNs mit dem adaptierten IPv6. Im IoT ist ebenso wie im herkömmlichen Internet ein Routing-Protokoll nötig. Hierfür wurde das 'IPv6 Routing Protocol for Low-Power and Lossy Networks' (RPL) entwickelt und kann auch als Routing-Protokoll für das IoT angesehen werden. Für den Einsatz im IoT als Applikationsprotokoll wurde das webspezifische Protokoll CoAP (Constrained Application Protocol) vorgesehen. CoAP ist ähnlich wie das Protokoll HTTP (Hypertext Transfer Protocol) konzipiert.

Überblick über das Kapitel

Nach einer Darstellung von technischen Grundlagen des IoT in Abschnitt 17.1 werden dessen Protokolle in den nächsten Abschnitten erläutert, und zwar: 6LoWPAN in Abschnitt 17.2, das als RPL (*Routing Protocol for Low power and Lossy Networks*) bezeichnete Routing-Protokoll in Abschnitt 17.3 und das Applikationsprotokoll CoAP (*Constrained Application Protocol*) in Abschnitt 17.5. Schlussbemerkungen in Abschnitt 17.5 runden dieses Kapitel ab.

Ziel dieses Kapitels In diesem Kapitel werden u.a. folgende Fragen beantwortet:

- Wie kann man sich technisch das IoT vorstellen?
- Welche Bedeutung für IoT haben Cloud Computing und Fog Computing?
- Welche Technologien und Protokolle sind zur Realisierung des IoT nötig?
- Wie kann SDN (*Software Defined Networking*) im IoT eingesetzt werden?
- Worin besteht die Adaption von IPv6 für dessen Einsatz im IoT?
- Wie wird das Protokoll RPL konzipiert und welche Funktionen liefert es?
- Welche Funktionen stellt das Applikationsprotokoll CoAP zur Verfügung?

17.1 Herkömmliches Internet und IoT

Das IoT kann als eine besondere Erweiterung des herkömmlichen Internet angesehen werden. Dies sowie weitere strukturelle und funktionale Besonderheiten von IoT werden in den Abschnitten 17.1.1, 17.1.2 und 17.1.3 präsentiert.

Cloud und Fog Computing

Um IoT Services zu erbringen, diese entsprechend den von Menschen gestellten Anforderungen zu dokumentieren und in einer visuellen Form präsentieren zu können, ist eine spezielle *IoT Service Platform* nötig. Ihre Funktionalität wird durch verschiedene IoT-spezifische Clouds erbracht. Demzufolge spielt *Cloud Computing*, als eine zentralisierte Form von Computing im IoT, eine wichtige Rolle. Zusätzlich zu dieser zentralisierten Form ist eine dezentralisierte/verteilte Form von Computing im IoT nötig, und sie wird durch eine nebelartige Installation von Fog Nodes (einer Art *Mini Clouds*) am Rande des Internet erbracht. In diesem Zusammenhang spricht man im IoT von *Fog Computing*. Auf diese beiden Arten von Computing geht Abschnitt 17.1.4 detaillierter ein.

Cloud, Fog und ROOF Computing

Es wird gezeigt, dass man eine hierarchische, aus Clouds, Fog Nodes und ROOF-Komponents (*Real-time Onsite Operations Facilitation*) am Rande des Internet bestehende Struktur benötigt, um an IoT Services gestellte zeitkritische Anforderungen erfüllen und sogenannte Near Real-Time IoT Services erbringen zu können. Aus diesem Grund wird im Hinblick auf das IoT von *Cloud Computing*, *Fog Computing* und *ROOF Computing* gesprochen [Abschnitt 17.1.5].

Im Weiteren werden insbesondere die folgenden Besonderheiten von IoT zum Ausdruck gebracht:

Multilayer-Modell von IoT

- Dass mehrere funktionale Bereiche im IoT zu unterscheiden sind, dass diese in einer Hierarchie zueinander stehen und dass diese Hierarchie der funktionalen Bereiche zu einem Multilayer-Modell von IoT führt [Abschnitt 17.1.6].

Einsatz von SDN im IoT

- Um IoT Services ständig, quasi dynamisch, an neue Anforderungen anpassen zu können, können die Prinzipien von SDN (*Software Defined Networking*) zur adaptiven Gestaltung von IoT Services verwendet werden. Die Bedeutung von SDN im IoT präsentiert Abschnitt 17.1.7.

Protokoll-architektur von IoT Devices

- Dass die Protokollarchitektur von IoT Devices, wie die Protokollarchitektur von Rechnern im herkömmlichen Internet, auch als eine aus mehreren Schichten bestehende hierarchische Architektur dargestellt werden kann und dass die Hierarchie der IoT-Protokolle ein Multilayer-Modell dargestellt [Abschnitt 17.1.8].

6LoWPAN auf dem Layer 3

- Dass der Layer 2 in der Protokollarchitektur von IoT Devices die Frames als eine Art *Container* für die Übermittlung von Daten in IPv6-Paketen bereitstellt, dass die geringe Größe dieser Frames eine große Auswirkung auf den Layer 3 mit dem IPv6 hat und dass infolgedessen eine 'abgespeckte', als 6LoWPAN (*IPv6 over Low-power Wireless Personal Area Networks*) bezeichnete, Fassung des IPv6 auf dem Layer 3 zum Einsatz kommen muss [Abschnitt 17.1.9].

17.1.1 Allgemeine Definition von IoT

Bevor wir uns verschiedenen technischen IoT-Aspekten widmen, soll zuerst die allgemeine Definition von IoT aus der Recommendation ITU-T Y.2060 (06/2012) unter dem Titel 'Overview of the Internet of things' aufgegriffen und hier zitiert werden. In der Recommendation ITU-T Y.2060[1] wird das IoT wie folgt definiert:

> **Internet of Things (IoT)**: A global infrastructure for the information society, enabling advanced services by interconnecting (physical and virtual) things based on, existing and evolving, interoperable information and communication technologies.

Allgemeine Definition von IoT

Diese Definition kann etwas präziser und das IoT ausführlicher, aber immer noch sehr allgemein, mit dem folgenden Satz verfasst werden:

> Das IoT ist eine besondere globale Informations- und Kommunikationsinfrastruktur, mit *der unterschiedliche ubiquitäre (allgegenwärtige) Anwendungen und Services* durch diverse, auf der Basis von bestehenden und zukünftigen Informations- und Kommunikationstechnologien realisierten, *in der Regel drahtlosen, Vernetzungen verschiedener physikalischer und virtueller Dinge (Objekte) sowohl untereinander als auch mit dem herkömmlichen Internet* erbracht und *dann überall und jederzeit verfügbar gemacht werden können*.

IoT-Besonderheiten

Diese dargelegte Definition soll folgende drei IoT-Besonderheiten (oben unterstrichen) zum Ausdruck bringen, und zwar:

1. IoT-Anwendungen und -Services werden zukünftig überall und jederzeit verfügbar sein. Folglich können sie auch allen Menschen zugänglich gemacht werden.
2. Das IoT realisiert eine Kopplung diverser und in der Regel drahtlos verbundener verschiedener physikalischer sowie virtueller 'intelligenter Dinge', *Smart Objects* genannt, sowohl untereinander als auch mit dem herkömmlichen Internet. Demzufolge ist das IoT kein separates physikalisches Netz, sondern als eine besondere Ergänzung zum herkömmlichen Internet anzusehen.
3. IoT-Anwendungen und -Services können über das herkömmliche Internet, über andere spezielle Netze, insbesondere über Mobilfunknetze der neuesten Generationen 4G und 5G sowie über WLANs (*Wireless Local Area Networks*), überall und jederzeit verfügbar gemacht werden Abb. 17.1-2.

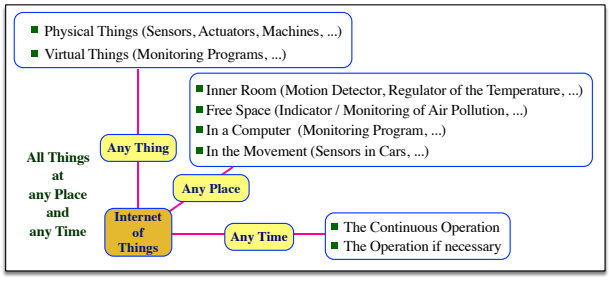


Abb. 17.1-1: Die drei Hauptmerkmale von IoT als dessen Dimensionen

[1] siehe: https://www.itu.int/rec/T-REC-Y.2060-201206-I

IoT-Dimensionen

Aus der hier präsentierten Definition des IoT gehen (aus theoretischer Sicht) drei Hauptbesonderheiten der Kommunikation hervor, die sind:

- 'Any Thing',
- 'Any Place' und
- 'Any Time'.

Wir können also festhalten: Alle Dinge sind überall und jederzeit zugänglich. Diese Besonderheiten des IoT können auch als 'IoT-Dimensionen' angesehen werden, wie in Abb. 17.1-1 verdeutlicht.

Hauptziel von IoT

Basierend auf den hier zum Ausdruck gebrachten IoT-Dimensionen könnte man das Hauptziel von IoT mit einem Satz wie folgt erfassen:

> Mittels des IoT sollen die technischen Dinge unseres alltäglichen Lebens allen Menschen überall und jederzeit zugänglich und nutzbar sein.

17.1.2 IoT aus funktionaler Sicht

IoT als funktionale Erweiterung des Internet

Die eben präsentierte allgemeine Definition von IoT soll nun aus einer funktionalen Sicht verdeutlicht werden. Zu diesem Zweck illustriert Abb. 17.1-2 die grundlegende Idee von IoT, und zwar die, dass IoT eine funktionale Erweiterung des herkömmlichen Internet um verschiedene 'smarte technische Dinge' unserer alltäglichen Umgebung darstellt. Dadurch können innovative und intelligente Anwendungen und Services realisiert und den Menschen überall und jederzeit verfügbar gemacht werden.

Access Gateways als Border Routers

Es sei angemerkt, dass die funktionale Erweiterung des herkömmlichen Internet um Smart Objects, de facto durch eine Erweiterung um das IoT, mithilfe intelligenter *Access Gateways* erfolgt; Abb. 17.1-2 bringt dies auch zum Ausdruck. Es sei angemerkt, dass diese Access Gateways auch die Funktion von Routern erbringen. Aus diesem Grund werden die Access Gateways auch als *Border Router* bezeichnet.

Kommunikationsarten über das herkömmliche Internet

Betrachtet man das herkömmliche Internet nur im Hinblick auf die Kommunikationsarten, also Datenflüsse, stellt man schnell fest, dass hier die zwei Akteure Menschen und Rechner im 'Spiel' sind. Im herkömmlichen Internet realisiert man somit nur folgende drei Kommunikationsarten:

P2P

1. *Person to Person – Mensch zu Mensch*: Diese Art der Kommunikation realisiert man beispielsweise mit dem E-Mail-Dienst.

P2C & C2P

2. *Person to Computer – Mensch zu Rechner* und *Person to Computer – Rechner zu Mensch*: Diese Art der gerichteten Kommunikation findet beim Abrufen von Webinhalten statt, und zwar wie folgt: Ein Mensch sendet eine Anforderung mit der Angabe, was er haben will (P2C), an einen Rechner, welcher als Webserver dient, und der Rechner übermittelt ihm die gewünschten Webinhalte (C2P).

C2C

3. *Computer to Computer – Rechner-zu-Rechner*: Beim jedem Austausch von Daten zwischen zwei Rechnern findet diese Art der Kommunikation statt.

Kommunikationsarten über das IoT

Zu den beiden Akteuren Mensch und Rechner des herkömmlichen Internet kommt nun mit dem IoT noch ein dritter hinzu – nämlich das 'Ding' (Thing). Als *Ding*

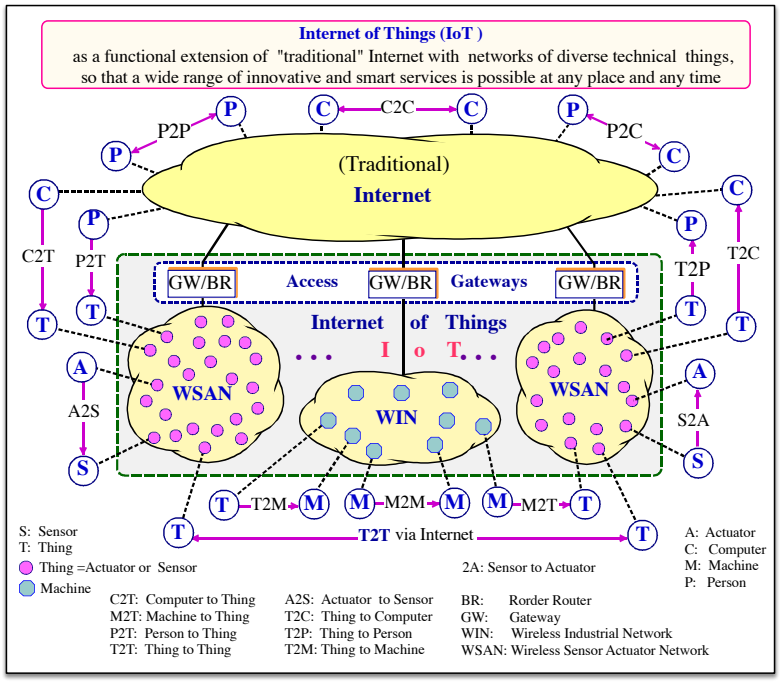


Abb. 17.1-2: Entstehung von IoT durch eine funktionale Erweiterung des herkömmlichen Internet um verschiedene 'smarte technische Dinge', also um 'smart technical things'

bezeichnen wir hier alle möglichen Alltagseinrichtungen (Geräte, Maschinen, Sensoren, Aktuatoren ...), die überall in unserem alltäglichen Leben vorgefunden werden können.

Weil IoT eine funktionale Erweiterung des herkömmlichen Internet darstellt, kommen also noch folgende Kommunikationsarten hinzu:

- *Person to Thing – Mensch zu Ding* und *Thing to Person (T2P) – Ding zu Mensch*: Diese beiden Arten der gerichteten Kommunikation werden bei typischen IoT-Anwendungen realisiert, die es Menschen ermöglichen, z.B. mithilfe von Smartphones verschiedene technische Dinge (wie Temperatur- oder Lichtregler) über das Internet anzusteuern. Bei solchen Anwendungen sendet ein Mensch eine Aufforderung (einen Befehl) an ein Ding (P2T). Dieses führt die Aufforderung aus und bestätigt dem Menschen die Ausführung der erhaltenen 'Aufgabe' durch das Absenden einer entsprechenden Mitteilung (T2P). P2T & T2P
- *Computer to Thing – Rechner zu Ding* und *Thing to Computer (T2C) – Ding zu Rechner*: Diese gerichteten Kommunikationsarten findet man bei der Ansteuerung technischer Dinge durch verschiedene Steuerungsrechner. Nehmen wir an: ein Rechner hat die Aufgabe, mehrere als Aktuatoren (z.B. Regler, Antriebselemente) dienende Dinge anzusteuern. In einem solchen Fall verläuft die Kommunikation typischerweise zwischen einem Rechner und technischen Dingen so, dass der Rech- C2T & T2C

ner einen Befehl an ein Ding sendet (C2T), dieses den erhaltenen Befehl ausführt und dem Rechner durch das Absenden einer Bestätigung mitteilt, dass es seine Aufgabe erledigt hat (T2C).

T2T

- *Thing to Thing (T2T) – Ding zu Ding*: Diese Art der Kommunikation findet man dann, wenn mehrere Dinge, genauer gesagt verschiedene technische Einrichtungen, untereinander vernetzt sind. Dies ist insbesondere dann der Fall, wenn eine von ihnen als Koordinator anderer Dinge fungiert – siehe hierfür Abb. 17.1-3. Da ein Thing als ein *Actuator* (A) oder ein *Sensor* (S) fungiert, unterscheidet man bei der Kommunikation T2T zwischen zwei derer Arten A2S und S2A.

M2M

- *Machine to Machine (M2M) – Maschine zu Maschine*: Die Kommunikationsart M2M hat bereits zu revolutionären Lösungen in der Industrie geführt. Man bezeichnet sie heute als '4. industrielle Revolution' und spricht in diesem Zusammengang auch von der 'Industrie 4.0'.

Anmerkung: Als *Aktuatoren*, vom englischen Begriff *Actuator* abgeleitet und auch *Aktoren* genannt, bezeichnet man alle kleinen technischen Komponenten, welche elektrische Signale (z.B. vom Steuerungsprozessor kommende Befehle) in physikalische Größen (z.B. Temperatur, Druck, ...) bzw. in mechanische Bewegung umsetzen und daher als Ausführungselemente (Antriebselemente, Regler, ...) dienen können. Ein Sensor, der als 'Fühler' dient, ist somit ein Gegenteil eines Aktuators.

IoT aus funktionaler Sicht

Betrachtet man das IoT aus rein funktionaler Sicht, so wie Abb. 17.1-2 es zeigt, dann ist es wie folgt zu interpretieren:

> Das IoT ist eine funktionale Erweiterung des Internet mit dem Ziel, Alltagseinrichtungen (Geräte, Maschinen, Sensoren, Aktuatoren, ...) unterschiedlicher Art und mit unterschiedlichen Fähigkeiten – also verschiedene smarte Dinge (*Smart Objects*), im Allgemeinen als Devices bezeichnet – sowohl untereinander als auch mit Rechnern am Internet so zu vernetzen, dass sie alle möglichen Internetdienste nutzen können, um dadurch die Erbringung einer breiten Palette neuer innovativer und intelligenter Applikationen und Services überall und jederzeit zu ermöglichen.

Das herkömmliche Internet vernetzt auf verschiedene Arten weltweit nur Menschen und Rechner untereinander. Durch die Migration zum IoT werden bald alle 'wichtigen technischen Dinge' der Welt sowohl mit Menschen und Rechnern als auch untereinander vernetzt werden. Dadurch werden sich für die Menschen vollkommen neue Perspektiven ergeben – um u.a. mehr Lebensqualität und eine bessere Energieeffizienz erreichen sowie einen größeren Umweltschutz garantieren zu können.

17.1.3 Grundlegendes technisches Konzept von IoT

Nachdem das IoT definiert und dessen Hauptziele erläutert wurden, soll nun kurz auf das grundlegende technische Konzept von IoT eingegangen werden. Abb. 17.1-3 illustriert dieses.

IoT als eine funktionale Erweiterung des Internet

Wie bereits hervorgehoben wurde [Abb. 17.1-2], stellt das IoT eine funktionale Erweiterung des herkömmlichen Internet dar. Diese Erweiterung besteht einerseits darin, dass verschiedene, zum IoT gehörende drahtlose Sensor-Aktuator-Netze/Netzwerke,

allgemein als WSANs (*Wireless Sensor Actuator Networks*) bezeichnet, über spezielle IoT Access Gateways an das herkömmliche Internet angebunden werden. Diese IoT Access Gateways, im Weiteren kurz *Access Gateways* bzw. *Gateways* genannt, können auch über kleine Datenbanken (Databases) verfügen, in denen die von Sensoren gemeldeten Ereignisse gesammelt und zwischengespeichert werden können, bevor sie über das Internet zuerst an IoT Clouds zur Bearbeitung/Verarbeitung und dann in einer entsprechenden Form an IoT-Leitstellen übermittelt werden.

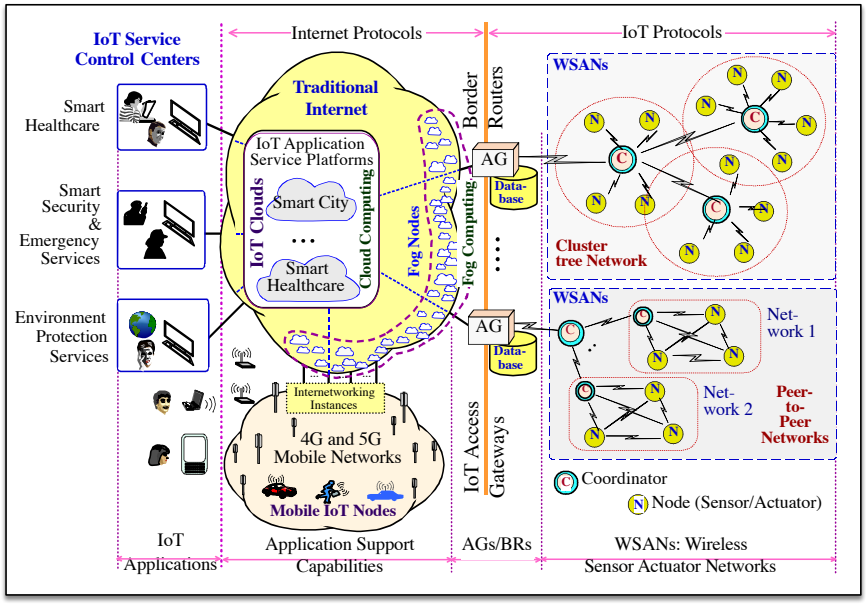


Abb. 17.1-3: Grundlegendes technisches Konzept von IoT: IoT als funktionale Erweiterung des herkömmlichen Internet

WSANs mit IoT Devices sind LLNs

WSANs mit Sensoren und Aktuatoren, mit IoT Devices also, sind aber sehr stark ressourcenbeschränkt, insbesondere energiearm (*low power*), und wegen der oft schlechten Signal-Rausch-Verhältnisse auch verlustbehaftet (*lossy network*). Sie werden deshalb als *Constrained Networks* bzw. als LLNs (*Low-Power and Lossy Networks*) bezeichnet.

6LoWPAN als Network Layer Protocol im IoT

Im IoT, genauer gesagt in WSANs als LLNs, soll das Protokoll IP zum Einsatz kommen. Da eine enorm große Menge von Sensoren und Aktuatoren im IoT enthalten sein kann, muss, damit man diese mit IP-Adressen 'versorgen' kann, das Protokoll IP in der Version IPv6 im IoT eingesetzt werden. Um das IPv6 in LLNs aber nutzen zu können, muss dieses entsprechend an die Besonderheiten von LLNs angepasst (adaptiert) werden. Diese speziell für den Einsatz in LLNs vereinfachte Form von IPv6 wird 6LoWPAN (*IPv6 over Low-power Wireless Personal Area Network*) genannt. Abschnitt 17.2 geht auf 6LoWPAN detailliert ein.

Mobile IoT-Devices am Internet

Abb. 17.1-3 soll auch verdeutlichen, dass alle in das herkömmliche Internet integrierte Netze wie z.B. WLANs und Mobilfunknetze der Generationen 4G und 5G als

ein heterogenes Zubringernetz zum Internet für diverse drahtlose und mobile Sensor/Aktuatoren, welche de facto mobile IoT-Devices darstellen, dienen können.

Fog Computing

Infolge der rasch voranschreitenden Entwicklung von IoT entstehen insbesondere am Internet-Rand (Internet Edge), an dem diverse allgegenwärtige Dinge an dieses angebunden werden, enorm große und rund um Erdkugel verstreute Mengen von Daten. Diese – eine Art Nebel bzw. auf Englisch Fog – müssen häufig fast in Echtzeit, folglich nahe ihrer Entstehung verarbeitet werden. Die daraus resultierende Notwendigkeit, diese großen, weit verstreuten Datenmengen am Internet-Rand, nahe ihrer Entstehung, schnellstmöglich zu verarbeiten, hat zur Entwicklung einer dezentralisierten Art der Datenverarbeitung in sogenannte *Fog Nodes* am Internet-Rand geführt. Sie wird als *Fog Computing* bezeichnet.

Cloud Computing

Durch die lokale Verarbeitung von Daten mithilfe von Fog Computing am Internet-Rand wird ihre Verzögerungszeit (Latenzzeit) stark reduziert [Abb. 17.1-4]. Die im IoT entstehenden riesigen Datenmengen – auch *Big Data* genannt – müssen aber oft auch noch zentral verarbeitet werden. Dies erfolgt mittels einer hierfür in Datacentern in Form von sogenannten *Clouds* zur Verfügung gestellten Rechenkapazität. Man spricht in diesem Zusammenhang von *Cloud Computing*.

Symbiose von Cloud und Fog Computing

Diese beiden Arten von Computing, das heißt Cloud Computing und Fog Computing, ergänzen sich im IoT ideal, sodass man sogar von einer Symbiose sprechen kann. Bei dieser Symbiose nimmt das Cloud Computing dem Fog Computing gegenüber eine übergeordnete, koordinierende Rolle ein. Dies bedeutet, dass einem Cloud Server mehrere Fog Nodes, eine Art 'Mini Clouds', untergeordnet werden, diese also seitens des Cloud Servers koordiniert werden [Abb. 17.1-4].

IoT Application Support Bereich

Allgemein gesehen müssen die IoT Services für Benutzer auf eine bestimmte Art und Weise aufbereitet werden, z.B. müssen bestimmte Ereignisse visuell angezeigt/signalisiert, entsprechend gespeichert und dokumentiert werden. Für diese Aufbereitung der IoT Services sind seitens der Benutzer spezielle IoT Service Platforms nötig. Diese Funktionalität wird durch eine enorm große Vielzahl von kleinen Fog Nodes dezentral und durch verschiedene, den Fog Nodes übergeordnete, IoT spezifische Clouds zentral erbracht. Betrachtet man alle Fog Nodes und Clouds zusammen, so bilden sie im Internet quasi einen breit verteilten funktionalen Bereich; dieser wird hier als *IoT Application Support* Bereich bezeichnet.

Leitstellen für IoT Services

Zur Steuerung und Nutzung bestimmter Kategorien der von IoT-Devices in WSANs bzw. in G4/G5-Mobilfunknetzen erbrachten Dienste müssen im herkömmlichen Internet spezielle Leitstellen für verschiedene Arten der IoT Services (z.B. Umweltschutz-, Notrufleitstelle) eingerichtet werden. Einige einfache und individuelle IoT Services (z.B. Temperatureinstellungen in Wohnungen) können von Menschen selbst unterwegs mithilfe tragbarer Rechner bzw. Smartphones in Anspruch genommen werden.

17.1.4 Cloud Computing und Fog Computing im IoT

Infolge der immensen Mengen verschiedener, weltweit im IoT verteilter Devices und Objekte werden riesige global verstreute Datenmengen erzeugt. Sie müssen zuerst entsprechend gespeichert und dann schnell verarbeitet werden. Hierfür kommt, wie

in Abb. 17.1-3 gezeigt, das Konzept von Cloud Computing zum Einsatz. Die IoT-spezifischen Fog Nodes und Clouds können als 'IoT Service Platforms' betrachtet werden.

Definition von Cloud Computing

Für eine kurze, fundierte Erläuterung der Funktionen von Cloud Computing ist eine Definition von Cloud Computing sehr hilfreich. Die in der Fachliteratur am häufigsten zitierte ist die des National Institute of Standards and Technology (NIST). Sie beschreibt das Cloud Computing wie folgt[2]:

> "A model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction".

Cloud Computing als Servicemodell

Aus dieser Definition geht hervor, dass man Cloud Computing als Servicemodell ansehen kann, welches bei Bedarf allgegenwärtige, 'bequeme' und über Netzwerke verlaufende Zugriffe auf einen gemeinsam genutzten Pool konfigurierbarer Computing-Ressourcen (z.B. Netzwerke, Server, Speicher, Applikationen und Dienste) ermöglicht, wobei die Ressourcen seitens des Service-Providers schnell, mit minimalem Managementaufwand bereitgestellt und freigegeben werden können.

Die IoT Services müssen auf eine bestimmte Art und Weise für Benutzer aufbereitet werden: Beispielweise müssen einige von IoT Devices/Objects stammende Ereignisse visuell angezeigt/signalisiert, entsprechend gespeichert und dokumentiert werden. Für diese Aufbereitung von IoT Services ist ein spezieller funktionaler Bereich – quasi Layer – nötig. Er fungiert als eine Art 'IoT Application Support'. Seine Funktionen werden durch spezielle IoT Clouds, also durch *Cloud Computing*, erbracht.

Schwächen von Cloud Computing im IoT

Wie bereits erwähnt, werden zwecks der Erbringung bestimmter IoT Services die von IoT Devices und anderen IoT Objekten stammenden Daten mithilfe von Cloud Computing zentral im Internet aufbereitet. Da IoT Services oft zeitkritisch sind und zahlreiche Services fast in Echtzeit realisiert werden müssen, sind sie wegen der relativ großen Entfernungen zwischen zentralen Clouds und IoT Devices/Objects mithilfe von Cloud Computing nicht realisierbar. Aus diesem Grund muss das Cloud Computing zur Ermöglichung stark zeitkritischer IoT Services zusätzlich in dezentralisierter Form am Internet-Rand, und somit in der Nähe von IoT Devices/Objects, verfügbar sein.

Definition von Fog Computing

Die Kompensation der soeben erwähnten Schwäche von Cloud Computing wird in Form von einer Vielzahl von Mini-Clouds (Mini-Wolken), quasi eines Nebels (Fog), realisiert, mit denen die Funktionalität einiger zentraler IoT-spezifischer Clouds in dezentraler Form am Internet-Rand erbracht wird. Diese Vorgehensweise hat zur Entstehung von Fog Computing geführt. Dieser Begriff wurde 2012 von der Firma Cisco eingeführt. Die am häufigsten in der Fachliteratur zitierte, kurze und gut zutreffende Definition von Fog Computing vom OpenFog Consortium[3] lautet:

[2] siehe: https://csrc.nist.gov/publications/detail/sp/800-145/final
[3] siehe: https://www.openfogconsortium.org/page-section/definition-of-fog-computing/

> "A horizontal, system-level architecture that distributes computing, storage, control and networking functions closer to the users along a cloud-to-thing continuum".

Besonderheiten von Fog Computing

Fog Computing kann als eine Ergänzung zum Cloud Computing angesehen werden. Aus der hier zitierten Definition geht hervor, dass es eine horizontal 'verstreute', insbesondere dem Computing, Storage, Control und Networking dienende Systemarchitektur mit verteilten Funktionen darstellt. Diese Funktionen, wie in Abb. 17.1-3 gezeigt, werden mithilfe von Fog Nodes entlang des Internet-Kontinuums zwischen Clouds und Things näher an Benutzer und somit näher an IoT Devices/Objects 'gebracht'. Fog Computing führt also zu einer Verteilung von Computing-Ressourcen, durch die diese überall, möglichst nah an Benutzern und IoT Devices/Objects platziert werden können. Die am Internet-Rand entlang platzierten Computing Ressourcen tragen dazu bei, dass die 'Intelligenz' verteilter Systeme aus IoT Clouds an die diese 'Intelligenz' benötigten 'IoT-Objekte' angenähert werden kann.

Flexibilität und Agilität mit Fog Computing

Fog Computing ist eine extrem bedeutende Weiterentwicklung von Cloud Computing, denn dem Konzept von Fog Computing liegt im Gegensatz zu dem von Cloud Computing ein dezentrales, horizontal ausgelegtes Modell für das verteilte Computing zugrunde. Das dezentrale Modell ist flexibler und agiler als das traditionelle zentralisierte Computing-Paradigma. Diese Flexibilität und Agilität, insbesondere beim Einsatz von SDN (*Software Defined Networking*) zur Orchestrierung von Fog Nodes, ist die Voraussetzung für die Entwicklung einer breiten Palette weltweit verteilter IoT Services.

17.1.5 Near Real-Time IoT Services mit Fog Computing

Begriff: Near Real-Time

Es gibt eine Vielzahl unterschiedlicher Typen von IoT Services, die in gewisser Weise bzw. in einem bestimmten Grade zeitkritisch sind. Man bezeichnet sie als *Near Real-Time Services* bzw. kurz *NRT-Services*. Auf Deutsch könnte man diese Services *Nahezu-Echtzeit-Services* nennen. Der Begriff 'Near Real-Time' bezieht sich auf die Latenzzeit (Verzögerungszeit) – also auf den Zeitraum zwischen einem Ereignis und einer sichtbaren Reaktion auf dieses. Die Latenzzeit in IoT Services entsteht hauptsächlich durch den für die Verarbeitung der das Ereignis betreffenden Daten erforderlichen Zeitaufwand und ihrer Übermittlung in Netzwerken.

Begriff: Real-Time

Im IoT werden aber unter anderem sehr stark zeitkritische Services realisiert. Diese werden als *Real-Time Services* (RT-Services) bezeichnet. Ein Beispiel ist das 'Support of Autonomous Driving'. Sollen selbstfahrende Autos auf öffentliche Straßen geschickt werden, so müssen sie in Echtzeit auf sicherheitsrelevante Ereignisse reagieren, was verdeutlicht, dass die Möglichkeit der Realisierung von RT-Services im IoT als eine Grundvoraussetzung für das *Internet of Vehicles*indexInternet of Vehicles (IoV) angesehen werden muss.

Near Real-Time versus Real-Time

Es sei angemerkt, dass der Unterschied zwischen den Begriffen Near Real-Time (NRT) und Real-Time (RT) unpräzise ist. Es ist nicht allgemeingültig festgelegt, bis zu welcher Latenzzeit es sich um einen RT-Service handelt und ab welcher Latenzzeit von einem NRT-Service die Rede sein sollte. In jedem Fall muss daher bestimmt werden,

ob es sich um einen RT-Service oder um einen NRT-Service handelt. Die Latenzzeiten bei NRT-Services liegen typischerweise im Bereich von einigen Sekunden.

Realisierung von NRT- und von RT-Services

Bei der Realisierung von NRT- und von RT-Services im IoT spielt das Fog Computing eine entscheidende Rolle. Abb. 17.1-4 verdeutlicht dies.

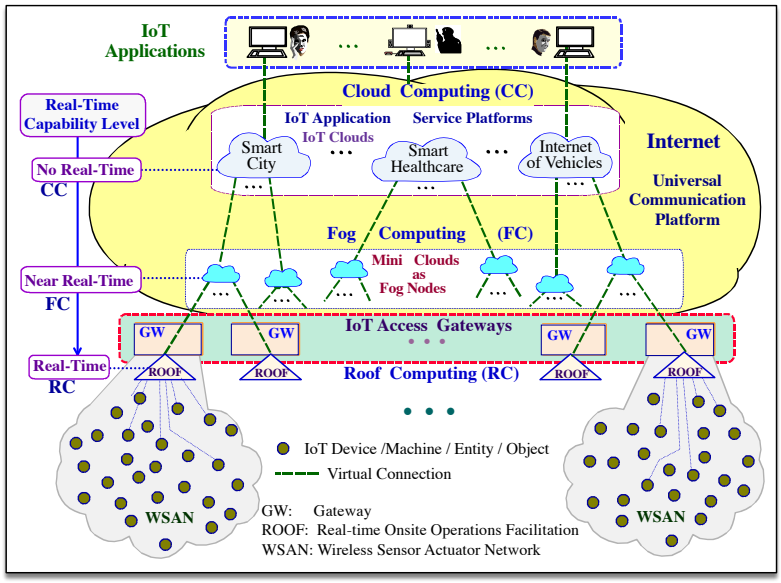


Abb. 17.1-4: Prinzip der Realisierung von RT- und NRT-Services im IoT

NRT-Service mit Fog Computing

Typischerweise wird Fog Computing am Internet-Rand mittels von in der Nähe von IoT Access Gateways installierten Fog Nodes realisiert. Zur Verwirklichung eines geografisch breit verteilten NRT-Services wie z.B. Smart City oder Smart Healthcare müssen meist mehrere Fog Nodes entsprechend räumlich verteilt und seitens einer zentralen Cloud koordiniert werden. In Clouds werden IoT-betreffende, zeitunkritische Daten verarbeitet, gespeichert und in einigen Fällen auch langfristig archiviert. Zeitkritische IoT-Daten werden währenddessen in den Fog Nodes verarbeitet und dort gegebenenfalls auch gespeichert. Ein Beispiel ist der IoT Service 'Smart City'. Bei diesem können Fog Nodes in Gebäuden bzw. in anderen städtischen Einrichtungen untergebracht werden.

> **Anmerkung**: Es sei hervorgehoben, dass man unter einem Fog Node im Allgemeinen lediglich eine Hardware- und Software-Struktur versteht, welche in der Regel die Verfügbarkeit von Funktionen wie Computing, Storage, Communication und Management garantiert. Oft werden diese Funktionen in Form einer verteilten Struktur ausgeführt.

RT-Service mit Roof Computing

Um RT-Services im IoT ermöglichen zu können, müssen in direkter Nähe von IoT Devices/Objects die hierfür speziell vorgesehenen Komponenten installiert werden. Zur weltweit einheitlichen Bereitstellung verschiedener RT-Services auf der Grundlage von IoT ist ein standardisiertes Konzept für eine neue Art von Real-Time Computing nötig. Die Entwicklung eines solchen Konzeptes wird von der Arbeitsgruppe ROOF (*Real-time Onsite Operations Facilitation*) bei der IEEE (*Institute of Electrical*

and Electronics Engineers) koordiniert. Aus diesem Grund wird diese neue Art von Real-Time Computing als Roof Computing bezeichnet. Abb. 17.1-4 illustriert die Bedeutung von Roof Computing und bringt zum Ausdruck, dass sich Cloud Computing, Fog Computing und Roof Computing sowohl gegenseitig ergänzen als auch in einer Hierarchie zueinander stehen. Das Roof Computing kann als eine grundsätzlich neue Vorgehensweise im IoT und somit als neues Computing-Paradigma angesehen werden.

IEEE-Standard 1931.1

Der von der IEEE verfasste 'Standard for an Architectural Framework for Real-time Onsite Operations Facilitation (ROOF) for the Internet of Things' mit der Identifikation `IEEE 1931.1` definiert die Prinzipien für Roof-Computing und -Vernetzung sowie für die technische und funktionale Interoperabilität von IoT-Systemen zwecks einer sicheren und in Echtzeit stattfindenden Kooperation in einem lokalen Bereich – beispielsweise zu Hause, am Arbeitsplatz, in einer Produktionshalle etc.

17.1.6 Funktionales Multilayer-Modell von IoT

Betrachtet man das in Abb. 17.1-3 dargestellte technische Konzept von IoT näher, so ist eine schichtenartige funktionale Architektur erkennbar. Diese kann als eine Art 'funktionale Architektur von IoT' betrachtet werden. Abb. 17.1-5 zeigt diese Architektur.

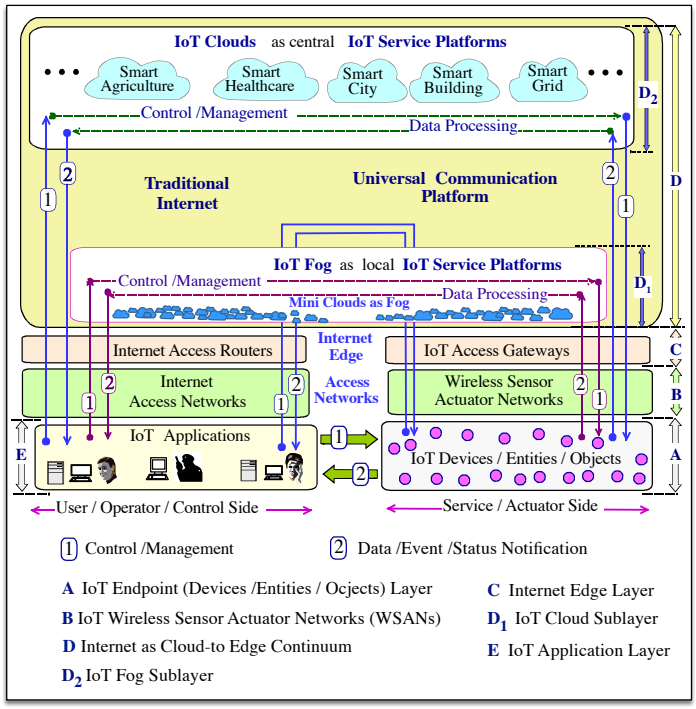


Abb. 17.1-5: Allgemeine funktionale Architektur von IoT

Es sei hervorgehoben, dass zwei besondere Seiten (Sides) innerhalb der in Abb. 17.1-5 gezeigten funktionalen IoT-Architektur zu unterscheiden sind, und zwar:

IoT-Seite

- **IoT Service/Actuator Side**: Diese Seite stellt die funktionale Erweiterung des herkömmlichen Internet um IoT-spezifische Systemkomponenten dar. Sie repräsentiert eine über spezielle Access Gateways an das herkömmliche Internet erbrachte Anbindung verschiedener IoT Devices/Objects, die typischerweise auch untereinander, oft über WSANs (Wireless Sensor Actuator Networks), verbunden sind.

Internet-Seite

- **User/Control Side**: Diese Seite repräsentiert die Endkomponenten des herkömmlichen Internet, die von Benutzern (Users) benötigt werden, um verschiedene IoT Devices/Objects steuern, deren Status einstellen und abfragen zu können. Hierfür werden am herkömmlichen Internet oft spezielle, dienstspezifische Leitstellen (z.B. Umweltschutz-, Notrufleitstelle) zur Nutzung der von IoT Devices/Objects erbrachten Dienste eingerichtet.

Multilayer-Modell von IoT nach ITU

Die in Abb. 17.1-5 dargestellten funktionalen Bereiche/Layers im IoT – bezeichnet als A, B, C, D, D_1, D_2 und E – ermöglichen eine anschauliche Illustration des allgemeinen Konzeptes von IoT in Form eines funktionalen Multilayer-Modells. Abb. 17.1-6 zeigt ein solches Modell. Ein ähnliches Multilayer-Modell des IoT, aber ohne Fog Computing Layer, wurde bereits von der ITU-T (International Telecommunication Union/Telecommunication Standardization Sector) vorgeschlagen.

Nach dem funktionalen Multilayer-Modell von IoT sind folgende Bereiche, die man als 'funktionale IoT Layers' betrachten kann, zu unterscheiden:

1. **IoT Device Layer**: Zu diesem Bereich gehören WSANs mit diversen IoT Devices und Access Gateways zur Anbindung WSANs an das herkömmliche Internet. Demzufolge wird der Device Layer in folgende drei Sublayers aufgeteilt:

   A): *Sensor und Actuator Sublayer*: Zu diesem Sublayer gehören Sensoren und Aktuatoren, also de facto die Endeinrichtungen des IoT.

   B): *Sensor und Actuator Networking Sublayer*: Dieser Sublayer repräsentiert Netze, welche in der Regel WSANs sind und dazu dienen, Sensoren und Aktuatoren untereinander zu vernetzen und diese auch über Access Gateway an das herkömmliche Internet anzubinden.

   C): *Access Gateway Sublayer*: Dieser Sublayer repräsentiert Access Gateways, über welche WSANs mit dem herkömmlichen Internet verbunden sind.

   Anmerkung: Die Sensoren haben sehr oft eine ganz andere Leistungsfähigkeit als Rechner am herkömmlichen Internet. Sensoren am IoT sind oft winzig und beziehen ihre Energie aus kleinen Batterien – sind also energiearm. Folglich ist ihre Rechenkapazität limitiert. U.a. infolge dieser Besonderheiten können die gut bekannten Protokolle IP, TCP und HTTP aus dem herkömmlichen Internet im IoT nicht eingesetzt werden. In WSANs werden daher spezielle IoT-spezifische Protokolle verwendet. Diese sind: 6LoWPAN (*IPv6 over Low-power WPAN*) und CoAP (*Constrained Application Protocol*).

2. **Network Layer**: Diesen Bereich, der die Kommunikation zwischen IoT-spezifischen, oft als Border Router dienenden Access Gateways und IoT-Applikationen ermöglicht, bilden Teile des herkömmlichen Internet zusammen mit WLANs und Mobilfunknetzen der vierten und fünften Generation (d.h. 4G und

5G). Das herkömmliche Internet zusammen mit WLANs und den Mobilfunknetzen G4 und G5 stellt somit die Grundlage für das Erbringen verschiedener IoT Services und die Realisierung eines ubiquitären, überall möglichen Computing (Ubiquitous Computing) dar. Im Network Layer sind die folgenden, zwei besonderen Sublayers enthalten:

D_1: *Fog Computing Sublayer*: Dieser Sublayer zeichnet sich dadurch aus, dass seine Funktionalität durch eine enorm große Vielzahl verschiedener Fog Nodes, einer Art 'Mini Clouds', in der Nähe von IoT Devices, entlang vom Internet-Rand nämlich, erbracht wird. Da virtuelle Rechner, sogenannte Virtual Machines, oft als Fog Nodes eingesetzt werden, können dadurch diverse virtualisierte Netzwerkfunktionen am Internet-Rand für IoT-Zwecke zur Verfügung gestellt werden.

D_2: *Cloud Computing Layer*: In diesem Bereich werden die IoT Services in Clouds quasi zentral aufbereitet (unter anderem signalisiert und visualisiert). Daher fungieren Clouds als 'Service Plattforms' für wichtige IoT Services, z.B. für Smart City, Smart Energy, Smart Environment und eHealth.

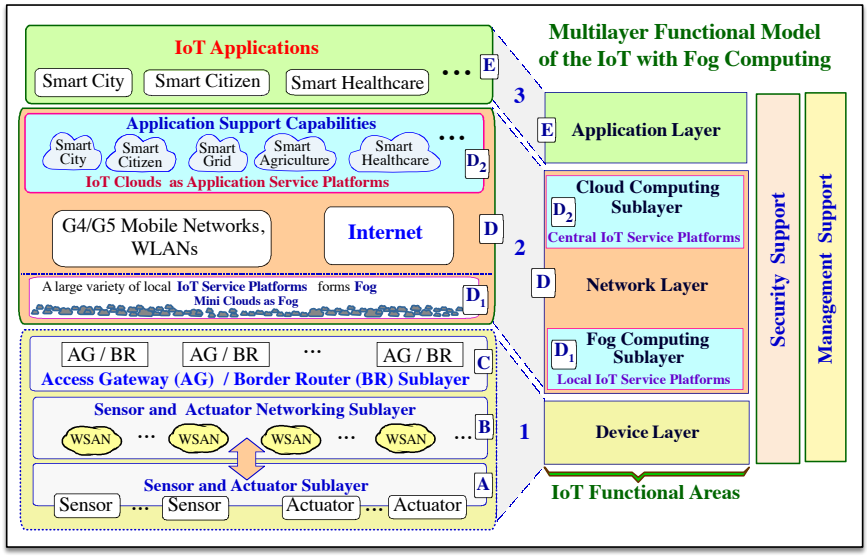


Abb. 17.1-6: Allgemeines funktionales Multilayer-Modell von IoT

3. **Application Layer**: IoT Services werden Benutzern mithilfe bestimmter IoT-Applikationen zugänglich gemacht. Da es sich dabei in der Regel um komplexe, verteilte, oft eine audiovisuelle Kommunikation über das Internet voraussetzende Anwendungen handelt, versucht man sie als Quasistandards zu verfassen. Als Beispiel sei die – vor allem der Patientenüberwachung dienende – IoT-Applikation eHealth genannt.

17.1.7 Bedeutung von SDN im IoT

Logische Architektur eines Servers und SDN

SDN (*Software Defined Networking*) kann als Framework zum Einrichten flexibler, bedarfsgerechter, quasi programmierbarer Netzwerkdienste angesehen werden [xx]. Abb. 17.1-7 bringt dies zum Ausdruck und verdeutlicht dabei, dass die logische Struktur eines Netzwerks beim Einsatz von SDN der allgemeinen logischen Architektur eines physischen Servers ähnelt. Hierbei gibt es folgende Entsprechungen:

- Die *Ethernet-Adapterkarten* (Network Interface Controller, NIC) im Server entsprechen dem Network Device Layer, der eine Vernetzung von sogenannten *SDN-enabled Devices* repräsentiert.
- Das *Operation System* (Betriebssystem) im Server entspricht dem Control Layer, d.h. dem Layer mit Systemkomponenten, in denen die sogenannten SDN Controllers untergebracht sind und mit deren Hilfe die *SDN-enabled Devices* konfiguriert, koordiniert und gesteuert werden können. Es sei hervorgehoben, dass man im IoT, neben Fog Nodes, mithilfe von SDN Controllern auch andere Devices (wie Access Gateways, siehe Abb. 17.1-8) ansteuern kann und diese so auch SDN-enabled sein können.

Operation System versus SDN Controller

- Den *Applikationen im Server* entsprechen weitgehend den Applikationen innerhalb einer SDN-based Network Infrastructure. Es handelt sich dabei um Applikationen, von denen entsprechende Steuerungsvorgaben, eine Art Direktiven, an SDN Controller übermittelt werden.

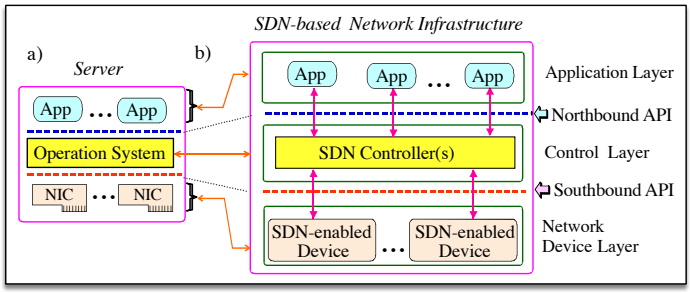


Abb. 17.1-7: Grundlegende Idee von SDN: a) Logische Serverstruktur als 'Quelle' der SDN-Idee, b) Funktionale Komponenten einer SDN-based Network Infrastructure
API: Application Programming Interface, App:Application, NIC: Network Interface Controller (Card)

SDN-based Network Infrastructure

Die drei funktionalen, als Layers bezeichneten Schichten innerhalb der logischen Architektur von SDN-based Network Infrastructure sind:

- **Network Device Layer**: Dieser Layer repräsentiert eine Vernetzung von SDN-enabled Devices untereinander. Im IoT können vor allem Fog Nodes und Access Gateways am Internet Edge auf eine besondere Art 'SDN-enabled' sein; siehe Abb. 17.1-8 und Abb. 17.1-9.
- **Control Plane**: Zu dieser Plane gehören SDN Controller. Sie setzen die von Applikationen generierten Anweisungen (Richtlinien, Policies) in an die SDN-enabled Fog Devices übermittelte Nachrichten um. Dadurch können Fog Nodes dem aktu-

ellen Bedarf entsprechend über eine genormte, sogenannte Southbound API sogar IoT-weit einheitlich angesteuert werden.

- **Application Plane**: Innerhalb dieser Plane werden verschiedene, als SDN-Applikationen bezeichnete, Software Tools zum Einrichten von Netzwerkdiensten angesiedelt. Genau genommen dienen diese Applikationen der Erstellung von Anweisungen zur Ansteuerung der SDN-enabled Fog Devices. Die sogenannte *Northbound API*, als offene Softwareschnittstelle, soll jede den aktuellen Anforderungen entsprechende Konfiguration der Netzwerkdienste auf eine einheitliche Art und Weise ermöglichen.

SDN Controllers in IoT Clouds

Im IoT können spezielle SDN Controllers in IoT Clouds eingesetzt werden, um SDN-enabled Access Gateways am Internet Edge entsprechend den aktuell geltenden Anforderungen (z.B. den aktuellen Sicherheitsanforderungen) von einer Cloud über das Internet schnell konfigurieren zu können. Abb. 17.1-8 illustriert eine hierfür geeignete Systemlösung.

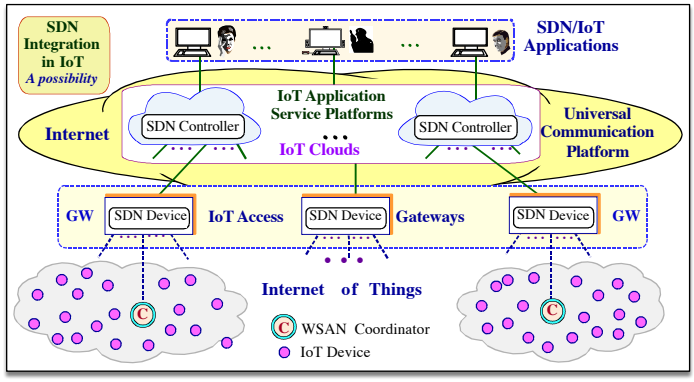


Abb. 17.1-8: Beispiel für den Einsatz von SDN in IoT Access Gateways

Garantie der Sicherheit

Garantie der Sicherheit

Wir wollen betonen, dass der Einsatz von SDN in IoT Access Gateways die Nutzung immer aktuell geltender Security Policies zwecks Gewährung der Sicherheit im Zugangsbereich zu privaten WSANs auf einem konstant hohen Niveau ermöglichen kann. In IoT Access Gateways können spezielle, häufig als SDN Switches bezeichnete funktionelle Komponenten enthalten sein. Diese können von in Clouds untergebrachten SDN Controllers konfiguriert und auf diese Weise schnell an aktuell geltende Anforderungen angepasst werden.

SDN-enabled Fog Computing

Beim Einsatz von SDN zur Unterstützung von Fog Computing spricht man von SDN-enabled Fog Computing. Diese besondere Art von Fog Computing besteht in der Konfigurierung und Steuerung der Fog Nodes durch SDN Controllers. Hierfür können Controllers sowohl in Clouds als auch in speziellen, übergeordneten Fog Nodes installiert werden. So ergeben sich die zwei typischen, in Abb. 17.1-9 dargestellten Vernetzungsstrukturen für die Realisierung von SDN-enabled Fog Computing.

SDN Controller in einer Cloud

Abb. 17.1-9a zeigt eine Variante von Fog Computing, die sich dadurch charakterisiert, dass SDN Controllers zur Konfiguration und Steuerung von Fog Devices in Clouds untergebracht sind. Mit einem dort untergebrachten Controller können zahlreiche ho-

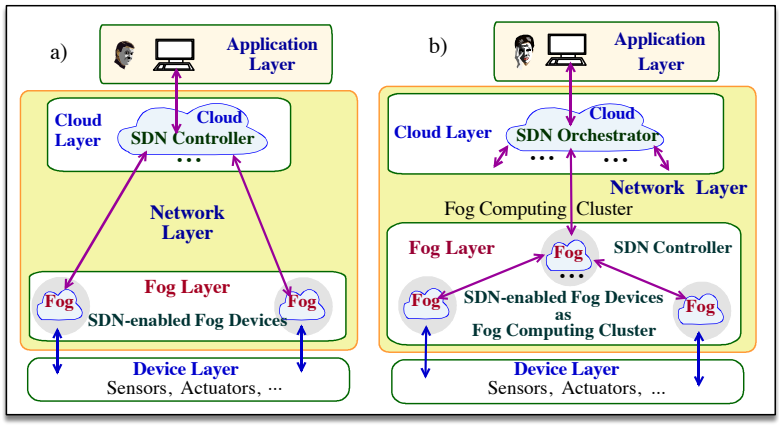


Abb. 17.1-9: Typische Vernetzungsstrukturen für SDN-enabled Fog Computing mit SDN Controllern: a) in einer Cloud, b) in einem übergeordneten Fog Node

rizontal in der Nähe von IoT Devices installierte Fog Devices konfiguriert, gesteuert sowie deren Parameter abgefragt werden. Diese Variante von SDN-enabled Fog Computing eignet sich insbesondere für einen Einsatz, bei dem der Device Layer räumlich weit verteilt ist, also z.B. zur Steuerung von Verkehrsflüssen auf Autobahnen oder zum Umweltmonitoring.

SDN Controller in einem Fog Node

Sind mehrere Fog Devices räumlich nicht weit voneinander installiert, können sie zu einem Fog Computing Cluster 'gruppiert' werden. In einem solchen Cluster kommt ein ausgewählter Fog Node als quasi zentraler und übergeordneter Fog Node zum Einsatz. In ihm wird ein SDN Controller installiert, mit dem die restlichen Fog Nodes im Cluster aktuellen Anforderungen entsprechend konfiguriert werden. Abb. 17.1-9b illustriert diese Variante von SDN-enabled Fog Computing. Eine wichtige Besonderheit einer solchen Lösung besteht darin, dass mehrere Fog Computing Clusters mit einem in einer Cloud untergebrachten sogenannten SDN Orchestrator gemäß den durch IoT-Applikationen aktuell gestellten Anforderungen konfiguriert, koordiniert und administriert werden können. Diese Fog-Computing-Lösung ist flexibel konfigurierbar und eignet sich zum Beispiel besonders gut zur Umsetzung der Idee von Smart Cities.

17.1.8 Protokollarchitektur von Devices im IoT

Protokollarchitektur als Multilayer-Modell

Die Protokollarchitektur von Devices im IoT kann – ähnlich wie Protokollarchitektur von Rechnern im Internet – in Form eines aus mehreren Schichten bestehenden Modells dargestellt werden. Abb. 17.1-10 zeigt dieses Schichtenmodell und bringt damit zum Ausdruck, dass die Protokollarchitektur von Devices im IoT in Form eines sogenannten Multilayer-Modells dargestellt werden kann.

Die unteren Schichten im IoT-Schichtenmodell

Die ersten unteren zwei Schichten – Physical Layer und Data Link Layer (auch als MAC Layer bezeichnet) – werden oft im IoT-Schichtenmodell durch IEEE-Standards festgelegt. Von diesen sind insbesondere die folgenden Standards hervorzuheben:

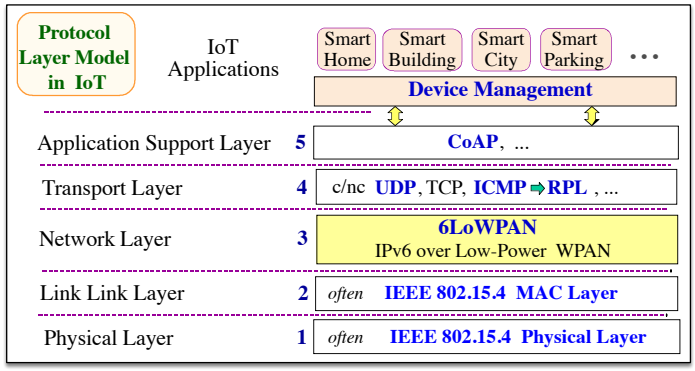


Abb. 17.1-10: Protokollarchitektur von Devices im IoT bildet eine Multilayer-Struktur
CoAP: Constrained Application Protocol, c/nCUDP: compressed / non compressed UDP, MAC: Media Access Control; RPL: Routing Protocol for Low power and Lossy Networks, WPAN: Wireless Personal Area Network

- Standard 802.15.4 mit der Spezifikation von LR-WPAN (*Low-Rate Wireless Personal Area Networks*) und Standard 802.15.4e,
- Standard 802.15.4e mit der Spezifikation des Verfahrens TSCH (*Low-Rate Wireless Personal Area Networks*) zum Aufbau von Wireless Networks in industriellen Bereichen insbesondere zur Kommunikation Maschine-zu-Maschine,
- Standard 802.15.6 für LR-WBANs (*Low-Rate Wireless Body Area Networks*).

Besonderheiten von LR-WPANs und LR-WBANs

Die Standards für LR-WPAN, LR-WBANs spezifizieren die untersten zwei Schichten, d.h. Physical Layer und MAC Layer (*Media Access Control*). Dabei beschreiben sie spezielle Funktionen, die nur in drahtlosen, energiearmen drahtlosen Sensor-Aktuator-Netzen, generell bezeichnet als WSANs (*Wireless Sensor Actuator Networks*), realisiert werden müssen. In Sensor-Aktuator-Netzen werden neben stationären (ortsgebundenen) Sensoren, die eine externe Stromversorgung benötigen, oft auch kleine Funksensoren eingesetzt, die ihre Energieversorgung aus winzigen Batterien beziehen oder die benötigte Energie aus ihrer Umwelt in Form von Solarenergie selbst gewinnen können. Um den Energieverbrauch in Sensoren zu reduzieren, werden diese oft in den Schlafmodus versetzt. Soll ein 'schlafender' Sensor aktiv werden, so muss er zuerst geweckt werden. Die Funktion 'Wecken schlafender Sensoren' wird innerhalb der Schicht 2 im IoT-Protokollmodell realisiert.

LR-WPANs und LR-WBANs sind LLNs

Die WSANs – nach IEEE-Standards für LR-WPANs und LR-WBANs – sind als LLNs (*Low-Power and Lossy Networks*) bzw. als Constrained Networks zu betrachten. In LLNs wird nur das Protokoll IPv6 eingesetzt. Um das IPv6 in stark ressourcenbeschränkten Netzwerken einsetzen zu können, muss dieses entsprechend angepasst (adaptiert) werden. Die speziell für den Einsatz von IPv6 in LLNs realisierte Vereinfachung und Komprimierung von IPv6 wird 6LoWPAN (*IPv6 over Low-power Wireless Personal Area Networks*) genannt.

6LoWPAN als Network Layer Protocol

6LoWPAN fungiert als Protokoll des Network Layer innerhalb der Protokollarchitektur von Devices im IoT und stellt eine an die Anforderungen von LLNs ad-

aptierte und komprimierte Form des Protokolls IPv6 dar. Der Network Layer mit 6LoWPAN in der Protokollstruktur von IoT Devices kann auch als *IPv6 Adaptation Layer* angesehen werden. Mit 6LoWPAN wird der Header des Protokolls IPv6 komprimiert und an die Besonderheiten von LLNs angepasst [Abschnitt 17.2].

Transport Layer im IoT

Auf dem Transport Layer (Layer 4) im IoT kommt als Transportprotokoll hauptsächlich das verbindungslos arbeitende User Datagram Protocol (UDP) zum Einsatz, wobei auch dessen Header komprimiert werden kann. Folglich unterscheidet man zwischen compressed UDP (cUDP) und non-compressed UDP (ncUDP). 6LoWPAN ermöglicht auch den Einsatz des verbindungsorientierten Transportprotokolls TCP (*Transmission Control Protocol*) und des ICMP (*Internet Control Message Protocol*).

RPL als eine Art 'Ableger vom ICMP'

Auf dem Transport Layer spielt das Protokoll ICMP eine besonders wichtige Rolle. Mit seiner Hilfe können andere Protokolle zwecks Management und Routing im IoT als eine Art ICMP-Anwendung 'generiert' werden. Die Nachrichten von RPL, des Routing-Protokolls im IoT also, stellen einen besonderen Typ von ICMP-Nachrichten dar. *Infolgedessen ist das RPL als Protokoll auf dem Layer 4 (Transport Layer) und als eine modifizierte Variante von ICMP anzusehen.* Abb. 17.1-6 bringt dies zum Ausdruck. Auf RPL geht Abschnitt 17.3 detaillierter ein.

CoAP als Applikationsprotokoll im IoT

Da drahtlose Sensor-Aktuator-Netzwerke sehr stark ressourcenbeschränkt (resource constrained), energiearm (low power) sind und wegen der oft schlechten Signal-Rausch-Verhältnisse auch verlustbehaftet (lossy network) arbeiten, bezeichnet man sie als *Constrained Networks*. Davon abgeleitet bezeichnet man das in diesen Netzwerken eingesetzte webspezifische Applikationsprotokoll als *Constrained Application Protocol* (CoAP). Das CoAP kann als eine Art Webtransferprotokoll in Constrained Networks angesehen werden, welches das klassische Webtransferprotokoll HTTP im IoT ersetzt. So kann man verschiedene IoT-Anwendungen in herkömmliche Webanwendungen integrieren. Das Konzept von CoAP wird in Abschnitt 17.4 detaillierter erläutert. Auf dem Layer 6 der Protokollstruktur von IoT Devices, dem Application Layer, ist das Device Management für die verschiedenen Anwendungen des IoT positioniert.

17.1.9 Protokollarchitektur von IoT Access Gateways

Wie bereits in Abb. 17.1-3 gezeigt wurde, sind für die Anbindung drahtloser Sensor-Aktuator-Netze an das herkömmliche Internet spezielle IoT Access Gateways notwendig. Abb. 17.1-11 illustriert die Protokollarchitektur und damit auch die Hauptaufgaben der Access Gateways.

Warum IPv6 im IoT ein-gesetzt werden muss?

Aus Abb. 17.1-11 geht hervor, dass das 'neue' Internetprotokoll IPv6 als Übermittlungsprotokoll im herkömmlichen Internet zur Kommunikation mit Sensor-Aktuator-Netzen innerhalb der Schicht 3 (Paketübermittlungsschicht) verwendet wird. Der Grund dafür ist folgender: Die Adressen des alten Internetprotokolls IPv4, die sogenannte *IPv4-Adressen*, sind *nicht mehr vorhanden*; dagegen sind die Adressen des Protokolls IPv6, kurz *IPv6-Adressen* genannt, *reichlich vorhanden*, und mit ihnen kann eine enorme Vielzahl von Sensoren und Aktuatoren adressiert werden.

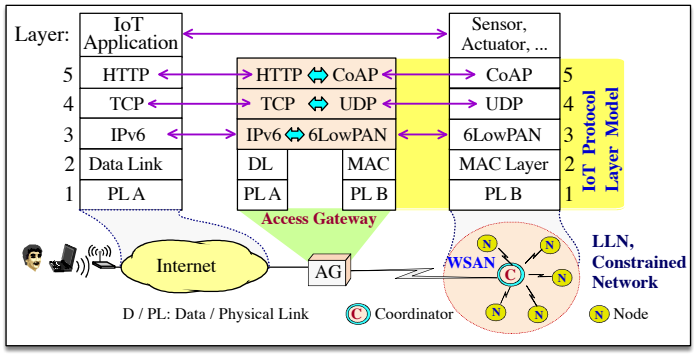


Abb. 17.1-11: Protokollarchitektur und wesentlichen Aufgaben der IoT Access Gateways
6LoWPAN: IPv6 over Low-power WPAN, DL: Data Link, HTTP: Hypertext Transfer Protocol, PL: Physical Layer, WSAN: Wireless Sensor Actuator Network

Mapping zwischen IPv6 und 6LoWPAN

Da drahtlose Sensor-Aktuator-Netzwerke im IoT die sogenannte LLNs (Low Power and Lossy Networks) darstellen, muss IPv6 für den Einsatz in LLNs entsprechend adaptiert werden – insbesondere muss der Header vom IPv6 stark komprimiert werden. Die für den Einsatz von IPv6 in LLNs realisierte Adaption von IPv6 wird 6LoWPAN (*IPv6 over Low-power Wireless Personal Area Network*) genannt. Somit enthält die Schicht 3 in Sensor-Aktuator-Netzen, das Protokoll 6LoWPAN – also das 'magere' IPv6. In Access Gateways muss man dieses 'magere' Protokoll bei der Übermittlung von Daten an das herkömmliche Internet zum normalen Protokoll IPv6 'ausbauen'. Bei der Übermittlung von Daten in Gegenrichtung wird dann das IPv6 zum 6LoWPAN 'abgespeckt'; also quasi ein *Mapping zwischen IPv6 und 6LoWPAN*.

UDP in LLNs

In Sensor-Aktuator-Netzen, welche LLNs sind, werden einerseits keine großen Dateien transportiert, und andererseits ist das verbindungsorientierte und zuverlässige Transportprotokoll TCP für den Einsatz in LLNs nicht geeignet – denn es ist für LLNs zu komplex. In LLNs innerhalb der Schicht 4 (Transportschicht) kann daher nur das verbindungslose und unzuverlässige Transportprotokoll UDP verwendet werden.

CoAP statt HTTP im IoT

IoT-Applikationen seitens des herkömmlichen Internet verwenden das Webprotokoll HTTP (*Hypertext Transfer Protocol*). Dieses ist komplex und eignet sich somit nicht zum Einsatz in LLNs – also in Sensor-Aktuator-Netzen. Aus diesem Grund wurde von der IETF Working Group CoRE (*Constrained RESTful Environment*) das Protokoll CoAP (*Constrained Application Protocol*) spezifiziert.

Mapping zwischen HTTP und CoAP

Wie in Abb. 17.1-11 dargestellt, dient das CoAP als Application Support Protocol in Sensor-Aktuator-Netzen. In Access Gateways innerhalb der Schicht 5, die als Application Support Layer angesehen werden kann, muss daher eine Übersetzung (Translation) HTTP ⇔ CoAP, also als *Mapping zwischen HTTP und CoAP* realisiert werden.

17.1.10 Struktur von MAC-Frames in Low Rate WPANs

MAC-Frame nach lEEE 802.15.4

Wie in Abb. 17.1-10 gezeigt, stellt der Layer 2 in der Protokollarchitektur von IoT Devices oft den MAC-Layer (*Medium Access Control*) nach dem Standard IEEE

802.15.4 mit der Spezifikation von LR-WPAN (*Low-Rate Wireless Personal Area Networks*) dar. Demzufolge entspricht oft die Struktur von MAC-Frame in drahtlose Sensor-Aktuator-Netzwerke, die sogenannte LLNs sind, im IoT oft vollkommen der Struktur von MAC-Frames nach dem Standard lEEE 802.15.4. Abb. 17.1-12 illustriert die Struktur und den Inhalt von diesen MAC-Frames.

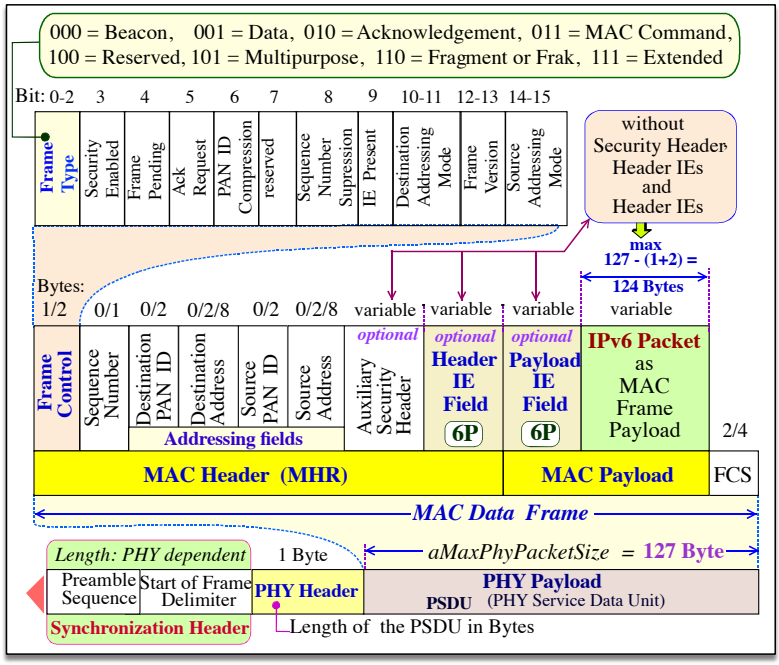


Abb. 17.1-12: Struktur und Inhalt von MAC-Frames in LR-WPANs nach dem Standard IEEE 802.15.4 aus dem Jahr 2015 FCS: Frame Check Sequence, IE: Information Element, MAC: Medium Access Control, PAN ID: Personal Area Network Identifier, PHY: Physical Layer, Max. Länge von MAC Frames auf 127 Byte begrenzt

Insbesondere wird hier zum Ausdruck gebracht, dass die maximale Länge von PHY-Payload, d.h. die maximale Länge von MAC Data Frames mit den in ihnen eingekapselten IPv6-Paketen, als deren Nutzlast (PHY-Payload), auf 127 Byte begrenzt ist. Demzufolge ist die Komprimierung von Overhead in IPv6-Paketen, also des IPv6- und des UDP-Header, in LR-WPANs mit den Physical Layer und MAC Layer nach dem Standard IEEE 80.15.4 unabdingbar. Aus diesem Grund wird als Protokoll auf dem Layer 3 im IoT auf Basis von diesen LR-WPANs das 'compressed IPv6', d.h. *6LoWPAN*, eingesetzt.

Diverse MAC Control Frames

Der Header in MAC-Frames wird mit dem Ziel strukturiert, eine entsprechende Identifizierung unterschiedlicher Arten von Steuerungsangaben als besondere Nutzlast und ihre Übermittlung in MAC-Frames zu ermöglichen und dadurch unterschiedliche Arten von MAC-Frames zu 'erzeugen'. Man spricht in diesem Zusammenhang von MAC *Control Frames* und unterscheidet dabei zwischen verschiedenen Typen. Um welchen Typ Control Frame als Payload in einem MAC-Frame es sich handelt, wird innerhalb des MAC-Headers im Feld Frame Control als Frame Type eingetragen. Auf

welche Weise dies markiert wird, ist in Abb. 17.1-12 ersichtlich. So steht beispielsweise `000` für *Beacon Frames*, `001` für *Data Frames*, `010` für *Acknowledgement* und `011` für *MAC Command Frame*.

Bedeutung von `IE` Fields

Bei einigen Protokollen, insbesondere bei 6P (*6top Protocol*) im *Industrial IoT* (IIoT) mit dem MAC Layer nach dem Konzept TSCH (*Time-Slotted Channel Hopping*) spielen die in den IE Fields (`IE`: *Information Element*) von MAC-Frames übermittelten IEs eine besondere Rolle. Sie dienen quasi als Container zur Übermittlung diverser Steuerungsangaben und auch Nutzdaten, d.h. Payload. In einem MAC-Frame können mehrere, diesem Zweck dienende IEs übermittelt werden. Wie Abb. 17.1-12 illustriert, kann in einem MAC-Frame enthalten sein:

- ein *Header* `IE` *Field*, oft mit mehreren Header IEs, als Teil des MAC-Headers und
- ein *Payload* `IE` *Field*, oft mit mehreren Payload IEs, als MAC-Payload.

Auf diese Art und Weise werden verschiedene IEs zur Übermittlung von Nachrichten mit Daten und Steuerungsangaben zwischen speziellen Steuerungsinstanzen auf dem MAC Layer eingesetzt.

Zwei Kategorien von MAC-Frames

In einigen IoT-Netzwerken zwischen jeweils zwei kommunizierenden Nodes können quasi zwei Kategorien von MAC-Frames übermittelt werden. Zur ersten Kategorie gehören die MAC-Frames, in denen die komprimierten IPv6-Pakete enthalten sind. Die MAC-Frames dieser Kategorie übermitteln Anwendungsdaten zwischen Applikationen in beiden kommunizierenden Nodes. Zur zweiten Kategorie gehören die MAC-Frames, in denen Payload IEs enthalten sind. Die MAC-Frames dieser Kategorie übermitteln in Payload IEs die Daten zwischen den innerhalb des MAC Layer angesiedelten Steuerungsinstanzen.

17.2 6LoWPAN – IPv6-Adaption für das IoT

Notwendigkeit der IPv6-Adaption

Die drahtlosen Sensor-Aktuator-Netze im IoT werden häufig als sogenannte LR-WPANs (*Low-Rate Wireless Personal Area Networks*) nach dem IEEE-Standard 802.15.4 eingerichtet. Dieser Standard stellt wichtige Anforderungen, die in drahtlosen Sensor-Aktuator-Netzen mit energiearmen Komponenten eingehalten werden müssen. Insbesondere darf die maximale Länge der in LR-WPANs übermittelten MAC Data Frames mit den in ihnen eingekapselten IPv6-Paketen nur 127 Byte betragen. Im IPv6-Standard werden aber IPv6-Pakete bis zur Länge von 1280 Byte zugelassen. Aus diesem Grund müssen die IPv6-Pakete für die Übermittlung in energiearmen und mit Datenverlust behafteten LR-WPANs entsprechend angepasst und auch komprimiert werden. Folglich ist eine Adaptation des Protokolls IPv6 an die Anforderungen solcher LR-WPANs unabdingbar. Diese IPv6-Adaptation bezeichnet man als *IPv6 over Low-power Wireless Personal Area Network*, kurz *6LoWPAN*.

Ein äußerst wichtiger Teil der Adaption des Protokolls IPv6 zum Einsatz in energiearmen und mit Datenverlust behafteten WPANs ist die Komprimierung von IPv6-Paketen und dabei auch die Fragmentierung der größeren Pakete [Abschnitt 17.2.9].

18 Networking-Trends

Networking Trends 2019 im Überblick

Die wichtigsten Meilensteine der Internet-Entwicklung bis Ende 2014, d.h. bis zum Zeitpunkt der Erstellung der 3. Auflage dieses Buches, präsentiert der damals verfasste Abschnitt 1.1.4. In darauffolgenden vier Jahren haben sich bedeutende Entwicklungstrends auf dem Gebiet Networking, auf dem das Protokoll IP in beiden Versionen 4 und 6 eine fundamentale Rolle spielt, herauskristallisiert. Das Ziel dieses Abschnittes ist es, die aktuellen, in Abb. 18.0-1 dargestellten 'Top 10 Networking Trends' näher zu erläutern.

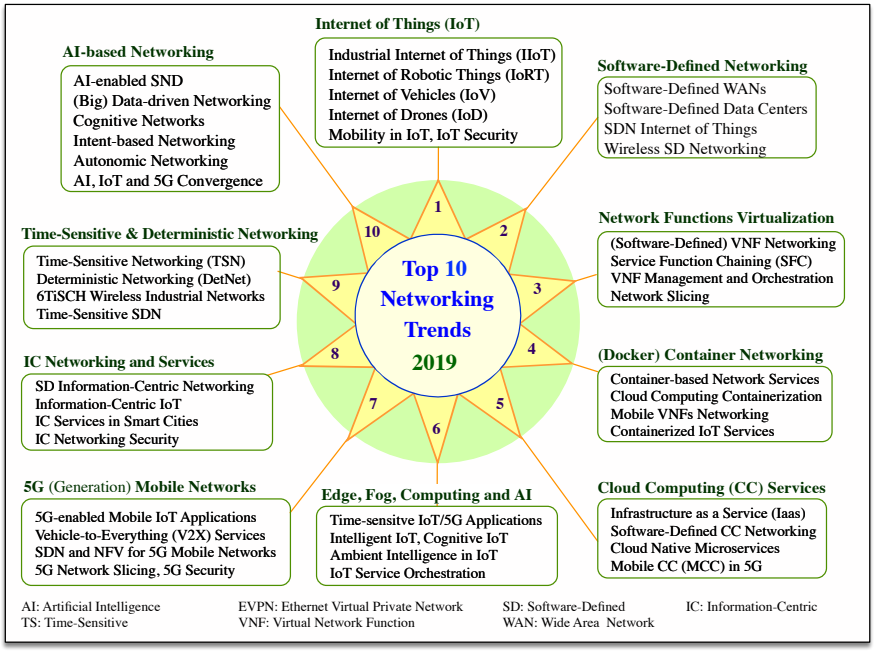


Abb. 18.0-1: Internet, Internet of Things und andere IP-Netze; Top 10 Networking Trends

Die aktuellen 'Top 10 Networking Trends' wollen wir nun abschließend kurz beleuchten. Da diese **Entwicklungen** derzeit in Fluss sind, können die Resultate daher nicht in der notwendigen fachlichen Durchdringung dargestellt werden. Allerdings kann sich jeder Leser mittels der angegebenen Quellen ein Bild des aktuellen Standes bilden, wobei natürlich auch dann im Verborgenen bleibt, welche der aufgezeigten Tendenzen tatsächlich eingesetzt werden.

18.1 Internet of Things (IoT)

IoT-betreffende Networking Trends

Die Integration verschiedener allgegenwärtiger, zu unserem Alltag gehörender Dinge (*Things*) in das Internet führt zur Entstehung einer besonderen Internet-Erweiterung. Diese wird als *Internet of Things* (IoT) bzw. als *Internet der Dinge* bezeichnet. Das Konzept von IoT und seine Protokolle präsentiert das Kapitel 17. An dieser Stelle sei aber darauf hingewiesen und dies geht aus Abb. 1.1-6 hervor, dass diverse Networking Trends auf eine Art mit dem IoT zusammenhängen und folglich auch seine weiteren Entwicklungen im gewissen Grade beeinflussen. Die folgenden Trends in der Weiterentwicklung von IoT sind hervorzuheben: *Industrial IoT* (IIoT), *Internet of Robotic Things* (IoRT), *Internet of Vehicles* (IoV), *Internet of Drones* (IoD), *Mobility in IoT* und *IoT Security*. Für Näheres darüber siehe:

1.a i-SCOOP: The Internet of Things (IoT) – essential IoT business guide

1.b i-SCOOP: Blockchain and the Internet of Things: the IoT blockchain opportunity and challenge

1.c Jorge Granjal, Edmundo Monteiro, Jorge Sá Silva: Security for the Internet of Things: A Survey of Existing Protocols and Open Research Issues; IEEE Communications Surveys & Tutorials, Vol. 17(3), Jul 2015; DOI: 10.1109/COMST.2015.2388550

1.c Misty Blowers, Jose Iribarne, Edward Colbert, Alexander Kott: The Future Internet of Things and Security of its Control Systems; arXiv:1610.01953v1; Oct 2016

1.d P.P. Ray: A survey on Internet of Things architectures; Journal of King Saud University - Computer and Information Sciences; Vol. 30 (3), Jul 2018, DOI: 10.1016/j.jksuci.2016.10.003

1.e Onoriode Uviase, Gerald Kotonya: IoT Architectural Framework: Connection and Integration Framework for IoT Systems; arXiv:1803.04780v1, 2018, DOI: 10.4204/EPTCS.264.1

1.f Alem Čolaković, Mesud Hadžialić: Internet of Things (IoT): A Review of Enabling Technologies, Challenges, and Open Research Issues; Computer Networks, Vol. 144, Oct 2018; DOI: 10.1016/j.comnet.2018.07.017

18.1.1 Industrial Internet of Things (IIoT)

Das IoT hat eine enorme Bedeutung für die Industrie, denn Dank ihm – insbesondere der Nutzung von Sensordaten und der M2M-Kommunikation (M2M: *Machine to Machine*) kann die Effizienz von Maschinen und Automaten verbessert werden. Die M2M-Kommunikation stellt hohe QoS-Anforderungen (QoS: *Quality of Service*) an die industriellen Netzwerke und folglich an das IoT. Zu ihrer Erfüllung ist eine besondere Realisierung des IoT erforderlich. Diese an die industriellen Anforderungen angepasste Art von IoT wird als IIoT (*Industrial IoT*) bezeichnet. Für Näheres über IIoT siehe: [1.1.a], [1.1.c] und [1.1.h].

IIoT und TSCH

Um hohe QoS-Anforderungen erfüllen zu können, müssen für das IIoT spezielle Netzwerke eingerichtet werden. Zu deren Aufbau wurde das Verfahren TSCH (*Time-Slotted Channel Hopping*) konzipiert. Es stellt eine spezielle zeitsynchrone Realisierung des MAC-Protokolls (*Media Access Control*) zum Aufbau drahtloser industrieller Netzwerke dar. Bei TSCH handelt sich um eine Kombination des Frequenzmultiplexes und des synchronen Zeitmultiplexes, also um eine Variante des Frequenz- und Zeitmultiplexverfahrens FTDM (*Frequency-Time Division Multiplexing*). Die auf TSCH basierenden Netzwerke ermöglichen es im industriellen Bereich, das sog. *Deterministic Networking* zu realisieren. Für weitere Informationen über IIoT und TSCH sei verwiesen auf:

1.1.a i-SCOOP: The Industrial Internet of Things (IIoT): the business guide to Industrial IoT

1.1.b i-SCOOP: IT and OT convergence – two worlds converging in Industrial IoT

1.1.c Li Da Xu, Wu He, Shancang Li: Internet of Things in Industries: A Survey; IEEE Transactions on Industrial Informatics“, Vol. 10(4) , Nov 2014; DOI: 10.1109/TII.2014.2300753

1.1.c Martin Wollschlaeger, Thilo Sauter, Jürgen Jasperneite: The Future of Industrial Communication: Automation Networks in the Era of the Internet of Things and Industry 4.0; IEEE Electronics Magazine, Mar 2017

1.1.d Anatol Badach: TSCH – Time-Slotted Channel Hopping; In book: Protokolle und Dienste der Informationstechnologie; WEKA, Ed.: Heinz Schulte; Jan 2018

1.1.f Jiangfeng Cheng, Weihai Chen, Fei Tao, Chun-Liang Lin: Industrial IoT in 5G environment towards smart manufacturing; Journal of Industrial Information Integration, Vol. 10, Jun 2018, DOI: 10.1016/j.jii.2018.04.001

1.1.g Emiliano Sisinni, Abusayeed Saifullah, Song Han, Ulf Jennehag, Mikael Gidlund: Industrial Internet of Things: Challenges, Opportunities, and Directions; IEEE Transactions on Industrial Informatics (Early Access), Jul 2018; DOI: 10.1109/TII.2018.2852491

1.1.h Hugh Boyes, Bil Hallaq, Joe Cunningham, Tim Watson: The industrial internet of things (IIoT): An analysis framework; Computers in Industry, Vol. 101, Oct 2018; DOI: 10.1016/j.compind.2018.04.015

18.1.2 Internet of Robotic Things

Dank der modernsten Technologien werden mehr und mehr Roboter überall, vor allem in der Industrie, eingesetzt. Sie werden auch auf eine spezielle Art und Weise in das IoT integriert, und dies führt zur Entstehung einer besonderen IoT-Art, die als *Internet of Robotic Things* (IoRT) bezeichnet wird. Mit dem IoRT wird das Ziel verfolgt, sowohl diverse Roboter untereinander als auch weit im IoT verbreitete Sensoren und als Aktuatoren bezeichnete intelligente Antriebselemente mit autonomen Robotern zu vernetzen. Diese Vernetzung bedeutet eine Verschmelzung von Robotik- und IoT-Technologien und ermöglicht, die IoT-Services um die Fähigkeiten intelli-

genter Robotersysteme zu erweitern. Dadurch kann ein neues breites Spektrum von IoRT-basierten Services angeboten werden. Für Näheres über IoRT siehe:

1.2.a Partha Pratim Ray: Internet of Robotic Things: Concept, Technologies, and Challenges; IEEE Access, Vol. 4, 2016; DOI: 10.1109/ACCESS.2017.2647747

1.2.c Cristanel Razafimandimby, Valeria Loscri, Anna Maria Vegni: A neural network and IoT based scheme for performance assessment in Internet of Robotic Things; 2016 IEEE First International Conference on Internet-of-Things Design and Implementation (IoTDI), Apr 2016; DOI: 10.1109/IoTDI.2015.10

1.2.d Cristanel Razafimandimby, Valeria Loscri, Anna Maria Veg: Towards Efficient Deployment in Internet of Robotic Things; In book: Gravina R., et al. (Eds) 'Integration, Interconnection, and Interoperability of IoT Systems. Internet of Things (Technology, Communications and Computing)'. Springer, Jul 2017; DOI: 10.1007/978-3-319-61300-0_2

1.2.e Pieter Simoens, Mauro Dragone, Alessandro Saffiotti: The Internet of Robotic Things: A review of concept, added value and applications; International Journal of Advanced Robotic Systems, Vol. 15(1), Jan - Feb 2018; DOI: 10.1177/1729881418759424

18.1.3 Internet of Vehicles

Zur Unterstützung autonom fahrender Fahrzeuge mithilfe von IoT müssen im IoT verschiedene zeitkritische Services realisiert werden. Die Unterstützung des autonomen Fahrens führt daher zur Entstehung einer besonderen, als *Internet of Vehicles* (IoV) bezeichneten Variante von IoT. Dabei kann Fog Computing [Abschnitt 17.1] als Basis für das IoV angesehen werden. Im IoV sollen verschiedene Arten der drahtlosen Vernetzung von autonom fahrenden Fahrzeugen realisiert werden, insbesondere:

V2V
- *Vehicle-to-Vehicle*: Fahrzeug-zu-Fahrzeug-Kommunikation ermöglicht, dass autonom fahrende, benachbarte Fahrzeuge untereinander kommunizieren und sich damit z.B. über den Straßenzustand informieren können.

V2I
- *Vehicle-to-Infrastructure*: Mittels Fahrzeug-zu-Infrastruktur-Kommunikation können autonom fahrende Fahrzeuge auf straßenseitige Infrastrukturelemente (z.B. entlang von Straßen installierte Verkehrsleitsysteme) wie z.B. Ampeln und andere Verkehrszeichen aufmerksam gemacht werden. Hiermit kann die Verkehrssicherheit verbessert werden.

V2P
- *Vehicle-to-Pedestrian*: Fahrzeug-zu-Fußgänger-Kommunikation kann als präventive Maßnahme betrachtet werden, bei dem ein autonom fahrendes Fahrzeuge Fußgänger und Radfahrer erkennen kann, um folglich eine potenzielle Kollision zu verhindern und Schaden von den Verkehrsteilnehmern abzuwenden.

V2N
- *Vehicle-to-Network*: Fahrzeug-zu-Netzwerk-Kommunikation erlaubt z.B. über Mobilfunk Verbindungen mit großer Reichweite zwischen Fahrzeug und Netzwerk, um z.B. einen Cloud-Zugang aufbauen zu können.

Für weitere Informationen über IoV siehe:

1.3.a Yang Fangchun, Wang Shangguang, Li Jinglin, Liu Zhihan, Sun Qibo: An Overview of Internet of Vehicles; China Communications, Oct 2014

1.3.b Kang Kai, Wang Cong and Luo Tao: Fog Computing for Vehicular Ad Hoc Networks: Paradigms, Scenarios, and Issues; The Journal of China Universities of Posts and Telecommunications,Vol. 23(2), Apr 2016; DOI 10.1016/S1005-8885(16)60021-3

1.3.c Omprakash Kaiwartya, et al.: Internet of Vehicles: Motivation, Layered Architecture, Network Model, Challenges, and Future Aspects; IEEE Access, Vol. 4, Sep 2016; DOI: 10.1109/ACCESS.2016.2603219

1.3.f Juan Contreras, Sherali Zeadally, Juan Antonio Guerrero-Ibanez: Internet of Vehicles: Architecture, Protocols and Security; IEEE Internet of Things Journal (Early Access), Apr 2017; DOI: 10.1109/JIOT.2017.2690902

1.3.g Fangchun Yang, Jinglin Li, Tao Lei, Shangguang Wang: Architecture and key technologies for Internet of Vehicles: a survey; Networks, Vol. 2(2), Jun 2017; DOI: 10.1007/s41650-017-0018-6

1.3.h Eugen Borcoci, Serban Obreja, Marius Vochin: Internet of Vehicles Functional Architectures-Comparative Critical Study, The Ninth International Conference on Advances in Future Internet AFIN, Sep 2017

1.3.i Min Chen, Yuanwen Tian, Giancarlo Fortino, Jing Zhang, Iztok Humar: Cognitive Internet of Vehicles, Computer Communications; Vol. 120, May 2018; DOI: 10.1016/j.comcom.2018.02.006

18.1.4 Internet of Drones

Die Drohnen, unbemannte und kurz als UAVs (*Unmanned Aerial Vehicle*) bezeichnete Luftfahrzeuge, gewinnen immer mehr an Bedeutung. Sie können ohne eine an Bord befindliche menschliche Besatzung selbständig durch einen Computer oder vom Boden über eine Fernsteuerung navigiert und für verschiedene Zwecke eingesetzt werden. Aus diesem Grund werden Drohnen als mobile Objekte in das Internet, insbesondere in das IoT, integriert. Dadurch entsteht eine besondere Art von IoT, und man nennt diese *Internet of Drones* (IoD). Als Literatur über IoD siehe:

1.4.a Ilker Bekmezci, Ozgur Koray Sahingoz, Samil Temel: Flying Ad-Hoc Networks (FANETs): A Survey; Vol. 11(3), May 2013; DOI: 10.1016/j.adhoc.2012.12.004

1.4.b Ozgur Koray Sahingoz: Networking Models in Flying Ad-Hoc Networks (FANETs): Concepts and Challenges; J. Intelligent & Robotic Systems, Vol. 74 (1), Apr 2014; DOI 10.1007/s10846-013-9959-7

1.4.c Naser Hossein Motlagh, Tarik Taleb, Osama Arouk: Low-Altitude Un-manned Aerial Vehicles-Based Internet of Things Services: Comprehensive Survey and Future Perspectives; IEEE Internet of Things Journal, Vol. 3(6), Dec 2016; DOI: 10.1109/JIOT.2016.2612119

1.4.d Armir Bujari, et al.: Flying ad-hoc network application scenarios and mobility models; International Journal of Distributed Sensor Networks; Vol. 13(10), 2017; DOI: 10.1177/1550147717738192

1.4.e Mariana Rodrigues, et al.: UAV Integration Into IoIT: Opportunities and Challenges; ICAS 2017: The Thirteenth International Conference on Autonomic and Autonomous Systems, May 2017

1.4.f Muhammad Asghar Khan, Ijaz Mansoor Qureshi, Engr Alamgir Safi, Inam Ullah Khan: Flying Ad-Hoc Networks (FANETs): A Review of Communication architectures, and Routing protocols; First International Conference on Latest trends in Electrical Engineering and Computing Technologies (Intellect 2017); Nov 2017; DOI: 10.1109/INTELLECT.2017.8277614

1.4.g Weisen Shi, et al.: Drone Assisted Vehicular Networks: Architecture, Challenges and Opportunities; IEEE Network, Vol.32(3), May/Jun 2018; DOI: 10.1109/MNET.2017.1700206

1.4.h Gaurav Choudhary, Vishal Sharma, Takshi Gupta, Jiyoon Kim, Ilsun You: Internet of Drones (IoD): Threats, Vulnerability, and Security Perspectives; arXiv:1808.00203v2, Aug 2018

1.4.i Mohammad Wazid, Ashok Kumar Das, Jong Hyouk Lee: Authentication protocols for the internet of drones: taxonomy, analysis and future directions; Journal of Ambient Intelligence and Humanized Computing; Aug 2018; DOI: 10.1007/s12652-018-1006-x

18.1.5 Mobility in IoT

An das IoT wird u.a. eine breite Palette von mobilen *Sensoren*, *Aktuatoren* und mobile virtualisierte Rechner in Form von VMs (*Virtual Machines*) angebunden. Diese können z.B. in selbstfahrenden Autos, in Drohnen bzw. in anderen UAVs installiert sein. Mit mobilen VMs am IoT-Rande (IoT Edge) werden verschiedene mobile Services erbracht, sodass man von *Mobile Edge Computing* (MEC) spricht. Mobile VMs können aber auch als mobile Fog Nodes fungieren [Abschnitt 17.1]. Dies deutet darauf hin, dass die Unterstützung der Mobilität im IoT (Mobility in IoT) von enorm großer Bedeutung ist. Für Näheres über Mobilität im IoT sei verwiesen auf:

1.5.a Rodrigo Roman, Javier Lopez, Masahiro Mambo: Mobile edge computing, Fog et al.: A survey and analysis of security threats and challenges; arXiv:1602.00484v2, Nov 2016; DOI: 10.1016/j.future.2016.11.009

1.5.b Safwan M. Ghaleb, et al.: Mobility management for IoT: a survey; EURASIP Journal on Wireless Communications and Networking, Dec 2016; DOI 10.1186/s13638-016-0659-4

1.5.c Jabiry M. Mohammed, Bi-Lynn Ong, R. Badlishan Ahmad, Mohammed Hakawati: Internet of Things (IoT) Mobility Support based on distributed Sensor Proxy MIPv6; Journal of Theoretical and Applied Information Technology, Vol. 95(17), Sept. 2017

1.5.d Tuan Nguyen gia, et al.: Fog Computing Approach for Mobility Support in Internet-of-Things Systems; IEEE Access PP(99):1-1; Jun 2018, DOI:10.1109/ACCESS.2018.2848119

18.1.6 IoT Security

Mit IoT soll die Vision verwirklicht werden, unsere Alltagseinrichtungen unterschiedlicher Art und mit unterschiedlichen Fähigkeiten sowohl untereinander als auch mit Rechnern am Internet so zu vernetzen, dass sie alle Internet-Services nutzen können. Mit der Verwirklichung dieser Vision entstehen aber verschiedene potenzielle Sicherheitsbedrohungen, dabei aber auch solche, die wir heute noch nicht kennen. Somit ist die IoT-Sicherheit (IoT Security) von enorm großer Bedeutung. Um die IoT-Sicherheit möglichst auf einem hohen Niveau zu gewährleisten, sind klassische Ansätze nicht mehr ausreichend. Hierfür ist zurzeit noch ein breites Spektrum von Forschungs- und Entwicklungsaktivitäten erforderlich. Vor allem müssen unsere Alltagseinrichtungen am IoT eine besondere sicherheitsrelevante Intelligenz besitzen, also sicherheitsbewusst (*security aware*) sein. Für Näheres über IoT-Sicherheit siehe:

1.6.a T. Heer, et al.: Security Challenges in the IP-based Internet of Things, Wireless Personal Communications; Vol. 61(3), Dec 2011; DOI: 10.1007/s11277-011-0385-5

1.6.b Sabrina Sicari, et al.: Security, Privacy & Trust in Internet of Things: The road ahead; Computer Networks Vol. 76, Jan 2015; DOI: 10.1016/j.comnet.2014.11.008

1.6.c Jorge Granjal, Edmundo Monteiro, Jorge Sá Silva: Security for the Internet of Things: A Survey of Existing Protocols and Open Research Issues; IEEE Communications Surveys & Tutorials, Vol. 17(3), Jul 2015; DOI: 10.1109/COMST.2015.2388550

1.6.e M. A. Ferrag, L. A. Maglaras, H. Janicke , J. Jiang: Authentication Protocols for Internet of Things: A Comprehensive Survey, arXiv, Dec 2016

1.6.e Rodrigo Roman, Javier Lopez, Masahiro Mambo: Mobile edge computing, Fog et al.: A survey and analysis of security threats and challenges; arXiv:1602.00484v2, Nov 2016; DOI: 10.1016/j.future.2016.11.009

1.6.e Shancang Li, Li Da Xu, Imed Romdhani (Contributor): Securing the Internet of Things: 2017 Elsevier Inc.; ISBN: 978-0-12-804458-2

1.6.f Xiruo Liu, Meiyuan Zhao, Sugang Li, Feixiong Zhang, Wade Trappe: A Security Framework for the Internet of Things in the Future Internet Architecture; Future Internet, Vol. 9(3), 2017; DOI: 10.3390/fi9030027

1.6.g B. B. Zarpelão, R. S. Miani, C. T. Kawakani and S. C. de Alvarenga: A Survey of Intrusion Detection in Internet of Things, Journal of Network and Computer Applications; Vol. 84, Apr 2017; DOI: 10.1016/j.jnca.2017.02.009

1.6.h Pooja, Dr. R. K. Chauhan: Review on Security Attacks and Countermeasures in Wireless Sensor Networks; International Journal of Advanced Research in Computer Science, Vol. 8(5), May-Jun 2017

1.6.i Chao Lin, et al.: Security and Privacy for the Internet of Drones: Challenges and Solutions; IEEE Communications Magazine, Vol. 56(1), Jan 2018; DOI: 10.1109/MCOM.2017.1700390

1.6.j Djedjig Nabil, D. Tandjaoui, Imed Romdhani, Faiza Medjek: Trust Management in Internet of Things; In book: Security and Privacy in Smart Sensor Networks; May 2018; DOI: 10.4018/978-1-5225-5736-4.ch007

1.6.k Francesco Restuccia, Salvatore D'Oro, Tommaso Melodia: Securing the Internet of Things in the Age of Machine Learning and Software-defined Networking; IEEE Internet of Things Journal (Early Access), Jun 2018; DOI: 10.1109/JIOT.2018.2846040

1.6.l Djamel Eddine Kouicem, Abdelmadjid Bouabdallah, Hicham Lakhlef: Internet of things security: A top-down survey; Computer Networks, Vol. 141, Aug 2018; DOI: 10.1016/j.comnet.2018.03.012

1.6.m Shui Yu, Guojun Wang, Xiting Liu, Jianwei Niu: Security and Privacy in the Age of the Smart Internet of Things: An Overview from a Networking Perspective; IEEE Communications Magazine, Vol. 56(9), Sep 2018; DOI: 10.1109/MCOM.2018.1701204

18.2 Software-Defined Networking (SDN)

Programmable Network Services

Die grundlegende Idee von SDN besteht darin [Abschnitt 17.1.7], die Software in zur Übermittlung von Daten bestimmten Netzwerkkomponenten möglichst von ihrer Hardware zu trennen und diese dann auf eine bzw. mehrere zentrale, als Controller bezeichnete Steuerungskomponente/n auszulagern. Dies führt zu neuen Möglichkeiten der Entwicklung, de facto Programmierung, diverser Network Services. SDN ermöglicht auf diese Weise eine dem Bedarf entsprechende, schnelle Einrichtung verschiedener SD Networks und SD Network Services. Folglich kann sogar von *Programmable Network Services* bzw. von *Network Programmability* gesprochen werden. SDN liegt insbesondere den Networking Trends *SD WANs*, *SD Data Centers*, *SD IoT*, *SD Mobile Networking* und *Wireless SD Networking* zugrunde. Für Näheres über SDN siehe:

2.a Anatol Badach: SDN – Software Defined Networking; In book: Protokolle und Dienste der Informationstechnologie; WEKA, Ed.: Heinz Schulte; Dec 2012, DOI 10.13140/RG.2.1.4963.4001

2.b Sandra Scott-Hayward, Gemma O'Callaghan, Sakir Sezer: SDN Security: A Survey; IEEE SDN for Future Networks and Services (SDN4FNS), Nov 2013; DOI: 10.1109/SDN4FNS.2013.6702553

2.c Yosr Jarraya, Taous Madi, Mourad Debbabi: A Survey and a Layered Taxonomy of Software-Defined Networking; IEEE Communications Surveys & Tutorials, Vol. 16(4), 2014; DOI: 10.1109/COMST.2014.2320094

2.d RFC 7426: Software-Defined Networking (SDN): Layers and Architecture Terminology; Jan 2015

2.e Hamid Farhady, HyunYong Lee, Akihiro Nakao: Software-Defined Networking: A survey; Computer Networks, Vol. 81, Apr 2015; DOI: 10.1016/j.comnet.2015.02.014

2.f Akram Hakiri, et al.: Software-Defined Networking: Challenges and research opportunities for Future Internet; Computer Networks, Vol. 75, Part A, Dec 2014; DOI: 10.1016/j.comnet.2014.10.015

2.g Bruno Astuto A. Nunes, et al.: A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks; IEEE Communications Surveys & Tutorials, Vol. 16(3), 2014; DOI: 10.1109/SURV.2014.012214.00180

2.h Diego Kreutz, et al.: Software-Defined Networking: A Comprehensive Survey; Proceedings of the IEEE, Vol. 103(1), Jan 2015; DOI: 10.1109/JPROC.2014.2371999

2.i Junjie Xie, Deke Guo, Zhiyao Hu, Ting Qu, Pin Lv: Control plane of software defined networks: A survey; Computer Communications, Vol. 67, Aug 2015, DOI: 10.1016/j.comcom.2015.06.004

2.j Izzat Alsmadi, Dianxiang Xu: Security of Software Defined Networks: A survey; Computers & Security, Vol. 53, Sep 2015; DOI: 10.1016/j.cose.2015.05.006

2.k ONF TR-521: SDN Architecture; Issue 1.1; 2016

2.l Rahim Masoudi, AliGhaffari: Software defined networks: A survey; Journal of Network and Computer Applications, Vol. 67, May 2016; DOI: 10.1016/j.jnca.2016.03.016

2.m Taimur Bakhshi: State of the Art and Recent Research Advances in Software Defined Networking; Wireless Communications and Mobile Computing, Vol. 2017, Article ID 7191647; Jan 2017; DOI: 10.1155/2017/7191647

2.n Murat Karakus, Arjan Durresi: Quality of Service (QoS) in Software Defined Networking (SDN): A survey; Journal of Network and Computer Applications, Vol. 80, Feb 2017; DOI: 10.1016/j.jnca.2016.12.019

2.o Mehmet Fatih Tuysuz, Zekiye Kubra Ankarali, Didem Gözüpek: A survey on energy efficiency in software defined networks; Computer Networks, Vol. 113, Feb 2017; DOI: 10.1016/j.comnet.2016.12.012

2.p Sanjev Singh, Rakesh Kumar Jha: A Survey on Software Defined Networking: Architecture for Next Generation Network, Journal of Network and Systems Management, 2017, Vol. 25, DOI 10.1007/s10922-016-9393-9

2.q Fetia Bannour, Sami Souihi, Abdelhamid Mellouk: Distributed SDN Control: Survey, Taxonomy and Challenges; IEEE Communications Surveys and Tutorials, 2017, DOI 10.1109/COMST.2017.2782482

2.r Yustus Eko Oktian, SangGon Lee, HoonJae Lee, JunHuy Lam: Distributed SDN controller system: A survey on design choice; Computer Networks, Vol. 121, Jul 2017; DOI: 10.1016/j.comnet.2017.04.038

2.s Sibylle Schallera, Dave Hoodb: Software defined networking architecture standardization; Computer Standards & Interfaces, Vol. 54, 2017; DOI: 10.1016/j.csi.2017.01.005

2.t Michel S. Bonfim, Kelvin L. Dias, Stenio F. L. Fernandes: Integrated NFV/SDN Architectures: A Systematic Literature Review; Jan 2018; arXiv:1801.01516v1

2.u Rashid Amin, Martin Reisslein, Nadir Shah: Hybrid SDN Networks: A Survey of Existing Approaches; IEEE Communications Surveys & Tutorials, Vol. 20(4), May 2108; DOI: 10.1109/COMST.2018.2837161

18.2.1 SD WANs

Das Konzept von SDN kann zur flexiblen Gestaltung und zum Management privater WANs eingesetzt werden, um diese schnell an aktuelle Anforderungen, z.B. im Hinblick auf die Erfüllung von QoS (*Quality of Service*) und Garantie der Sicherheit, anpassen zu können. In diesem Zusammenhang spricht man von *Software-Defined WANs* (SD-WANs). Ein SD-WAN kann vereinfacht als Software-basierte Lösung für WANs angesehen werden. Zum Einrichten eines SD-WAN für den Transport von IP-Datenpaketen können unterschiedliche Netzwerktechnologien eingesetzt werden. Dabei erfolgt die Konfiguration und Steuerung der Funktionen und Leistungsmerkmale über eine oder mehrere Software-Instanzen innerhalb einer speziellen, den Hardware-Komponenten übergeordneten SD-WAN Control Plane. Für weitere Information über SD WANs siehe:

2.1.a Happiest Minds Technologies Pvt. Ltd.; Author: Purnendu: SDWAN: Re-architecting WAN with Software Defined Networking

2.1.b CATO-Networks: MPLS, SD-WAN, Internet, and Cloud Network; Understanding the Trade-offs for Your Next Generation WAN

2.1.c pwc: SD-WAN for service providers; Threat or opportunity?

2.1.d ONUG-SD-WAN-WG-Whitepaper: Software-Defined WAN Use Case; Oct 2014

2.1.d Sanjay Uppal, Steve Woo, Dan Pitt, Lee Doyle (Special Foreword): Software Defined WAN For Dummies; John Wiley & Sons, 2015, ISBN: 978-1-119-101482

2.1.e MEF Forum: Understanding SD-WAN Managed Services: Service Components, MEF LSO Reference Architecture and Use Cases, Jul 2017

2.1.f Fan Gu: SD-WAN Strategy to Address Key Trends and Scalability; IEEE Softwarization, Sep 2017

2.1.g Juniper Networks Contrail SD-WAN Design & Architecture Guide; 2018

2.1.h Rodolfo Alvizu, et al.: Comprehensive Survey on T-SDN: Software-Defined Networking for Transport Networks; IEEE Communications Surveys & Tutorials, Vol. 19(4), Jun 2017; DOI: 10.1109/COMST.2017.2715220

18.2.2 Software Defined Data Centers (SDDCs)

SD Data Centers

Die Virtualisierung von Hard- und Software-Ressourcen im Datacenter führt zur Entstehung virtueller Datacenter. Ein virtuelles Datacenter (*Virtual Data Center*,

VDC) kann aus einer vollständig virtualisierten, isolierten und verteilten Computing-Infrastruktur bestehen. Dies bedeutet, dass die wesentlichen funktionellen Komponenten dieser Infrastruktur virtualisiert und als *Data Center Services* (DC Services) eingerichtet werden können. Die hauptsächlichen Bausteine von DC Services sind: Netzwerk-Virtualisierung, Storage-Virtualisierung und Server-Virtualisierung. In einem physischen Datacenter können somit mehrere voneinander isolierte, dem individuellen Bedarf von Kunden angepasste virtuelle Datacenter eingerichtet werden. Das dem Konzept von SDN grundlegende Prinzip eignet sich ideal zur flexiblen Gestaltung solcher Services. Beim Einsatz der Idee von SDN zwecks der Bereitstellung virtueller Datacenter spricht man von *Software Defined Data Centers* (SDDCs). Ein VDC kann somit als SDDC bezeichnet werden. Für Näheres über SDDCs sei auf die folgende Literatur verwiesen:

2.2.a Distributed Management Task Force, Inc. (DMTF): Software Defined Data Center (SDDC) Definition; Document Identifier: DSP-IS0501; 2015-11-23; Version: 1.0.0

2.2.b Dell Inc.: The Future of the Data Center is Software Defined; Mar 2016

2.2.c Jeremy van Doorn: Software Defined Data Centers Network Virtualization & Security

2.2.d M. Gharbaoui, B. Martini, D. Adami, S. Giordano, P. Castoldi: Cloud and network orchestration in SDN data centers: Design principles and performance evaluation; Computer Networks, Vol. 108, Oct 2016; DOI: 10.1016/j.comnet.2016.08.029

2.2.e Guan Xu, Bin Dai, Benxiong Huang, Jun Yang, Sheng Wen: Bandwidth-aware energy efficient flow scheduling with SDN in data center networks; Future Generation Computer Systems, Vol. 68, Mar 2017; DOI: 10.1016/j.future.2016.08.024

2.2.f Hong Zhong, Yaming Fang, Jie Cui: LBBSRT: An efficient SDN load balancing scheme based on server response time; Future Generation Computer Systems, Vol. 68, Mar 2017; DOI: 10.1016/j.future.2016.10.001

2.2.g Bin Dai, Guan Xu, Bengxiong Huang, Peng Qin, Yang Xu: Enabling network innovation in data center networks with software defined networking: A survey, Journal of Network and Computer Applications, Vol. 94, Sep 2017; DOI: 10.1016/j.jnca.2017.07.004

2.2.h VMware: Architecture and Design: VMware Validated Design for Software-Defined Data Center 4.2; Feb 2018

18.2.3 Software Defined IoT (SD IoT)

SD IoT

Das Konzept von SDN [Abschnitt 17.1.7] hat eine ausgesprochen große Bedeutung im IoT, denn insbesondere in IoT-Clouds können spezielle, sog. SDN-Controller eingesetzt werden, um SDN-enabled Access Gateways an der Internet Edge entsprechend der aktuell geltenden Anforderungen (z.B. in Bezug auf Sicherheit) mittels einer Cloud über das Internet schnell zu konfigurieren [Abb. 17.1-9]. Darüber hinaus soll das IoT einer enorm großen Menge auf der Erdkugel verteilter Geräte (Sensoren,

Aktuatoren, ...) das Sammeln und somit die Verarbeitung von Echtzeitinformationen ermöglichen, was dazu führt, dass diese zur Bereitstellung neuer intelligenter IoT Services genutzt werden können. Beim Einsatz von SDN zur Bereitstellung intelligenter IoT Services spricht man von *Software Defined IoT* (SD IoT). Für weitere Informationen über SD IoT siehe:

2.3.a Ian Ku, et al.: Towards Software-Defined VANET: Architecture and Services; 13th Annual Mediterranean Ad Hoc Networking Workshop (MED-HOC-NET), Jun 2014; DOI: 10.1109/MedHocNet.2014.6849111

2.3.b Nachikethas A. Jagadeesan, Bhaskar Krishnamachari: Software-defined networking paradigms in wireless networks: A survey; ACM Computing Surveys, Vol. 47(2), Jan 2015

2.3.c Bin Cao, Fang He, Yun Li, Chonggang Wang, Wenqiang Lang: Software defined virtual wireless network: framework and challenges; IEEE Network, Vol. 29(4), Jul-Aug 2015; DOI: 10.1109/MNET.2015.7166185

2.3.d S. V. Manisekaran, R. Venkatesan: An analysis of software-defined routing approach for wireless sensor networks; Computers & Electrical Engineering, Vol. 56, Nov 2016

2.3.e Yanwen Wang, Hainan Chen, Xiaoling Wu, Lei Shu: An energy-efficient SDN based sleep scheduling algorithm for WSNs; Journal of Network and Computer Applications, Vol. 59, Jan 2016; DOI: 10.1016/j.jnca.2015.05.002

2.3.f Israat Tanzeena Haque, Nael Abu-Ghazaleh: Wireless Software Defined Networking: A Survey and Taxonomy; IEEE Communications Surveys & Tutorials, Vol. 18(4), May 2016; DOI: 10.1109/COMST.2016.2571118

2.3.g Hans C. Yu, Giorgio Quer, Ramesh R. Rao: Wireless SDN mobile ad hoc network: From theory to practice; 2017 IEEE International Conference on Communications (ICC), May 2017; DOI: 10.1109/ICC.2017.7996340

2.3.h Junfeng Wang, et al.: A software defined network routing in wireless multihop network; Journal of Network and Computer Applications, Vol. 85, May 2017; DOI: 10.1016/j.jnca.2016.12.007

2.3.i Manisha Chahal, et al.: A Survey on software-defined networking in vehicular ad hoc networks: Challenges, applications and use cases; Sustainable Cities and Society, Vol. 35, Nov 2017

2.3.j Afsane Zahmatkesh; Thomas Kunz: Software Defined Multihop Wireless Networks: Promises and Challenges; Journal of Communications and Networks, Vol. 19(6), Dec 2017; DOI: 10.1109/JCN.2017.000094

2.3.k Márcio L. F. Miguel, et al.: A CoAP Based Control Plane for Software Defined Wireless Sensor Networks; Journal of Communications and Networks, Vol. 19(6), Dec 2017; DOI: 10.1109/JCN.2017.000095

2.3.l Konstantinos Poularakis, George Iosifidis, Leandros Tassiulas: SDN-enabled Tactical Ad Hoc Networks: Extending Programmable Control to the Edge; arXiv:1801.02909v1, Jan. 2018

2.3.m Paolo Bellavista, Alessandro Dolci, Carlo Giannelli: MANET-oriented SDN: Motivations, Challenges, and a Solution Prototype; 2018 IEEE 19th Interna-

tional Symposium on 'A World of Wireless, Mobile and Multimedia Networks' (WoWMoM), Jun 2018; DOI: 10.1109/WoWMoM.2018.8449805

18.2.4 Wireless Software Defined Networking

Wireless SD Networking

Die grundlegende, auf der Trennung zwischen der Hardware-Plane und der Control-Plane in Netzwerken beruhende Idee von SDN [Abb. 12.1-3] eignet sich ideal für eine neue Gestaltung der Mobilität im IoT. In diesem Zusammenhang spricht man von *Wireless SD Networking*. Insbesondere die SDN-Idee bei der Integration verschiedener Ad-hoc Networks wie z.B. MANET (*Mobile Ad-hoc-NETwork*), VANET (*Vehicle Ad-hoc-NETwork*) und FANET (*Flying Ad-hoc Network*) in das IoT kann die Bereitstellung von intelligenten IoT Services ermöglichen. Für weitere Informationen über Wireless SD Networking siehe:

2.4.a Ian Ku, et al.: Towards Software-Defined VANET: Architecture and Services; 13th Annual Mediterranean Ad Hoc Networking Workshop (MED-HOC-NET), Jun 2014; DOI: 10.1109/MedHocNet.2014.6849111

2.4.b Nachikethas A. Jagadeesan, Bhaskar Krishnamachari: Software-defined networking paradigms in wireless networks: A survey; ACM Computing Surveys, Vol. 47(2), Jan 2015

2.4.c Bin Cao, Fang He, Yun Li, Chonggang Wang, Wenqiang Lang: Software defined virtual wireless network: framework and challenges; IEEE Network, Vol. 29(4), Jul-Aug 2015; DOI: 10.1109/MNET.2015.7166185

2.4.d S. V. Manisekaran, R. Venkatesan: An analysis of software-defined routing approach for wireless sensor networks; Computers & Electrical Engineering, Vol. 56, Nov 2016

2.4.e Yanwen Wang, Hainan Chen, Xiaoling Wu, Lei Shu: An energy-efficient SDN based sleep scheduling algorithm for WSNs; Journal of Network and Computer Applications, Vol. 59, Jan 2016; DOI: 10.1016/j.jnca.2015.05.002

2.4.f Israat Tanzeena Haque, Nael Abu-Ghazaleh: Wireless Software Defined Networking: A Survey and Taxonomy; IEEE Communications Surveys & Tutorials, Vol. 18(4), May 2016; DOI: 10.1109/COMST.2016.2571118

2.4.g Hans C. Yu, Giorgio Quer, Ramesh R. Rao: Wireless SDN mobile ad hoc network: From theory to practice; 2017 IEEE International Conference on Communications (ICC), May 2017; DOI: 10.1109/ICC.2017.7996340

2.4.h Junfeng Wang, et al.: A software defined network routing in wireless multihop network; Journal of Network and Computer Applications, Vol. 85, May 2017; DOI: 10.1016/j.jnca.2016.12.007

2.4.i Manisha Chahal, et al.: A Survey on software-defined networking in vehicular ad hoc networks: Challenges, applications and use cases; Sustainable Cities and Society, Vol. 35, Nov 2017

2.4.j Afsane Zahmatkesh; Thomas Kunz: Software Defined Multihop Wireless Networks: Promises and Challenges; Journal of Communications and Networks, Vol. 19(6), Dec 2017; DOI: 10.1109/JCN.2017.000094

2.4.k Márcio L. F. Miguel, et al.: A CoAP Based Control Plane for Software Defined Wireless Sensor Networks; Journal of Communications and Networks, Vol. 19(6), Dec 2017; DOI: 10.1109/JCN.2017.000095

2.4.l Konstantinos Poularakis, George Iosifidis, Leandros Tassiulas: SDN-enabled Tactical Ad Hoc Networks: Extending Programmable Control to the Edge; arXiv:1801.02909v1, Jan. 2018

2.4.m Paolo Bellavista, Alessandro Dolci, Carlo Giannelli: MANET-oriented SDN: Motivations, Challenges, and a Solution Prototype; 2018 IEEE 19th International Symposium on 'A World of Wireless, Mobile and Multimedia Networks' (WoWMoM), Jun 2018; DOI: 10.1109/WoWMoM.2018.8449805

18.3 Network Function Virtualization (NFV)

VM-based VNFs

Die Virtualisierung von Rechnern und deren Bereitstellung auf speziellen leistungsfähigen Wirt-Servern in Form von sog. *Virtual Machines* (VMs) sowie *Virtual Networking* als Vernetzung von sogar auf unterschiedlichen Wirt-Servern untergebrachten VMs sind in der Netzwerkwelt bereits geläufig. Und wenn man schon Rechner und ihre Vernetzung virtualisiert, so bietet es sich an, im folgenden Schritt verschiedene, auf der Basis von VMs realisierte Netzwerkkomponenten (Router, Switches, Firewalls usw.) und ihre Vernetzung, d.h. verschiedene Netzwerkfunktionen (Network Functions), als Softwarekomponenten bereitzustellen, will heißen zu virtualisieren. Diese Idee hat zu *Network Functions Virtualization* (NFV) geführt, bei der man von *virtualisierten Netzwerkfunktionen* spricht. Sie werden kurz VNFs (*Virtualised Network Functions*) genannt. Im Zusammenhang mit NFV sind die Networking Trends *Software Defined VNFs Networking* (*SD VNFs Networking*), *Service Function Chaining* (SFC), *VNFs Management and Orchestration* und *Network Slicing* hervorzuheben. Für Näheres über NFV siehe:

3.a ETSI GS NFV 001: Network Functions Virtualisation (NFV); Use Cases: Oct 2013

3.b ETSI GS NFV-SEC 001 - V1.1.1: Network Functions Virtualisation (NFV); NFV Security; Problem Statement; Oct 2014

3.c ETSI GS NFV 002 V1.2.1: Network Functions Virtualisation (NFV) - Architectural Framework; Dec 2014

3.d ETSI GS NFV-MAN 001 V1.1.1: Network Functions Virtualisation (NFV) - Management and Orchestration; Dec 2014

3.e ETSI GS NFV-INF 001 - V1.1.1: Network Functions Virtualisation (NFV); Infrastructure Overview; Jan 2015

3.f Anatol Badach: NFV – Network Functions Virtualisation; In book: Protokolle und Dienste der Informationstechnologie, WEKA-Verlag, Editor: Heinz Schulte; Jan 2015, DOI 10.13140/RG.2.1.4701.2562

3.g Rashid Mijumbi, et al.: Network Function Virtualization: State-of-the-Art and Research Challenges; IEEE Communications Surveys & Tutorials, Vol. 18(1), Sep 2016; DOI: 10.1109/COMST.2015.2477041

3.h Eugen Borcoci: Network Function Virtualization and Software Defined Networking Cooperation; InfoSys 2015 Conference, May 2015

3.i Rashid Mijumbi, et al.: Network Function Virtualization: State-of-the-Art and Research Challenges; IEEE Communications Surveys & Tutorials, Vol. 18(1), Sep 2016; DOI: 10.1109/COMST.2015.2477041

3.j ETSI GR NFV 001 - V1.2.1: Network Functions Virtualisation (NFV); Use Cases; May 2017

3.k Shariq Haseeb, et al.: Network Function Virtualization (NFV) based architecture to address connectivity, interoperability and manageability challenges in Internet of Things (IoT); IOP Conf. Series: Materials Science and Engineering 260 (2017); DOI: 10.1088/1757-899X/260/1/012033

3.l Michel S. Bonfim, Kelvin L. Dias, Stenio F. L. Fernandes: Integrated NFV/SDN Architectures: A Systematic Literature Review; Jan 2018; arXiv:1801.01516v1

18.3.1 Software Defined VNFs Networking

SD VNFs Networking

Von großer Bedeutung ist die Tatsache, dass die Konzepte SDN und NFV sich sowohl ähneln als auch ideal ergänzen. Vor allem der Orchestrator bei SDN entspricht der Funktion nach vollkommen dem Orchestrator bei NFV. Folglich können einige Ideen und somit auch Funktionskomponenten von SDN für NFV übernommen werden, was zu SD VNFs Networking führt. Für Näheres darüber siehe:

3.1.a Anatol Badach: NVGRE – Network Virtualization using Generic Routing Encapsulation; In book: Protokolle und Dienste der Informationstechnologie, WEKA-Verlag, Editor: Heinz Schulte; Oct 2014, DOI 10.13140/RG.2.1.2872.8802

3.1.b ETSI GS NFV-SWA 001 - V1.1.1: Network Functions Virtualisation (NFV); Virtual Network Functions Architecture; Dec 2014

3.1.c ETSI GS NFV 002 - V1.2.1: Network Functions Virtualisation (NFV); Architectural Framework; Dec 2014

3.1.d ETSI GS NFV-EVE 005 - V1.1.1: Network Functions Virtualisation (NFV); Ecosystem; Report on SDN Usage in NFV Architectural Framework; Dec 2015

3.1.e Lorena Isabel Barona López, et al.: Trends on virtualisation with software defined networking and network function virtualisation; IET Networks, Vol. 4(5), Sep 2015; DOI: 10.1049/iet-net.2014.0117

3.1.f Jie Li, Eitan Altman, Corinne Touati: A General SDN-based IoT Frame-work with NVF Implementation; ZTE Communications, ZTE Corporation, 2015, Vol. 13 (3)

3.1.g Yong Li, Min Chen: Software-Defined Network Function Virtualization: A Survey; IEEE Access, Vol. 3, Dec 2015c DOI: 10.1109/ACCESS.2015.2499271

3.1.h Verizon: Verizon Network Infrastructure Planning: SDN-NFV Reference Architecture; Version 1.0, Feb 2016

3.1.i Qiang Duan, Nirwan Ansari, Mehmet Toy: Software-Defined Network Virtualization: An Architectural Framework for Integrating SDN and NFV for Service Provisioning in Future Networks; IEEE Network, Vol. 30(5), Sep 2016; DOI: 10.1109/MNET.2016.7579021

3.1.j Shariq Haseeb, et al.: Network Function Virtualization (NFV) based architecture to address connectivity, interoperability and manageability challenges in Internet of Things (IoT); IOP Conference Series: Materials Science and Engineering, Vol. 260, 2017

18.3.2 Service Function Chaining (SFC)

Mit der Verfügbarkeit von Rechnern in Form von VMs eröffnen sich vollkommen neue Möglichkeiten, unterschiedliche an Geschäftsprozesse angepasste Netzwerkdienste spontan bereitzustellen. Denn auf der Basis von VMs können zur Einrichtung verschiedener komplexer Netzwerkdienste diverse Funktionskomponenten als virtuelle, standardisierte Netzwerkfunktionsmodule im Voraus für einen Einsatz vorbereitet und dann bei Bedarf als Softwarekomponenten ad hoc verfügbar gemacht werden. Sie werden kurz als VNFs (*Virtualised Network Functions*) bzw. auch als SFs (*Service Functions*) bezeichnet. Da die Realisierung jedes Netzwerkdienstes auf der Basis von durch virtualisierte Rechner erbrachte SFs oft zu einer Verkettung von SFs führt, spricht man in diesem Zusammenhang von *Service Function Chaining* (SFC). Unter SFC werden somit zahlreiche Konzepte und Lösungen zur Kooperation von hauptsächlich softwaremäßig realisierten SFs zwecks der Erbringung diverser Netzwerkdienste verstanden. Für weitere Information über SFC siehe:

3.2.a ETSI GS NFV 002, V1.2.1: Network Functions Virtualisation (NFV); Architectural Framework; Dec 2014

3.2.b Anatol Badach: SFC Service Function Chaining; In book: Protokolle und Dienste der Informationstechnologie, WEKA-Verlag, Editor: Heinz Schulte; Jan 2015, DOI 10.13140/RG.2.1.4701.2562

3.2.c Barbara Martini, Federica Paganelli: A Service-Oriented Approach for Dynamic Chaining of Virtual Network Functions over Multi-Provider Software-Defined Networks; Future Internet, Vol. 8(2), Jun 2016; DOI:10.3390/fi802002

3.2.d Deval Bhamare, Raj Jain, Mohammed Samaka, Aiman Erbad: A survey on service function chaining; Journal of Network and Computer Applications, Vol. 75, Nov 2016; DOI: 10.1016/j.jnca.2016.09.001

3.2.e Janos Elek, David Jocha, Robert Szabo: Network Function Chaining in DCs: the Unified Recurring Control Approach; Fourth European Workshop on Software Defined Networks, Oct 2015; DOI: 10.1109/EWSDN.2015.54

3.2.f Deval Bhamare, et al.: Optimal virtual network function placement in multi-cloud service function chaining architecture; Computer Communications, Vol. 102, Apr 2017; DOI: 10.1016/j.comcom.2017.02.011

3.2.g Ahmed AbdelSalam, et al.: Implementation of Virtual Network Function Chaining through Segment Routing in a Linux-based NFV Infrastructure, Apr 2017, arXiv:1702.05157v4

3.2.g Taixin Li, Huachun Zhou, Hongbin Luo: A new method for providing network services: Service function chain; Nov 17; DOI: 10.1016/j.osn.2015.09.005

3.2.i H. Kitada, H. Kojima, N. Takaya, and M. Aihara: Service Function Chain-ing Technology for Future Networks; NTT Technical Review

18.3.3 VNFs Management and Orchestration

Zur Schaffung eines Network Services auf der Grundlage von VNFs werden in der Regel mehrere VNFs benötigt. Diese VNFs werden durch verschiedene VMs erbracht und können räumlich weit auseinanderliegen. Ein solcher Network Service stellt de facto eine aus mehreren räumlich verteilten VNFs bestehende 'Service-Komposition' dar. Solch eine Komposition ist mit einer von mehreren Musikern gespielten Musikkomposition vergleichbar. Eine VNF entspricht in diesem Fall einem Musiker. Wird die Komposition von mehreren Musikern gespielt, müssen sie dirigiert werden, es ist also ein Dirigent nötig. So ist es auch im Falle der VNFs, wenn mehrere zusammen agieren müssen, um einen Network Service zu erbringen. Als ihr 'Dirigent' fungiert dann ein SDN Controller. So entspricht die Bereitstellung von Network Services, die durch mehrere VNFs erbracht werden, weitgehend der Vorbereitung eines von mehreren Musikern gespielten Konzerts. Es ist eine Orchestrierung von VNFs (*VNFs Orchestration*) und von Network Services sowie ein Management ihrer notwendig. Für weitere Details zum VNFs Management und Orchestration siehe:

3.3.a ETSI GS NFV-MAN 001, V1.1.1: Network Functions Virtualisation (NFV); Management and Orchestration; Dec 2014

3.3.b Daniel King, et al.: Network service orchestration standardization: A technology survey; Computer Standards & Interfaces, Vol. 54(4), Nov. 2017; DOI: 10.1016/j.csi.2016.12.006

3.3.c Charalampos Rotsos, et al.: Network service orchestration standardization: A technology survey; Computer Standards & Interfaces, Vol. 54(4), Nov 2017; DOI: 10.1016/j.csi.2016.12.006

3.3.d 3GPP TS 28.531: Management and orchestration of networks and network slicing; Provisioning

3.3.e 3GPP TS 28.533: Management and orchestration; Architecture framework

3.3.f ETSI GS NFV-IFA 014 V2.4.1: Network Functions Virtualisation (NFV) Release 2; Management and Orchestration; Network Service Templates Specification, Feb 2018

18.3.4 Network Slicing

Man könnte sich einzelne virtuelle private Netzwerkinfrastrukturen im Datacenter als 'virtuelle Netzwerkscheiben' vorstellen, welche voneinander isoliert werden. Da die

virtuellen Netzwerkscheiben als *Network Slices* bezeichnet werden, spricht man im Zusammenhang mit der Bildung dieser von *Network Slicing*. *Network Slices* können dank der Orchestrierung von VNFs gebildet werden. Ihr Management erfolgt nach dem Prinzip von SDN. Auf der Basis von Network Slicing werden beispielsweise private mobile 5G-Netzwerke gebildet. Es ist zu erwarten, dass Network Slicing in der Netzwerkwelt zukünftig von fundamentaler Bedeutung sein wird. Für weitere Details über Network Slicing siehe:

3.4.a NGMN Alliance: Description of Network Slicing Concept, Version 1.0, Jan 2016

3.4.b ETSI GR NFV-EVE 012, V3.1.1: Network Functions Virtualisation (NFV) Release 3; Evolution and Ecosystem; Report on Network Slicing Support with ETSI NFV Architecture Framework; Dec 2017

3.4.c 3GPP TR 28.801, V15.1.0: Study on management and orchestration of network slicing for next generation network (Release 15); Jan 2018

3.4.d Alex Galis: Perspectives on Network Slicing – Towards the New 'Bread and Butter' of Networking and Servicing; IEEE Softwarization, Jan 2018

3.4.e Luis M. Contreras, Diego R. López: A Network Service Provider Perspective on Network Slicing; IEEE Softwarization, Jan 2018

3.4.f Daniele Ceccarelli, Young Lee, Huawei: Transport Aspects of Network Slicing: Existing Solutions and Gaps; IEEE Softwarization, Jan 2018

3.4.g Huanzhuo Wu, et al.: Network Slicing for Conditional Monitoring in the Industrial Internet of Things; IEEE Softwarization, Jan 2018

3.4.h Sławomir Kukliński, et al.: A reference architecture for network slicing; NETSOFT 2018, 4th IEEE Conference on Network Softwarisation, Jun 2018

18.4 (Docker) Container Networking

Container Technologie als Virtualisierung auf Betriebssystemebene

Die Bereitstellung von *Virtual Machines* (VMs) und auf der Grundlage von VNFs (*Virtualised Network Functions*) kann als eine Art Virtualisierung auf Hardwareebene angesehen werden. Eine VNF kann aber auch auf der Basis eines oder mehrerer Rechner/s eingerichtet werden, in dem/denen – sozusagen als 'Virtualisierung auf Betriebssystemebene' – die sog. *Container Technologie* (CT) realisiert wird. Die Idee dieser Virtualisierungsart basiert darauf, dass in einem physischen Rechner mit seinem Betriebssystem, z.B. Linux, eine besondere Applikation installiert wird, die als virtualisiertes Betriebssystem mit Applikationen angesehen werden kann. Dabei können die Applikationen ihnen Funktionen entsprechend gruppiert und in sog. (virtuellen) Containern 'untergebracht' werden. Man spricht in diesem Zusammenhang von *Container-basierter Virtualisierung*.

Mobilität von Containern mit Applikationen

Man könnte sich so ein virtualisiertes Betriebssystem mit seinen Applikationen als ein als *Docker* bezeichnetes Schiff mit seinen Containern vorstellen. Ein wichtiger Vorteil nicht-virtueller Container besteht darin, dass sie relativ einfach von einem Ort zum anderen, z.B. Schiff zu Schiff, bewegt werden können. Auch die Mobilität

virtueller Container mit Applikationen kann einfach erreicht werden, und zwar dadurch, dass man diesen – auf einem Docker mit einer offiziellen IP-Adresse – private IP-Adressen zuweisen kann. Auf jedem Docker kann das NAT-Verfahren (*Network Address Translation*) realisiert werden [Abschnitt 6.3]. Durch dieses kann jeder virtuelle Container mit seiner privaten IP-Adresse auf jedem Docker neu installiert werden und dabei – auch häufig – seine private IP-Adresse erhalten bleiben. In Folge kann jeder Container mit seiner privaten IP-Adresse von einem physischen Rechner auf einen anderen bewegt werden. Dies bedeutet, dass eine uneingeschränkte Mobilität von Containern möglich ist, in denen verschiedene VNFs durch die in ihnen enthaltenen Applikationen realisiert werden können.

Die Mobilität von Containern trägt dazu bei, dass VNFs bei der Realisierung der Container Technologie in einer Cloud mithilfe der Datenübermittlung von einer Cloud in andere transportiert werden können. Container Networking liegt insbesondere den Networking Trends *Container-based Network Services* , *Cloud Computing Containerization*, *Mobile VNFs Networking und Containerized IoT Services* zugrunde. Für weitere Informationen zum Thema Container Networking siehe:

4.a Martin Fowler: Microservices Resource Guide

4.b Mark Church: Docker Reference Architecture: Designing Scalable, Portable Docker Container Networks

4.c David Bernstein: Containers and Cloud: From LXC to Docker to Kubernetes; IEEE Cloud Computing, Vol. 1(3), Sep 2014; DOI: 10.1109/MCC.2014.51

4.d Di Liu, Libin Zhao: The research and implementation of cloud computing platform based on docker; 11th International Computer Conference on Wavelet Active Media Technology and Information Processing(ICCWAMTIP), Dec 2014; DOI: 10.1109/ICCWAMTIP.2014.7073453

4.e Richard Cziva, Simon Jouet, Kyle J. S. White, Dimitrios P. Pezaros: Container-based Network Function Virtualization for Software-Defined Networks; 2015 IEEE Symposium on Computers and Communication (ISCC), Jul 2015 DOI: 10.1109/ISCC.2015.7405550

4.f Jason Anderson, et al.: Performance considerations of network functions virtualization using containers; International Conference on Computing, Networking and Communications (ICNC), Feb 2016; DOI: 10.1109/ICCNC.2016.7440668

4.g Mark Church, Lorenzo Fontana, Nico Kabar: Networking Workshop; Dockercon 2017

4.h Docker: Introduction to Container Security: Understanding the isolation properties of Docker: White Paper, Aug 2016

4.i Gaetano Borgione: Container Networking: Deep Dive; 2017

4.j Roberto Morabito: Virtualization on Internet of Things Edge Devices With Container Technologies: A Performance Evaluation; IEEE Access, Vol. 5, May 2017; DOI: 10.1109/ACCESS.2017.2704444

4.k Richard Cziva, Dimitrios P. Pezaros: Container Network Functions: Bringing NFV to the Network Edge; IEEE Communications Magazine Vol. 55(6), Jun 2017; DOI: 10.1109/MCOM.2017.1601039

4.l Phil Lowden: Cisco Container Networking Overview and Roadmap; Nov 2017

4.m VMware: CONTAINERS AND CONTAINER NETWORKING: For Network Engineers; Jan 2018

4.n Michael Hausenblas: Container Networking - From Docker to Kubernetes; O'Reilly Media, May 2018

18.4.1 Container-based Network Services

In einem Container können mehrere Applikationen enthalten sein, von denen jede eine eigenständige Ausführungsumgebung hat und dadurch isoliert von anderen Applikationen ausgeführt werden kann. Die voneinander isolierten Applikationen können dazu dienen, verschiedene Netzwerkfunktionen, sog. VNFs, zu kreieren. Daraufhin können mithilfe von VNFs bestimmte Netzwerkdienste erbracht werden. Sie werden als *Container-based Network Services* oder *Microservices* bezeichnet. Für weitere Informationen darüber siehe:

4.1.a Martin Šuňal: Container service chaining

4.1.b Roberto Morabito, Nicklas Beijar: Enabling Data Processing at the Network Edge through Lightweight Virtualization Technologies; IEEE International Conference on Sensing, Communication and Networking (SECON Workshops); Jun 2016; DOI: 10.1109/SECONW.2016.7746807

4.1.c Sergio Livi, et al.: Container-Based Service Chaining: A Performance Perspective; 5th IEEE International Conference on Cloud Networking (Cloudnet), Oct 2016; DOI: 10.1109/CloudNet.2016.51

4.1.d Kuljeet Kaur, Tanya Dhand, Neeraj Kumar, Sherali Zeadally: Container-as-a-Service at the Edge: Tradeoff between Energy Efficiency and Service Availability at Fog Nano Data Centers; IEEE Wireless Communications, Vol. 24(3) , Jun 2017; DOI: 10.1109/MWC.2017.1600427

4.1.e Kyungwoon Lee, Youngpil Kim, Chuck Yoo: The Impact of Container Virtualization on Network Performance of IoT Devices; Mobile Information Systems, Vol. 2018, Article ID 9570506, May 2018; DOI: 10.1155/2018/9570506

4.1.f Cisco: Cloud-Native Network Functions (CNFs), White Paper; Jun 2018; Document ID:1529344804993194

4.1.g Qi Zhang, Ling Liu, Calton Pu, Qiwei Dou, Liren Wu, Wei Zhou: Comparative Study of Containers and Virtual Machines in Big Data Environment; arXiv:1807.01842v1, Jul 2018

18.4.2 Cloud Computing Containerization

Sowohl in einem physischen als auch in einem virtualisierten Rechner, d.h. auf einer Virtual Machine (VM), können mehrere Container mit diversen Microservices eingerichtet werden. Auf der Basis der beiden Rechnerarten, in denen verschiedene *Container-based Network Services* in Form von VNFs verfügbar sind, können sog. *Clouds* gebildet werden. Infolgedessen kann die Container Technologie auch in

Clouds realisiert werden. Man spricht in diesem Zusammenhang von *Cloud Computing Containerization*. Für Näheres darüber siehe:

4.2.a Claus Pahl, Brian Lee: Containers and Clusters for Edge Cloud Architectures - A Technology Review; The 3rd International Conference on Future Internet of Things and Cloud (FiCloud 2015); Aug 2015; DOI: 10.1109/FiCloud.2015.35

4.2.b Hui Kang, Michael Le, Shu Tao: Container and Microservice Driven Design for Cloud Infrastructure DevOps, IEEE International Conference on Cloud Engineering (IC2E), Apr 2016; DOI: 10.1109/IC2E.2016.26

4.2.c Nane Kratzke: A Brief History of Cloud Application Architectures; Applied Sciences, Vol. 8(8), Aug 2018, DOI: 10.3390/app8081368

4.2.d ETSI GS NFV-EVE 011 V3.1.1 (2018-10): Network Functions Virtualisation (NFV) Release 3; Virtualised Network Function; Specification of the Classification of Cloud Native VNF Implementations Disclaimer

18.4.3 Mobile VNFs Networking

Von einem Container in einem physischen bzw. in einem virtuellen Rechner können mehrere VNFs erbracht werden. Jedem Container kann eine private IP-Adresse zugewiesen werden. Dadurch entsteht die Möglichkeit, dass jeder Container mit einer privaten IP-Adresse von einem Docker, quasi einem 'virtuellen Schiff', zu einem anderen transportiert werden kann. Diese Mobilität von Containern mit VNFs schließt die *Mobilität von VNFs* mit ein. Verschiedene mobile VNFs können dank der Realisierung des Mobile VNFs Networking untereinander auf eine Weise vernetzt werden, dass die Einrichtung eines Mobile Network Service möglich ist. Für weitere Informationen darüber siehe:

4.3.a Rajendra Chayapathi, Syed Farrukh Hassan, Paresh Shah: Network Functions Virtualization (NFV) with a Touch of SDN; Addison-Wesley, Nov 2016; ISBN-13: 978-0-13-446305-6

4.3.b Ajay Simha: Red Hat Reference Architecture Series: NFV reference architecture for deployment of mobile networks; Version 1.3, Jan 2017

4.3.c Project: H2020-ICT-2014-2 5G NORMA Definition and specification of connectivity and QoE/QoS management mechanisms; Final Report, Jun 2017

18.4.4 Containerized IoT Services

Im IoT hat sowohl Cloud Computing als auch Fog Computing eine fundamentale Bedeutung. Wie bereits erwähnt, führt ein Networking Trend zur *Cloud Computing Containerization*. Da Fog Computing [Abschnitt 17.2-5] als eine besondere Art von Cloud Computing betrachtet werden kann, bei der ein Mini Cloud einen Fog Node darstellt, führt der Einsatz der Container Technologie zur Containerization beider Arten von Computing. Dank dieser Containerization kommt die Container Technologie auch im IoT zum Einsatz, sodass man von *Containerized IoT Services* sprechen kann. Für weitere Informationen darüber sei verwiesen auf:

4.4.a Azhar Sayeed, Dejan Leskaroski: Cloud Native Applications in a Telco World - How Micro Do You Go?

4.4.b Bukhary Ikhwan Ismail, et al.: Evaluation of Docker as Edge computing platform; Proc. IEEE Conference on Open Systems (ICOS), Aug 2015; DOI: 10.1109/ICOS.2015.7377291

4.4.c Riccardo Petrolo, Roberto Morabito, Valeria Loscrì, Nathalie Mitton: The design of the gateway for the cloud of things; Annals of Telecommunications, Vol. 72, Issue 1–2, Feb 2017; DOI 10.1007/s12243-016-0521-z

4.4.d Roberto Morabito, Ivan Farris, Antonio Iera, Tarik Taleb: Evaluating Performance of Containerized IoT Services for Clustered Devices at the Network Edge; IEEE Internet of Things Journal, Vol. 4(4), Aug. 2017; DOI: 10.1109/JIOT.2017.2714638

4.4.e Koustabh Dolui, Csaba Kiraly: Towards Multi-container Deployment on IoT Gateways; Oct 2018; arXiv:1810.07753v1

18.5 Cloud Computing Services

Bedeutung von Cloud Computing

Mit der Entwicklung von IoT gewinnt das als Cloud Computing bezeichnete Service-Modell enorm an Bedeutung [vgl. Abb. 17.1-4]. Die große Bedeutung von Cloud Computing besteht hauptsächlich darin, dass man dem Cloud Computing zugrunde liegenden Modell nach bei Bedarf spontan über Netzwerke auf einen Pool konfigurierbarer Computing-Ressourcen (z.B. Netzwerke, Server, Speicher, Applikationen und Dienste) zugreifen kann. Infolge der immensen Menge verschiedener weltweit im IoT verteilter Devices und Objekte werden riesige global verstreute Datenmengen erzeugt. Diese müssen zuerst entsprechend gespeichert und dann schnell verarbeitet werden. Dabei kommt das Konzept von Cloud Computing zum Einsatz, wobei die IoT-spezifischen Clouds als eine Art 'IoT Service Platforms' fungieren. Mit Cloud Computing sind insbesondere die Networking Trends *Infrastructure-as-a-Service* (IaaS), *Software-Defined CC Networking*, *Cloud Native Microservices* und *Mobile CC in 5G* verbunden. Für weitere Informationen über Cloud Computing sei verwiesen auf die folgende Literatur:

5.a i-SCOOP: Cloud computing – from private, public and hybrid cloud to cloud services and cloud evolutions

5.b B. Wanga, Z. Qi, R. Ma, H. Guan, A. V. Vasilakos: A survey on data center networking for cloud computing; Computer Networks, Vol. 91, 2015; DOI: 10.1016/j.comnet.2015.08.040

5.c Jatinder Singh, Thomas Pasquier, Jean Bacon, Hajoon Ko, David Eyers: Twenty Security Considerations for Cloud-Supported Internet of Things; IEEE Internet of Things Journal, Vol. 3(3), Jun 2016; DOI: 10.1109/JIOT.2015.2460333

5.d Blesson Varghese, Rajkumar Buyya: Next Generation Cloud Computing: New Trends and Research Directions, Sep 2017, arXiv:1707.07452v3

5.e Syed Noorulhassan Shirazi, Antonios Gouglidis, Arsham Farshad, David Hutchison: The Extended Cloud: Review and Analysis of Mobile Edge Computing and

Fog from a Security and Resilience Perspective; IEEE Jour-nal on Selected Areas in Communications, Vol. 35(11) , Nov 2017, DOI: 10.1109/JSAC.2017.2760478

5.f Yahya Al-Dhuraibi, Fawaz Paraiso, Nabil Djarallah, Philippe Merle: Elasticity in Cloud Computing: State of the Art and Research Challenges; IEEE Transactions on Services Computing, Vol. 11(2), Mar - Apr 2018; DOI: 10.1109/TSC.2017.2711009

5.g Amirhossein Farahzadi, et al.: Middleware technologies for cloud of things: a survey; Digital Communications and Networks, Vol. 4(3), Aug 2018; DOI: 10.1016/j.dcan.2017.04.005

18.5.1 Infrastructure-as-a-Service (IaaS)

IaaS

Enthält ein Cloud einen derartigen Pool konfigurierbarer Computing-Ressourcen, um die bedarfsabhängige Bereitstellung durch einen Cloud-Provider einer virtuellen, aus verschiedenen VNFs bestehenden netzwerkspezifischen Infrastruktur zu ermöglichen, wird von einem *Infrastructure-as-a-Service* (IaaS) gesprochen. Als IaaS können verschiedene private und voneinander isolierte, virtuelle Infrastrukturen angeboten werden. Dies wird auch *Network Slicing* genannt. Bei der Realisierung von IaaS spielt *Service Function Chaining* (SFC) eine wichtige Rolle. Für Näheres über IaaS siehe:

5.1.a Antonio Celesti, Davide Mulfari, Maria Fazio, Massimo Villari, Antonio Puliafito: Exploring Container Virtualization in IoT Clouds; IEEE International Conference on Smart Computing (SMARTCOMP), May 2016; DOI: 10.1109/SMARTCOMP.2016.7501691

5.1.b Heli Amarasinghe, Abdallah Jarray, Ahmed Karmouch: Fault-tolerant IaaS management for networked cloud infrastructure with SDN; IEEE Interna-tional Conference on Communications (ICC'17), May 2017; DOI: 10.1109/ICC.2017.7996342

5.1.c Steven Van Rossem, et al.: A Vision for the Next Generation Platform-as-a-Service; Proceedings of 2018 IEEE 1st 5G World Forum

18.5.2 Software-Defined Cloud Computing Networking

Software-Defined CC Networking

Ein IaaS kann dem Bedarf entsprechend auf eine festgelegte Zeitdauer und sogar auf der Basis mehrerer vernetzter Clouds eingerichtet werden. Somit gilt im Allgemeinen die Vernetzung von Clouds, also *Cloud Computing Networking* (CC Networking), als Voraussetzung für die Bereitstellung von IaaS. Hierbei ist auch eine flexible Lösung nötig, um die notwendigen VNFs zu bestimmen, diese dem aktuellen Bedarf entsprechend untereinander virtuell zu vernetzen, zu konfigurieren und im Laufe der Zeit zu managen. Um diese Aufgaben bei IaaSs zu bewältigen, werden einige Konzepte von SDN übernommen, und demzufolge spricht man von *Software-Defined CC Networking*. Für weitere Information darüber siehe:

5.2.a Rajkumar Buyya, Rodrigo N. Calheiros, Jungmin Son, Amir Vahid Dastjerdi, Young Yoony: Software-Defined Cloud Computing: Architectural Elements and Open Challenges, Feb 2015; arXiv:1408.6891v2

5.2.b Foresta Francesco: Integration of SDN Frameworks And Cloud Computing Platforms: An Open Source Approach; Master Thesis, Universita di Bologna; Academic Year 2016/2017

5.2.c Jungmin Son, Rajkumar Buyya: A Taxonomy of Software-Defined Networking (SDN)-Enabled Cloud Computing; ACM Computing Surveys, Vol.51(3), May 2018; DOI: 10.1145/3190617

18.5.3 Cloud Native Microservices

Cloud-Native VNFs bzw. Microservices

Ein wichtiger Trend ist die Integration von Cloud Computing in öffentliche Netzwerkinfrastrukturen mit dem Ziel der Bereitstellung und flexiblen Nutzung virtueller Netzwerkfunktionen, d.h. VNFs, in diesen. Da VNFs als netzwerkspezifische Funktionsbausteine, also eine Art 'Network-App' dienen, werden sie auch als *Microservices* bezeichnet. Solche virtuellen Microservices (z.B. vRouter, vFirewalls, ...) ermöglichen die schnelle Einrichtung diverser komplexer Network Services. Wurden VNFs bzw. Microservices speziell für Cloud-Computing-Architekturen entwickelt und optimiert, so werden sie entsprechend *Cloud-Native VNFs* und *Cloud-Native Microservices* genannt. Für Weiteres darüber siehe:

5.3.a Azhar Sayeed, Dejan Leskaroski: Cloud Native Applications in a Telco World - How Micro Do You Go?

5.3.b Antonio Celesti, et al.: Exploring Container Virtualization in IoT Clouds; IEEE International Conference on Smart Computing (SMARTCOMP), May 2016; DOI: 10.1109/SMARTCOMP.2016.7501691

5.3.c Hamzeh Khazaei, Hadi Bannazadeh, Alberto Leon-Garcia: SAVI-IoT: A Self-Managing Containerized IoT Platform, IEEE 5th International Conference on Future Internet of Things and Cloud (FiCloud), Aug 2017; DOI: 10.1109/FiCloud.2017.27

5.3.d Cisco: Cloud-Native Network Functions (CNFs); White Paper, Jun 2018; Document ID:1529344804993194

5.3.e 5G-PPP Software Network Working Group: From Webscale to Telco, the Cloud Native Journey; July 2018

5.3.f Tetiana Yarygina: Exploring Microservice Security; PhD Thesis, Department of Informatics, University of Bergen, Jul 2018

5.3.g ETSI GS NFV-EVE 011 V3.1.1 (2018-10): Network Functions Virtualisation (NFV) Release 3; Virtualised Network Function; Specification of the Classification of Cloud Native VNF Implementations Disclaimer

18.5.4 Mobile Cloud Computing in 5G

Mobile Cloud Computing

Heutzutage haben mobile Kommunikation und mobiles Computing in allen Bereichen unseres Lebens eine große Bedeutung. Die Möglichkeiten der Nutzung des Mobile Computing kann noch durch dessen Symbiose mit Cloud Computing verbessert werden. Um dies zu erreichen, wird ein neuer Trend namens *Mobile Cloud Computing* verfolgt. Diesem wird aktuell, insbesondere im Hinblick auf die neu entwickelten zellularen 5G-Mobilfunknetze, im Forschungsbereich viel Aufmerksamkeit gewidmet. Für weitere Information über MCC siehe:

5.4.a Min Chen, Yin Zhang, Yong Li, Shiwen Mao, Victor C. M. Leung: EMC: Emotion-Aware Mobile Cloud Computing in 5G; IEEE Network, Vol. 29(2), Mar - Apr 2015; DOI: 10.1109/MNET.2015.7064900

5.4.b Nasir Abbas, Yan Zhang, Amir Taherkordi, Tor Skeie: Mobile Edge Computing: A Survey; IEEE Internet of Things Journal, 2018; DOI: 10.1109/JIOT.2017.2750180

5.4.c Talal H. Noor, Sherali Zeadally, Abdullah Alfazi, Quan Z. Sheng: Mobile cloud computing: Challenges and future research directions; Journal of Network and Computer Applications, Vol. 115, Aug 2018; DOI: 10.1016/j.jnca.2018.04.018

18.6 Fog Computing & Artificial Intelligence (AI)

Bedeutung von Fog Computing im IoT

Fog Computing (FC) kann als Ergänzung von Cloud Computing im IoT angesehen werden [Abschnitt 17.1.4]. Die Funktionen von FC werden im IoT durch eine horizontal verteilte große Menge von Fog Nodes, einer Art Mini-Clouds, entlang des Internet-Kontinuums zwischen Clouds und Things in die Nähe seiner Benutzer und somit der IoT-Devices gebracht. FC führt also zu einer Verteilung von Computing-Ressourcen. Dadurch können diese überall möglichst nah an Benutzern und IoT-Devices platziert werden. Die am Internet Edge entlang platzierten Computing-Ressourcen tragen dazu bei, dass die 'Intelligenz' verteilter Systeme aus IoT-Clouds zu den diese benötigenden 'IoT-Objekten' hin bewegt werden kann.

Fog Computing: 'Lieferant' der Intelligenz für IoT-Objekte

Fog Computing stellt eine Computing-Architektur dar, die am Internet Edge installierten IoT-Objekten die von ihnen benötigte, zusätzliche Intelligenz 'liefert'. Dadurch können diese Objekte lokal und alternativ zu Cloud Computing einige intelligente Funktionen durchführen. Aus diesem Grund gewinnen die von Fog Nodes am Internet Edge erbrachte künstliche Intelligenz (*Artificial Intelligence*, AI) und maschinelles Lernen (*Machine Learning*, ML) ständig an Bedeutung. Mittels Fog Nodes können auch diverse Microservices, u.a. welche mit VNFs, am Internet Edge verfügbar gemacht werden.

Edge Computing, Mobile Edge Computing

Die Idee, Intelligenz und Verarbeitungsfähigkeiten möglichst in direkter Nähe des Internet Edge zu positionieren, wird auch bei Edge Computing verfolgt. Fog Computing und Edge Computing unterscheiden sich hauptsächlich darin, wo genau die Intelligenz und Rechenleistung erbracht werden. Mit Fog Computing kann diese Funktionalität

überall im Internet-Kontinuum zwischen IoT Clouds und Internet Edge erbracht werden. Mit Edge Computing wird sie direkt am Internet Edge erbracht – und zwar oft in IoT Access Gateways. Um mobilen Endeinrichtungen in 5G-Mobilfunknetzen Intelligenz und Unterstützung bei der Informationsverarbeitung bereitzustellen, werden spezielle Clouds am Internet Edge in Basisstationen dieser Netze installiert. Man spricht in diesem Zusammenhang von *Mobile Edge Computing*.

Auf Fog Computing bzw. (Mobile) Edge Computing und der Nutzung von AI basieren die Networking Trends: *Time-sensitive IoT / 5G Applications*, *Intelligent IoT*, *Cognitive IoT*, *Ambient Intelligence in IoT* und *IoT Service Orchestration*. Für weitere Information über Fog Computing und die Bedeutung von Artificial Intelligence (AI) bei Fog Computing siehe:

6.a i-SCOOP: Edge computing and IoT 2018 – when intelligence moves to the edge

6.b i-SCOOP: Fog computing: fog and cloud along the Cloud-to-Thing continuum

6.c i-SCOOP: Artificial intelligence (AI) and cognitive computing: what, why and where

6.d i-SCOOP: The interaction and convergence of IoT and AI at work

6.e Flavio Bonomi, Rodolfo Milito, Jiang Zhu, Sateesh Addepalli: Fog Computing and Its Role in the Internet of Things; Workshop on Mobile Cloud Computing (MCC '12), Aug 2012; DOI: 10.1145/2342509.2342513

6.f Ivan Stojmenovic, Sheng Wen: The fog computing paradigm: Scenarios and security issues; Federated Conference on Computer Science and Information Systems, Sep 2014; DOI: 10.15439/2014F503

6.g Shanhe Yi, Cheng Li, Qun Li: Survey of Fog Computing: Concepts, Applications and Issues; Mobidata'15, Proceedings of the 2015 Workshop on Mobile Big Data, Jun 2015, DOI: 10.1145/2757384.2757397

6.h Yifan Wang, Tetsutaro Uehara, Ryoichi Sasaki: Fog computing: Issues and challenges in security and forensics; IEEE 39th Annual Computer Software and Applications Conference, Vol. 3, Jul 2015; DOI: 10.1109/COMPSAC.2015.173

6.i Tom H. Luan, et al.: Fog Computing: Focusing on Mobile Users at the Edge; Mar 2016; arXiv:1502.01815v3

6.j Lav Gupta, Raj Jain, H. Anthony Chan: Mobile Edge Computing – An Important Ingredient of 5G Networks; Mar 2016

6.k Firas Al-Doghman, et al.: A review on Fog Computing technology; IEEE International Conference on Systems, Man, and Cybernetics (SMC), Oct 2016; DOI: 10.1109/SMC.2016.7844455

6.l Eva Marín Tordera, et al.: What is a Fog Node? A Tutorial on Current Concepts towards a Common Definition; Nov 2016; arXiv:1611.09193v1

6.m Farhoud Hosseinpour, et al.: A Review on Fog Computing Systems; International Journal of Advancements in Computing Technology(IJACT), Vol. 8(5), Dec 2016

6.n Mung Chiang, Tao Zhang: Fog and IoT: An Overview of Research Opportunities; IEEE Internet of Things Journal, Vol. 3(6), Dec 2016; DOI: 10.1109/JIOT.2016.2584538

6.o Arwa Alrawais, Abdulrahman Alhothaily, Chunqiang Hu, Xiuzhen Cheng: Fog Computing for the Internet of Things: Security and Privacy Issues; IEEE Internet Computing, Vol. 21(2), Mar-Apr 2017; DOI: 10.1109/MIC.2017.37

6.p Mithun Mukherjee, et al.: Security and Privacy in Fog Computing: Challenges; IEEE Access, Vol. 5, Sep 2017; DOI: 10.1109/ACCESS.2017.2749422

6.q Redowan Mahmud, Ramamohanarao Kotagiri, Rajkumar Buyya: Fog Computing: A Taxonomy, Survey and Future Directions; arXiv:1611.05539v4, Oct 2017; DOI: 10.1007/978-981-10-5861-5_5

6.r Saad Khan, Simon Parkinson, Yongrui Qin: Fog computing security: a review of current applications and security solutions; Journal of Cloud Computing: Advances, Systems and Applications, Vol. 6(1), 2017c DOI 10.1186/s13677-017-0090-3

6.s Carla Mouradian, et al.: A Comprehensive Survey on Fog Computing: State-of-the-art and Research Challenges; IEEE Communications Surveys & Tutorials, Oct 2017; DOI: 10.1109/COMST.2017.2771153

6.t Pengfei Hu, Sahraoui Dhelim, Huansheng Ning, Tie Qiu: Survey on fog computing: architecture, key technologies, applications and open issues; Journal of Network and Computer Applications, Vol. 98, Nov 2017; DOI: 10.1016/j.jnca.2017.09.002

6.u David Bermbach, et al.: A Research Perspective on Fog Computing; Workshop on IoT Systems Provisioning and Management for Context-Aware Smart Cities; Nov 2017

6.v Shubha Brata Nath, Harshit Gupta, Sandip Chakraborty, Soumya K Ghosh: A Survey of Fog Computing and Communication: Current Researches and Future Directions; Apr 2018; arXiv:1804.04365v1

6.w Ranesh Kumar Naha, et al.: Fog Computing: Survey of Trends, Architectures, Requirements, and Research Directions, Jul 2018; arXiv:1807.00976v1

6.x Marcus Gomes, Miguel L. Pardal: Cloud vs Fog: assessment of alternative deployments for a latency-sensitive IoT application; Procedia Computer Science, Vol. 130, 2018; DOI: 10.1016/j.procs.2018.04.059

6.y Jaspreet Kaur, Prabhpreet Kaur: A Review: Artificial Neural Network; International Journal of Current Engineering and Technology, Vol. 8(4), July/Aug 2018; DOI: 10.14741/ijcet/v.8.4.2

18.6.1 Time-sensitive IoT/5G Applications

Es gibt einige intelligente IoT Applications, die in gewisser Weise zeitempfindlich (zeitsensitiv, time-sensitiv) sind. Sie werden häufig *Time-sensitive Applications* genannt. Zu dieser Gruppe gehören auch einige intelligente Applikationen, die in 5G-Mobilfunknetzen realisiert werden. Dies zu ermöglichen, ist die wesentliche Aufgabe von Edge und Fog Computing. Für weitere Information über zeitempfindliche IoT/5G Applications siehe:

6.1.a ETSI GS MEC 002 v1.1.1: Mobile Edge Computing (MEC): Technical Requirements; Mar 2016

6.1.b ETSI GS MEC 003 v1.1.1: Mobile Edge Computing (MEC): Framework and Reference Architecture; Mar 2016

6.1.c Rodrigo Roman, Javier Lopez, Masahiro Mambo: Mobile Edge Computing, Fog et al.: A Survey and Analysis of Security Threats and Challenges; arXiv:1602.00484v2, Nov 2016

6.1.d Ashkan Yousefpour, Genya Ishigaki, Jason P. Jue: Fog Computing: Towards Minimizing Delay in the Internet of Things; IEEE International Conference on Edge Computing (EDGE), Jun 2017; DOI: 10.1109/IEEE.EDGE.2017.12

6.1.e Gaolei Li, Jianhua Li, and Jun Wu: Fog-enabled Edge Learning for Cognitive Content-Centric Networking in 5G; arXiv:1808.09141v1, Aug 2018

18.6.2 Intelligent IoT, Cognitive IoT

Intelligent IoT

Ein wichtiger Trend bei der Weiterentwicklung von IoT trägt dazu bei, künstliche Intelligenz (*Artificial Intelligence*, AI) und die in IoT-Clouds gesammelten, immensen Datenmengen, sog. *Big Data*, im IoT zu analysieren (*Big Data Analytics*). Die Ergebnisse dieser Analysen sollen nicht nur dazu dienen, verschiedene IoT-Komponenten und -Objekte zum Erfassen von Informationen zu befähigen, damit sie diese Menschen signalisieren/anzeigen und die Informationen von Menschen genutzt werden. Die IoT-Komponenten und -Objekte sollen mittels der erfassten Informationen auch selbst lernen und sich dadurch in die Lage versetzen, intelligent (so wie Menschen, aber ohne menschliche Hilfe) auf verschiedene Situationen und plötzlich auftretende Ereignisse zu reagieren. Besäße das IoT die eben geschilderten Fähigkeiten, würde man von *Intelligent IoT* (IIoT) sprechen.

Cognitive IoT

Die sog. *kognitiven Fähigkeiten* des Menschen ermöglichen es ihm, Signale aus der Umwelt wahrzunehmen und weiterzuverarbeiten. Da die Fähigkeiten von Intelligent IoT im Wesentlichen den kognitiven Fähigkeiten des Menschen entsprechen, wird es auch als *Cognitive IoT* (CIoT) bezeichnet. Es sei hervorgehoben, dass AI, (Big) Data Analytics, Cloud Computing und Fog Computing als die vier wichtigsten treibenden Kräfte, quasi als 'Zugpferde' einer IoT Quadriga [Abb. 18.11-1], bei der IoT-Weiterentwicklung zum Intelligent IoT angesehen werden können.

Für detailliertere Information über Cognitive IoT und Intelligent IoT siehe:

6.2.a Charith Perera, Arkady Zaslavsky, Peter Christen, Dimitrios Georgakopoulos: Context Aware Computing for The Internet of Things: A Survey; IEEE Communications Surveys & Tutorials, Vol. 16(1), First Quarter 2014, DOI: 10.1109/SURV.2013.042313.00197

6.2.b Qihui Wu, et al.: Cognitive Internet of Things: A New Paradigm beyond Connection; Mar 2014; arXiv:1403.2498v1

6.2.c Djallel Eddine Boubiche, et al.: Advanced Industrial Wireless Sensor Networks and Intelligent IoT; IEEE Communications Magazine, Vol. 56(2), Feb 2018; DOI: 10.1109/MCOM.2018.8291108

6.2.d Omer Berat Sezer, Erdogan Dogdu, Ahmet Murat Ozbayoglu: Context-Aware Computing, Learning, and Big Data in Internet of Things: A Survey; IEEE Internet of Things Journal, Vol. 5(1), Feb 2018; DOI: 10.1109/JIOT.2017.2773600

6.2.e Xiaoming He, Kun Wang, Huawei Huang, Bo Liu: QoE-Driven Big Data Architecture for Smart City; IEEE Communications Magazine, Vol. 56(2), Feb 2018c DOI: 10.1109/MCOM.2018.1700231

6.2.f Mehdi Mohammadi, Ala Al-Fuqaha, Mohsen Guizani, Jun-Seok Oh: Semi-supervised Deep Reinforcement Learning in Support of IoT and Smart City Services; IEEE Internet of Things Journal, Vol. 5(2), Apr 2018; DOI: 10.1109/JIOT.2017.2712560

6.2.g Qian Mao, Fei Hu, Qi Hao: Deep Learning for Intelligent Wireless Networks: A Comprehensive Survey; IEEE Communications Surveys & Tutorials (Early Access), Jun 2018; DOI: 10.1109/COMST.2018.2846401

6.2.h Anatol Badach: Internet of Things (IoT) – Quo vadis?, Oct 2018, DOI: 10.13140/RG.2.2.29510.93764

6.2.i Nadeem Javaid, Arshad Sher, Hina Nasir, Nadra Guizani: Intelligence in IoT-Based 5G Networks: Opportunities and Challenges; IEEE Communications Magazine, Vol. 56(10), Oct 2018; DOI: 10.1109/MCOM.2018.1800036

18.6.3 Ambient Intelligence in IoT

Ein wichtiges Ziel der IoT-Entwicklung ist die Verbesserung der Lebensqualität von Menschen. Dies bedeutet, dass Intelligent IoT eine Intelligenz, die sich auf das direkte Lebensumfeld von Menschen bezieht, besitzen muss. Diese wird als *Umweltintelligenz* bzw. *Ambient Intelligence* (AmI) bezeichnet. Ein Beispiel für die Verbesserung der Lebensqualität von Menschen dank AmI im IoT ist *Ambient Assisted Living* (AAL). Als AAL wird ein IoT-Service verstanden, mit dem der Alltag älterer oder gesundheitlich benachteiligter Personen situationsabhängig und unaufdringlich unterstützt werden kann. Für detaillierte Informationen über Bedeutung von Ambient Intelligence in IoT sei verwiesen auf:

6.3.a Emile Aarts, Boris de Ruyte: New research perspectives on Ambient Intelligence; Journal of Ambient Intelligence and Smart Environments, Vol. 1(1), 2009; DOI: 10.3233/AIS-2009-0001

6.3.b A. Dohr, R. Modre-Osprian, M. Drobics, D. Hayn, G. Schreier: The Internet of Things for Ambient Assisted Living; Seventh International Conference on Information Technology: New Generations, Apr 2010; DOI: 10.1109/ITNG.2010.104

6.3.c Kun Wang, Yun Shao, Lei Shu, Guangjie Han, Chunsheng Zhu: LDPA: A Local Data Processing Architecture in Ambient Assisted Living Communications; IEEE Communications Magazine, Vol. 53(1), Jan 2015; DOI: 10.1109/MCOM.2015.7010516

18.6.4 IoT Service Orchestration

Müssen im IoT mehrere Fog Nodes beispielsweise von einer Cloud koordiniert werden, so ist die Nutzung von SDN fast unabdingbar. Bei einem Einsatz von SDN zur Unterstützung von Fog Computing spricht man von *SDN-enabled Fog Computing*. Diese besondere Art von Fog Computing besteht in der Konfigurierung und Steuerung der Fog Nodes durch SDN Controller. Wird ein IoT Service von mehreren Fog Nodes erbracht, so müssen diese entsprechend koordiniert werden, zum Beispiel von einem SDN Controller, der in einem Cloud untergebracht werden kann. Dies kann als eine Art 'IoT Service Orchestrierung' angesehen werden. Für Näheres darüber siehe:

6.4.a Harshit Gupta, et al.: SDFog: A Software Defined Computing Architecture for QoS Aware Service Orchestration over Edge Devices; Sep 2016; arXiv:1609.01190v1

6.4.b Karima Velasquez, et al.: Fog orchestration for the Internet of Everything: state-of-the-art and research challenges; Journal of Internet Services and Applications, Jul 2018; DOI: 10.1186/s13174-018-0086-3

6.4.c Ola Salman, Imad Elhajj, Ali Chehab, Ayman Kayssi: IoT Survey: An SDN and Fog Computing Perspective; Computer Networks, Vol. 143, Jul 2018; DOI: 10.1016/j.comnet.2018.07.020

6.4.d José Santos, Tim Wauters, Bruno Volckaert, Filip De Turck: Fog Computing: Enabling the Management and Orchestration of Smart City Applications in 5G Networks; Entropy, Vol. 20(1), Jan 2018; DOI:10.3390/e20010004

6.4.e Nam Yong Kim, et al.: CF-CloudOrch: container fog node-based cloud orchestration for IoT networks; The Journal of Supercomputing, Jul 2018c DOI: 10.1007/s11227-018-2493-4

6.4.f Salman Taherizadeh, Vlado Stankovski, Marko Grobelni: A Capillary Computing Architecture for Dynamic Internet of Things: Orchestration of Microservices from Edge Devices to Fog and Cloud Providers; Sensors, Vol. 18(9), Sep 2018.

18.7 5G (Generation) Mobile Networks

Was bedeutet 5G?

Mit 5G wird die der heutigen 4. folgende fünfte Generation von Mobilfunknetzen bezeichnet. Es wird davon ausgegangen, dass die Mobilfunknetze der 5. Generation erst um das Jahr 2020 verfügbar sein werden.

5G als Ergänzung zu 4G

Die 5G-Mobilfunknetze, als nächste Generation zellularer Mobilfunknetze, sollen die heutigen 4G-Mobilfunknetze nicht ersetzen, sondern mit erheblich höheren Übertragungskapazitäten und geringeren Latenzzeiten ergänzen. 5G-Mobilfunknetze können auf Basis kleiner Zellen eingerichtet werden und ermöglichen eine Übertragung von Daten über kurze Entfernungen mit großen, in Bereich von Gbit/s liegenden Bitraten. Mit einer breiten Verfügbarkeit von 5G-Mobilfunknetzen wird eine Netzwerkarchitektur geschaffen, die den vielfältigen Anforderungen des IoT sehr gerecht wird.

Auf dem Gebiet 5G Mobile Networks sind insbesondere die Networking-Trends *5G-enabled Mobile IoT Applications*, *Vehicle-to-Everything* (V2X) Services, *SDN and NFV for 5G Mobile Networks*, *5G Network Slicing* und *5G Security* hervorzuheben. Für allgemeine Information über 5G-Mobilfunknetze sei verwiesen auf:

7.a IEEE SPECTRUM: What Does Every Engineer Need to Know about 5G?

7.b SDxCentral: Defining 5G Architecture

7.c Chih-Lin I, Corbett Rowell, Shuangfeng Han, Zhikun Xu, Gang Li, Zhengang Pan: Toward green and soft: a 5G perspective; IEEE Communications Magazine, Vol. 52(2), Feb 2014; DOI: 10.1109/MCOM.2014.6736745

7.d Patrick Kwadwo Agyapong, et al.: Design considerations for a 5G network architecture; IEEE Communications Magazine,, Vol. 52(11), Nov 2014; DOI: 10.1109/MCOM.2014.6957145

7.e NGMN Alliance: 5G White Paper - Executive Version; Dec 2014

7.f Akhil Gupta, Rakesh Kumar Jha: A Survey of 5G Network: Architecture and Emerging Technologies: IEEE Access, Vol. 3, Jul 2015; DOI: 10.1109/ACCESS.2015.2461602

7.g Huawei Technologies: 5G Network Architecture: A High-Level Perspective; Technical Report, 2016

7.h 5G PPP Architecture Working Group: View on 5G Architecture, Version 1.0; Technical Report, Version 1.0, Jul 2016

7.i Bundesministerium für Verkehr und digitale Infrastruktur (Hrsg.): 5G-Strategie für Deutschland: Eine Offensive für die Entwicklung Deutschlands zum Leitmarkt für 5G-Netze und -Anwendungen; Juli 2017

7.j 5G PPP Architecture Working Group: View on 5G Architecture; Version 2.0; Technical Report, Jul 2017

7.k Ian F. Akyildiz, Shuai Nie, Shih-Chun Lin, Manoj Chandrasekaran: 5G roadmap: 10 key enabling technologies; Computer Networks, Vol. 106, 4 Sep 2016; DOI: 10.1016/j.comnet.2016.06.010

7.l 5G PPP Architecture Working Group: View on 5G Architecture; Technical Report, Version 2.0, Jun 2017

7.m ETSI TR 138 913 V14.3.0: 5G; Study on scenarios and requirements for next generation access technologies (3GPP TR 38.913 version 14.3.0 Release 14); Oct 2017

7.n Rojeena Bajracharya, et al.: LWA in 5G: State-of-the-Art Architecture, Opportunities, and Research Challenges; IEEE Communications Magazine, Vol. 56(10), Oct 2018; DOI: 10.1109/MCOM.2018.1701177

18.7.1 5G-enabled Mobile IoT Applications

In zellularen 5G-Mobilfunknetzen werden Funktechnologien genutzt, in denen Signale im elektromagnetischen Spektrum im Millimeterbereich verwendet werden. 5G-Mobilfunknetze basieren also auf der Ausbreitung von sog. Millimeterwellen. Hierdurch sind in diesen mobilen Netzen kleine Zellen und große Übertragungsgeschwin-

digkeiten möglich. Und aus diesem Grund werden 5G-Mobilfunknetze bis zu 100 Mal schneller als bestehende 4G-Mobilfunknetze sein. Dies wird von großer Bedeutung bei der Integration von 5G-Mobilnetztechnologien in das IoT sein – insbesondere für Industrial IoT (IIoT) – und bestimmte andere Branchen, z.B. die Landwirtschaft. Die soeben erwähnten, IoT-relevanten Besonderheiten der 5G-Mobilnetztechnologien werden mit den drei Begriffen eMBB, URLLC und mMTC umschrieben. Dabei steht:

- **eMBB** (*enhanced Mobile Broadband*) für ein erweitertes mobiles Breitband (Nutzung von Millimeterwellen),
- **URLLC** (*Ultra Reliable Low Latency Communications*) für sehr zuverlässige Kommunikation mit geringer Latenz und
- **mMTC** (*massive Machine Type Communications*) für ein breites Spektrum von IoT-Einsatzmöglichkeiten.

Bedeutung von LiFi

Eine besonders wichtige 5G-Netztechnologie ist eine optische Übertragungstechnik, die als LiFi (*Light Fidelity*) bezeichnet wird. Sie wird zukünftig der lokalen Kommunikation, z.B. Realisierung von IoT-Services in großen Räumen, dienen können. LiFi stellt eine neue drahtlose Datenübertragungstechnik dar, die das Infrarot- und sichtbare Lichtspektrum nutzt und eine schnelle, bidirektionale drahtlose lokale Kommunikation ermöglicht. Mit LiFi ist eine Übertragungsgeschwindigkeit von 8 Gbit/s erreichbar. Für nähere Informationen über LiFi siehe [7.1.i].

Internet of Everything

Die Integration neuer G5-Mobilfunknetze in das IoT kann zu einer Erweiterung von IoT führen, bei der von *Internet of Everything* (IoE) gesprochen werden wird. Für weitere Information über verschiedene Arten von 5G-enabled Mobile IoT Applications siehe:

7.1.a i-SCOOP: 5G and IoT in 2018 and beyond: the mobile broadband future of IoT

7.1.b Doruk Sahinel, et al.: Beyond 5G Vision for IOLITE Community; IEEE Communications Magazine, Vol. 55(1), Jan 2017, DOI: 10.1109/MCOM.2017.1600372CM

7.1.c Sotirios K. Goudos, et al.: A survey of IoT Key Enabling and Future Technologies: 5G, Mobile IoT, Sematic Web and Applications; Wireless Personal Communications, Jul 2017; DOI: 10.1007/s11277-017-4647-8

7.1.d Godfrey Akpakwu, et al.: A Survey on 5G Networks for the Internet of Things: Communication Technologies and Challenges; IEEE Access, Vol. 5(12), Dec 2017; DOI: 10.1109/ACCESS.2017.2779844

7.1.e Yuh-Shyan Chen, Yi-Ting Tsai: A Mobility Management Using Follow-Me Cloud-Cloudlet in Fog-Computing-Based RANs for Smart Cities; Sensors, Vol. 18(2), Feb 2018; DOI: 10.3390/s18020489

7.1.f Shancang Li, Li Da Xu, Shanshan Zhao: 5G Internet of Things: A Survey; Journal of Industrial Information Integration, Vol. 10, Jun 2018, DOI: 10.1016/j.jii.2018.01.005

7.1.g By Biswa P. S. Sahoo, et al.: Enabling Millimeter-Wave 5G Networks for Massive IoT Applications; Aug 2018, arXiv:1808.04457v1

7.1.h Nadeem Javaid, Arshad Sher, Hina Nasir, Nadra Guizani: Intelligence in IoT-Based 5G Networks: Opportunities and Challenges; IEEE Communications Magazine, Vol. 56(10), Oct 2018; DOI: 10.1109/MCOM.2018.1800036

7.1.i Harald Haas: LiFi is a paradigm-shifting 5G technology; Reviews in Physics, Vol. 3, Nov. 2018, DOI: 10.1016/j.revip.2017.10.001

18.7.2 Vehicle-to-Everything (V2X) Services

In naher Zukunft wird es verschiedene autonom, d.h. ohne einen menschlichen Fahrer, fahrende Fahrzeuge geben. Um unfallfreies autonomes Fahren zu gewähren, sind aber noch diverse technische Probleme zu lösen. Flächendeckend muss eine besondere, als IoV (*Internet of Vehicles*) bezeichnete Variante von IoT eingerichtet werden. Dabei werden die 5G-Mobilfunknetze eine enorm große Rolle spielen. Jedes autonom fahrende Fahrzeug muss in der Lage sein, alles, was auf Straßen passieren kann, wahrzunehmen und darauf zu reagieren. Das lässt sich nur dadurch erreichen, dass jedes autonom fahrende Vehikel, nicht nur mit anderen Fahrzeugen, sondern mit allen Dingen unterwegs auf eine spezielle Art und Weise kommunizieren kann. Diese Art der Kommunikation soll durch sog. *Vehicle-to-Everything* (V2X) Services ermöglicht werden.

V2X als Oberbegriff für V2V, V2I, V2N und V2P

Zum sicheren autonomen Fahren müssen Fahrzeuge mit anderen Fahrzeugen (V2V), mit der angrenzenden Infrastruktur (V2I), mit Internet-basierten Netzwerken (V2N) und auch mit Fußgängern (V2P) kommunizieren können. Daher sind die soeben genannten Arten der Kommunikation – V2V (*Vehicle-to-Vehicle*), V2I (*Vehicle-to-Infrastructure*), V2N (*Vehicle-to-Network*) und V2P (*Vehicle-to-Pedestrian*) – die funktionellen Bestandteile von V2X. Somit ist V2X als Oberbegriff für V2V, V2I, V2N und V2P anzusehen. Für Näheres über V2X siehe:

7.2.a Dino Flore: 5G V2X: The automotive use-case for 5G; 2017

7.2.b Mate Boban, et al.: Use Cases, Requirements, and Design Considerations for 5G V2X; Dec 2017, arXiv:1712.01754v1

7.2.c Claudia Campolo, et al.: 5G Network Slicing for Vehicle-to-Everything Services; IEEE Wireless Communications, Vol. 24(6), Dec 2017; DOI: 10.1109/MWC.2017.1600408

7.2.d 5G Americas White Paper: Cellular V2X Communications Towards 5G; March 2018

18.7.3 SDN and NFV for 5G Mobile Networks

Die beiden Konzepte SDN (*Software-Defined Networking*) und NFV (*Network Functions Virtualisation*) gelten als neue, sich gegenseitig ergänzende, Entwicklungstrends auf dem Gebiet Networking und haben einen großen Einfluss auf die Entwicklungen von 5G Mobile Communications. Mit SDN ist es auf der Grundlage von 5G-Netzwerktechnologien möglich, private 5G-Netzwerke (*private 5G Networks*) zur Unterstützung von Multimedia- und Gruppenkommunikation, z.B. in einem Unter-

nehmen, einzurichten. Um private 5G-Netzwerke einrichten zu können, werden sehr häufig verschiedene virtualisierte netzwerkspezifische Funktionen, sog. VNFs (*Virtualised Network Functions*), eingesetzt. Das SDN-Konzept eignet sich ideal zur Einrichtung eines 5G Netzwerkes auf Basis von VNFs, zu dessen Management und Adaption an aktuellen Anforderungen. In diesem Zusammenhang kann von 'SD-enabled 5G Private Networks' gesprochen werden. Für weitere Informationen darüber siehe:

7.3.a Junyu Lai, et al.: Software-defined cellular networking: A practical path towards 5G, International Journal of Communication Networks and Distributed Systems; Vol. 14(1), Jan 2015, DOI 10.1504/IJCNDS.2015.066019

7.3.b Ian F. Akyildiz, Pu Wang, Shih-Chun Lin: SoftAir: A software defined networking architecture for 5G wireless systems; Computer Networks, Vol. 85, Jul 2015; DOI: 10.1016/j.comnet.2015.05.007

7.3.c Ricard Vilalta, et al.: SDN/NFV orchestration of multi-technology and multi-domain networks in cloud/fog architectures for 5G services; 21st OptoElectronics and Communications Conference (OECC) held jointly with 2016 International Conference on Photonics in Switching (PS), Jul 2016

7.3.d 5G PPP Software Networks WG: Vision on Software Networks and 5G; White Paper, Jan 2017

7.3.e Pedro Neves, et al.: Future mode of operations for 5G – The SELFNET approach enabled by SDN/NFV; Computer Standards & Interfaces, Vol. 54(4), Nov 2017; DOI: 10.1016/j.csi.2016.12.008

7.3.f Luis Tello-Oquendo, Ian Akyildiz, Shih-Chun Lin, Vicent Pla: SDN-Based Architecture for Providing Reliable Internet of Things Connectivity in 5G Systems; Jul 2018, HAL Id: hal-01832537

7.3.g Lu Ma, et al.: An SDN/NFV Based Framework for Management and Deployment of Service Based 5G Core Network; China Communications, Vol. 15(10), Oct. 2018; DOI: 10.1109/CC.2018.8485472

18.7.4 5G Network Slicing

In einem speziell für den Support von 5G Mobile Services eingerichteten Datacenter, bzw. in einer speziell hierfür installierten 5G Cloud, können aus einzelnen VNFs (eine Art *Microservices*) isolierte Gruppen untereinander vernetzter VNFs gebildet werden. Eine isolierte Gruppe vernetzter VNFs kann ein privates 5G-Netzwerk bilden. Solch eine Vorgehensweise basiert auf der Realisierung von 5G Network Slicing, und es kommen dabei die Konzepte von SDN und *Network Functions Virtualisation* (NFV) zum Einsatz. 5G Network Slicing wird allgemein als Schlüsseltechnologie zur Bildung spontaner, oft privater, an aktuelle Anforderungen anpassungsfähiger (*adaptiver*) 5G-Netzwerke angesehen. Für weitere Informationen siehe:

7.4.a SDxCentral: What is Network Slicing?

7.4.b SDxCentral: What is Dynamic Network Slicing?

7.4.c Jose Ordonez-Lucena, et al.: Network Slicing for 5G with SDN/NFV: Concepts, Architectures, and Challenges; IEEE Communications Magazine, Vol. 55(5), May 2017; DOI: 10.1109/MCOM.2017.1600935

7.4.d Xenofon Foukas, et al.: Network Slicing in 5G: Survey and Challenges; IEEE Communications Magazine, Vol. 55(5), May 2017), DOI: 10.1109/MCOM.2017.1600951

7.4.e Xin Li, et al.: Network Slicing for 5G: Challenges and Opportunities; IEEE Internet Computing, Vol. 21(5) Sep 2017; DOI: 10.1109/MIC.2017.3481355

7.4.f Tony Saboorian, Amanda Xiang: Network Slicing and 3GPP Service and Systems Aspects (SA) Standard; IEEE Softwarization, Dec 2017

7.4.g Claudia Campolo, et al.: 5G Network Slicing for Vehicle-to-Everything Services; IEEE Wireless Communications, Vol. 24(6), Dec 2017; DOI: 10.1109/MWC.2017.1600408

7.4.h Bessem Sayadi, Laurent Roullet: 5G: Platform and Not Protocol, IEEE Softwarization, Jan 2018

7.4.i Emmanuel Dotaro: 5G Network Slicing and Security; IEEE Softwarization, Jan 2018

18.7.5 5G Network Security

In den zukünftigen 5G-Mobilfunknetzen kommen die Netzwerktechnologien SDN und NFV zum Einsatz, und darüber hinaus werden diese Mobilfunknetze in das IoT integriert. Aus diesem Grund müssen alle Schwachstellen und Bedrohungen, die in IoT, SDN und NFV vorkommen können, auch bei der Analyse der Sicherheit in 5G-Mobilfunknetzen berücksichtigt werden. Die Bildung von virtuellen privaten 5G-Netzwerken mit SDN und NFV, was man als *5G Network Slicing* bezeichnet, schafft dabei auch zusätzlich neue Sicherheitslücken und Bedrohungen. Folglich ist die Sicherheit in zukünftigen 5G-Mobilfunknetzen, kurz als *5G Security* bezeichnet, ein sehr wichtiges, breites und komplexes Forschungs- und Entwicklungsgebiet.

Daher muss die Analyse der Sicherheit in 5G-Mobilfunknetzen global und systematisch durchgeführt werden, um dabei möglichst sämtliche Sicherheitslücken und Bedrohungen zu erfassen und damit die entsprechenden Sicherheitsmaßnahmen einplanen zu können. Eine Sammlung von Sicherheitsrichtlinien zur Analyse der Sicherheit in 5G-Mobilfunknetzen ist in `5G PPP Security WG` [7.5.g] und `x3GPP TS 33.501x` [7.5.k] enthalten. Für weitere Information zu ´5G Security´ ist zu empfehlen:

7.5.a 3GPP 5G Security

7.5.b Huawei White Paper: 5G Security: Forward Thinking

7.5.d SDxCentral: What Are the Top 5G Security Challenges?

7.5.f ITU-T: Security in Telecommunications and Information Technology; An overview of issues and the deployment of existing ITU-T Recommendations for secure telecommunications, September 2015

7.5.g 5G PPP Security WG: 5G PPP Phase1 Security Landscape; Jun 2017

7.5.h Ijaz Ahmad, et al.: 5G Security: Analysis of Threats and Solutions; 2017 IEEE Conference on Standards for Communications and Networking (CSCN), Sep 2017; DOI: 10.1109/CSCN.2017.8088621

7.5.i Emmanuel Dotaro: 5G Network Slicing and Security; IEEE Softwarization, Jan 2018

7.5.j Ijaz Ahmad, et al.: Overview of 5G Security Challenges and Solutions; IEEE Communications Standards Magazine, Vol. 2(1), Mar 2018; DOI: 10.1109/MCOMSTD.2018.1700063

7.5.k 3GPP TS 33.501: Technical Specification Group Services and System Aspects; Security architecture and procedures for 5G system; Sep 2018

18.8 Information-Centric Networking and Services

Host-Centric Networking im Internet

Das Internet dient heutzutage nicht nur der Kommunikation zwischen Rechnern, sondern ist ein universales Informationsnetz, d.h. quasi eine weltweit verteilte Informationsbasis. Da im Internet die Kommunikation zwischen Rechnern, sog. Hosts, heute aber noch im Vordergrund steht, ist es aktuell ein rechnerzentriertes Netz – also ein Host-Centric Network. Die Charakteristik eines rechnerzentrierten Netzes besteht darin, dass jede Internetadresse in der Form von URL (*Uniform Resource Locator*) bzw. von URI (*Uniform Resource Identifier*), in der Regel mithilfe von DNS, auf eine Host-Adresse aufgelöst (quasi abgebildet) werden muss. Das heutige DNS stellt de facto ein weltweit verteiltes Resolving System von URLs bzw. URIs auf Host-Adressen dar. Man könnte aber ein anderes globales Resolving-System bzw. mehrere lokale, weltweite verteilte, kleinere Resolving-Systeme nutzen. Die Folge wäre u.a. beispielsweise Information-Centric Networking.

Information-Centric Networking (ICN)

Damit das Internet zukünftig den stark wachsenden, mit einem universalen Informationsnetz verbundenen Herausforderungen gerecht werden kann, muss das ihm zugrunde liegende rechnerzentrierte Kommunikationsprinzip um ein neues, zukunftsweisendes Prinzip erweitert werden. Das Kommunikationsprinzip im Internet sollte in Zukunft nicht nur *Rechner-zentriert* (*Host-Centric*), sondern auch *Informations-zentriert* (*Information-Centric*) sein. Der Zugriff auf eine gewünschte Information sollte also nur durch die Angabe von deren Namen – und nicht die ihrer Lokation – erfolgen. In diesem Zusammenhang spricht man heute von *Information-Centric Networking*(ICN). Da Daten bzw. verschiedene Content-Arten (z.B. Audio, Video, Streaming Medien) allgemein betrachtet als Arten von Information anzusehen sind, werden als Synonyme für ICN häufig die Begriffe NDN (*Named Data Networking*) und CCN (*Content-Centric Networking*) verwendet.

Internet-Dual-Mode: 'Host-Centric' und 'Information-Centric'

Es sei angemerkt, dass das Internet (theoretisch betrachtet) nach zwei Prinzipien, also 'Host-Centric' und 'Information-Centric', genutzt werden kann. Somit könnte es in einer Art Dual-Mode betrieben werden. Zu diesem Zweck müssen weitere Resolving-Systeme – zusätzlich zum DNS – installiert werden. Mit deren Hilfe werden Informationsnamen, z.B in Form von URNs (*Uniform Resource Names*), auf spezielle Containern, quasi Speicherplätzen, in denen auf die Rechnern mit gewünschten Informationen verwiesen wird, aufgelöst/abgebildet. Nach einem ähnlichen Prinzip funktioniert das als ENUM bezeichnete Resolving von Telefonnummern auf IP-Adressen zwecks Internettelefonie [Abschnitt 5.7.2].

LoST als Support zu IC Services

ICN ist ein neuer Ansatz, der über das Internet den direkten Zugriff auf mit eindeutigen Namen/Identifikationen versehene Informationen, unabhängig von deren Herkunft und aktueller Lokation im Internet ermöglicht, ohne hierfür vorher Verbindungen zu Rechnern mit diesen Informationen aufbauen zu müssen. Es wird in diesem Zusammenhang von *IC Services* (IC: Information-Centric) gesprochen. Die Erbringung verschiedener Arten von IC Services lässt sich gut mit dem Konzept *Location-to-Service Translation* (LoST) unterstützen. LoST kann dazu verwendet werden, mittels mobiler Endgeräte (Smartphones, Bordcomputern in Autos, ...) unterwegs Informationen darüber zu erhalten, welche IoT-Services es in direkter Nähe gibt, damit diese daraufhin mit den mobilen Endgeräten abgerufen werden können [8.e].

Auf dem Konzept von ICN basieren u.a. die folgenden Networking-Trends: *Software-Defined ICN* (SD ICN), *IC Internet of Things* (IC IoT). Insbesondere sind *IC Services in Smart Cities* hervorzuheben. Diese eben genannten Trends werden mit dem Trend *Information-Centric Networking Security* (ICN Security) zusammengefasst. Für allgemeine Information darüber siehe:

8.a Ngoc-Thanh Dinh, Younghan Kim: Potential of information-centric wireless sensor and actor networking; International Conference on Computing, Management and Telecommunications (ComManTel), Jan 2013; DOI: 10.1109/ComManTel.2013.6482384

8.b Aytac Azgin, Ravishankar Ravindran, Guoqiang Wang: A Scalable Mobility-Centric Architecture for Named Data Networking; Jun 2014; arXiv:1406.7049v1

8.c Marica Amadeo, Claudia Campolo, Antonella Molinaro, Giuseppe Ruggeri: Content-centric wireless networking: A survey; Computer Networks, Vol. 72, Oct 2014; DOI: 10.1016/j.comnet.2014.07.003

8.d RFC 7476: Information-Centric Networking: Baseline Scenarios, Mar 2015

8.e Anatol Badach: LoST - Location-to-Service Translation; In book: Protokolle und Dienste der Informationstechnologie; WEKA, Ed.: Heinz Schulte; Aug 2015; DOI: 10.13140/RG.2.1.4314.2887

8.f RFC 7927: Information-Centric Networking (ICN) Research Challenges; Jul 2016

8.g José Quevedo, et al.: On the application of contextual IoT service discovery in Information Centric Networks; Computer Communications, Vol. 89–90, Sep 2016; DOI: 10.1016/j.comcom.2016.03.011

8.h Marica Amadeo, et al.: Information-centric networking for M2M communications: Design and deployment; Computer Communications, Vol. 89–90, Sep 2016; DOI: 10.1016/j.comcom.2016.03.009

8.i Bin Da, Richard Li, Xiaofei Xu: ID Oriented Networking (ION) for IoT Interoperation; 2017 Global Internet of Things Summit (GIoTS), Jun 2017; DOI: 10.1109/GIOTS.2017.8016237

8.j Dennis Grewe, et al.: Information-Centric Mobile Edge Computing for Connected Vehicle Environments: Challenges and Research Directions; MECOMM '17 Proceedings of the Workshop on Mobile Edge Communications, Aug 2017; DOI: 10.1145/3098208.3098210

8.k Cenk Gündogan, et al.: Information-Centric Networking for the Industrial IoT; ICN '17 Proceedings of the 4th ACM Conference on Information-Centric Networking, Sep 2017; DOI 10.1145/3125719.3132099

8.l Bin Da, et al.: Identity/Identifier-Enabled Networks (IDEAS) for Internet of Things (IoT); 2018 IEEE 4th World Forum on Internet of Things (WF-IoT), Feb 2018; DOI: 10.1109/WF-IoT.2018.8355102

18.8.1 Software-Defined ICN (SD ICN)

Moderne Geräte wie Smartphones, Laptops und Tablets, die über drahtlose Netzwerke den Zugang zum Internet haben, werden immer beliebter. Sie ermöglichen den Benutzern Mobilität und Flexibilität beim Zugriff auf das Internet an jedem Ort (z.B. zu Hause, im Büro, in Geschäften, Autos) und zu jeder Zeit. Die Internet-Nutzung mithilfe dieser Geräte sollte in der Zukunft nicht nur wie heute rechnerzentrisch (Host-Centric) sein, sondern auch informationszentrisch (Information-Centric). Es sollte möglich sein, die gewünschte Information durch die Angabe ihrer strukturierten Namen – einer Art URNs (*Uniform Resource Names*) – aus dem Internet zu erhalten, ohne dabei eine Suchmaschine (wie z.B. Google) nutzen zu müssen. Um eine solche Wunschvorstellung zu verwirklichen, also ein ICN einzurichten, können einige Konzepte aus dem Gebiet SDN (*Software-Defined Networking*) übernommen werden. Ist dies der Fall, spricht man von *Software-Defined ICN* (SD ICN). Dabei liefert SDN einige Ideen zur Bereitstellung von ICN-Architekturen und kann die Funktionalität und Verwaltung von ICNs erheblich vereinfachen. Für weitere Information über SD ICN ist zu empfehlen:

8.1.a Jinfan Wang, Wei Gao, Yuqing Liang, Rui Qin, Jianping Wang, Shucheng Liu: SD-ICN: An interoperable deployment framework for software-defined information-centric networks; IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), May 2014; DOI: 10.1109/INFCOMW.2014.6849199

8.1.b Shuai Gao, Yujing Zeng, Hongbin Luo, Hongke Zhang: Scalable Area-based Hierarchical Control Plane for Software Defined Information Centric Networking; 23rd International Conference on Computer Communication and Networks (ICCCN), Aug 2014; DOI: 10.1109/ICCCN.2014.6911839

8.1.c Niels L. M. van Adrichem, Fernando A. Kuipers: NDNFlow: Software-defined Named Data Networking, 1st IEEE Conference on Network Softwarization (NetSoft), Apr 2015; DOI: 10.1109/NETSOFT.2015.7116131

8.1.d Suyong Eum, Masahiro Jibiki, Masayuki Murata, Hitoshi Asaeda, Nozomu Nishinaga: A design of an ICN architecture within the framework of SDN; Seventh International Conference on Ubiquitous and Future Networks, Jul 2015; DOI: 10.1109/ICUFN.2015.7182521

8.1.e Alex F R Trajano, Marcial P Fernandez: ContentSDN: A Content-Based Transparent Proxy Architecture in Software-Defined Networking; IEEE 30th International Conference on Advanced Information Networking and Applications (AINA), Mar 2016; DOI: 10.1109/AINA.2016.103

8.1.f Sergio Charpinel, Celso Alberto Saibel Santos, Alex Borges Vieira, Magnos Martinello, Rodolfo Villaca: SDCCN: A Novel Software Defined Content-Centric Networking Approach; IEEE 30th International Conference on Advanced Information Networking and Applications (AINA), Mar 2016; DOI: 10.1109/AINA.2016.86

8.1.g Changyou Xing, Ke Ding, Chao Hu, Ming Chen, Bo Xu: SD-ICN: Toward Wide Area Deployable Software Defined Information Centric Networking; KSII Transactions on Internet and Information Systems, Vol. 10(5), May 2016; DOI: 10.3837/tiis.2016.05.017

8.1.h Anwar Kalghoum, Mettali Gammar: Towards New Information Centric Networking Strategy Based on Software Defined Networking; IEEE Wireless Communications and Networking Conference (WCNC), Mar 2017; DOI: 10.1109/WCNC.2017.7925536

8.1.i Rihab Jmal, Lamia Chaari Fourati: Content-Centric Networking Management Based on Software Defined Networks: Survey; IEEE Transactions on Network and Service Management, Vol. 14(4), Dec 2017; DOI: 10.1109/TNSM.2017.2758681

8.1.j Qing-Yi Zhang, Xing-Wei Wang, Min Huang, Ke-Qin Li, Sajal K. Das: Software Defined Networking Meets Information Centric Networking: A Survey; IEEE Access, Vol. 6, Jul 2018; DOI: 10.1109/ACCESS.2018.2855135

18.8.2 Information-Centric IoT (IC IoT)

IC Internet of Things (IC IoT)

Die Ideen von ICN werden auch in das Internet of Things (IoT) integriert. In diesem Zusammenhang wird von *ICN-based IoT* oder von *ICN-enabled IoT* gesprochen, und man bezeichnet diese Art von IoT kurz als *ICN IoT*. Um Information-Centric (IC) Services im IoT zu ermöglichen, können sowohl spezielle Clouds als auch Fog Nodes eingesetzt werden [Abb. 17.1-4], um IoT-relevante Informationen zentral in Clouds oder entlang des Internet Edge lokal in Fog Nodes zu speichern. In diesem Fall können Clouds als 'Central Information Caches' und Fog Nodes als 'Local Information Caches' dienen. Denn in diesen Caches gespeicherte Informationen können so strukturierte Namen wie z.B. URNs zugewiesen werden, damit sie durch die Angabe von URNs von mobilen Endgeräten abgerufen werden können. Für Näheres darüber siehe:

8.2.a Marica Amadeo, Claudia Campolo, Antonio Iera, Antonella Molinaro: Named data networking for IoT: An architectural perspective; European Conference on Networks and Communications (EuCNC), Jun 2014; DOI: 10.1109/EuCNC.2014.6882665

8.2.b Emmanuel Baccelli, et al.: Information Centric Networking in the IoT: Experiments with NDN in the Wild; Proceeding ACM-ICN '14, Proceedings of the 1st ACM Conference on Information-Centric Networking, Sep 2014: DOI: 10.1145/2660129.2660144

8.2.c José Quevedo, Daniel Corujo, Rui Aguiar: A case for ICN usage in IoT environments; IEEE Global Communications Conference, Dec 2014; DOI: 10.1109/GLOCOM.2014.7037227

8.2.d Marica Amadeo, Claudia Campolo, Antonella Molinaro: Forwarding strategies in named data wireless ad hoc networks: Design and evaluation; Journal of Network and Computer Applications, Vol. 50, Apr 2015; DOI: 10.1016/j.jnca.2014.06.007

8.2.e Riccardo Petrolo, Valeria Loscri, Nathalie Mitton: Towards a smart city based on cloud of things, a survey on the smart city vision and paradigms; Transactions on Emerging Telecommunications Technologies, Mar 2015; DOI: 10.1002/ett.2931

8.2.f Marica Amadeo, Claudia Campolo, Antonio Iera, Antonella Molinaro: Information Centric Networking in IoT scenarios: the Case of a Smart Home; IEEE International Conference on Communications (ICC), Jun 2015; DOI: 10.1109/ICC.2015.7248395

8.2.g Zhou Su, Qichao Xu: Content Distribution over Content-Centric Mobile Social Networks in 5G; IEEE Communications Magazine, Vol. 53(6) , Jun 2015; DOI: 10.1109/MCOM.2015.7120047

8.2.h Marica Amadeo, et al.: Information-centric networking for the internet of things: challenges and opportunities; IEEE Network, Vol. 30(2), Mar-Apr 2016; DOI: 10.1109/MNET.2016.7437030

8.2.i Yongrui Qin, et al.: When things matter: A survey on data-centric Internet of Things; Journal of Network and Computer Applications, Vol. 64, Apr 2016; DOI: 10.1016/j.jnca.2015.12.016

8.2.j Soumya Kanti Datta, et al.: DataTweet: An Architecture Enabling Data-Centric IoT Services; IEEE Region 10 Symposium (TENSYMP), May 2016; DOI: 10.1109/TENCONSpring.2016.7519430

8.2.k Marica Amadeo, et al.: Information-centric networking for M2M communications: Design and deployment; Computer Communications, Vol. 89–90, Sep 2016; DOI: 10.1016/j.comcom.2016.03.009

8.2.l Xuan Liu, Zhuo Li, Peng Yang, Yongqiang Dong: Information-centric mobile ad hoc networks and content routing: A survey; Ad Hoc Networks, Vol. 58, 2017; DOI: 10.1016/j.adhoc.2016.04.005

8.2.m Toshihiko Kurita, Izuru Sato, Kenichi Fukuda, Toshitaka Tsuda: An Extension of Information-Centric Networking for IoT Applications; International

Conference on Computing, Networking and Communications (ICNC), Jan 2017; DOI: 10.1109/ICCNC.2017.7

8.2.n draft-zhang-icnrg-icniot-architecture-01: ICN based Architecture for IoT; Jul 2017

8.2.o Sripriya Srikant Adhatarao, Mayutan Arumaithurai, Xiaoming Fu: FOGG: A Fog Computing Based Gateway to Integrate Sensor Networks to Internet; 29th International Teletraffic Congress (ITC 29), Sep 2017; DOI: 10.23919/ITC.2017.8065709

8.2.p Marica Amadeo, Claudia Campolo, Antonella Molinaro: A novel hybrid forwarding strategy for content delivery in wireless information-centric networks; Computer Communications, Vol. 109, Sep 2017; DOI: 10.1016/j.comcom.2017.05.012

8.2.q Maroua Meddeb: Information-Centric Networking, A natural design for IoT applications ?; INSA de Toulouse, 2017; HAL Id: tel-01661302

8.2.r Soumya Kanti Datta, Christian Bonnet: Next-Generation, Data Centric and End-to-End IoT Architecture Based on Microservices, 3rd International Conference on Consumer Elecrtronics (ICCE), Jun 2018

8.2.s Sobia Arshad, et al.: Recent Advances in Information-Centric Networking based Internet of Things (ICN-IoT); arXiv:1710.03473v3; Oct 2018

18.8.3 Information-Centric Services für Smart Cities

Location-based IC Services

Wir nutzen mobile Endgeräte wie Smartphones und Tablets mit Internetzugang. Diese Endgeräte verfügen in der Regel über eine wichtige Komponente – nämlich ein Navigationssystem. Dadurch können sie dem Internet die Angaben über ihre Lokation/Position auf der Erdkugel liefern. Das bereits eben erwähnte Konzept LoST kann dazu verwendet werden, die Lokation mobiler Endgeräte im Internet so zu nutzen, dass verschiedene, von den mobilen Internetnutzern nicht weit entfernte, verfügbare Dienste/Angebote ermittelt werden. Daraufhin können sie den Internetnutzern bekannt gemacht und somit von diesen unterwegs in Anspruch genommen werden. Dies kann zu neuen Formen von Internetdiensten führen. Diese Services kann man als Location-based IC Services bezeichnen [Abschnitt 17.4]

IC Services für Smart Cities

IC Services und insbesondere Location-based IC Services haben eine wichtige Bedeutung bei der Entwicklung der Konzepte für Smart Cities. Damit man diese Services realisieren kann, ist ein verteiltes (dezentrales) Resolving-System in Form von strukturierten Informationsnamen erforderlich, also von den lokalen/ortsgebundenen Informationen zugewiesenen URNs auf die Adressen von Rechnern mit diesen Informationen. Um Location-based IC Services mit einem mobilen Endgerät in Anspruch zu nehmen, muss die Angabe über die Geo-Lokation dieses Endgeräts auf die IP-Adresse des benötigten lokalen Resolving-Systems von strukturierten Informationsnamen (URNs nämlich) auf die Adressen von Rechnern mit diesen gewünschten Informationen erfolgen. Für weitere Information über IC Services für Smart Cities sei verwiesen auf:

8.3.a I. Cianci, G. Piro, L. A. Grieco, G. Boggia, P. Camarda: Content Centric Services in Smart Cities; Sixth International Conference on Next Generation Mobile Applications, Services and Technologies; Sep 2012; DOI 10.1109/NGMAST.2012.20

8.3.b G. Piro, et al.: Information Centric Services in Smart Cities; Journal of Systems and Software, Vol. 88(C), Feb 2014; DOI: 10.1016/j.jss.2013.10.029

8.3.c Marica Amadeo, et al.: Information-Centric Networking for Connected Vehicles: A survey and Future Perspectives; IEEE Communications Magazine, Vol. 54(2), Feb 2016; DOI: 10.1109/MCOM.2016.7402268

8.3.d Ali Shariat, Ali Tizghadam, Alberto Leon-Garcia: An ICN-Based Publish-Subscribe Platform to Deliver UAV Service in Smart Cities; IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Apr 2016, DOI: 10.1109/INFCOMW.2016.7562167

8.3.e Rida Khatoun, Sherali Zeadally: Smart cities: concepts, architectures, research opportunities; Communications of the ACM, Vol. 59(8), Aug 2016; DOI: 10.1145/2858789

8.3.f LASeR: Lightweight Authentication and Secured Routing for NDN IoT in Smart Cities; Mar 2017; arXiv:1703.08453v1

8.3.g Meng Wang, et al.: Toward mobility support for information-centric IoV in smart city using fog computing; IEEE International Conference on Smart Energy Grid Engineering (SEGE), Aug 2017; DOI: 10.1109/SEGE.2017.8052825

8.3.h M. Govoni, et al.: An Information-Centric Platform for Social- and Location-Aware IoT Applications in Smart Cities; EAI Endorsed Transactions on Internet of Things, Vol. 17(9), Aug 2017; DOI: 10.4108/eai.31-8-2017.153049

8.3.i Carmen Rotuna, et al.: Smart City Applications Built on Big Data Technologies and Secure IoT;

8.3.j Yang Zhang, Zehui Xiong, Dusit Niyato, Ping Wang, Zhu Han: Market-Oriented Information Trading in Internet of Things (IoT) for Smart Cities; Jun 2018, arXiv:1806.05583v1

8.3.k Sobia Arshad, et al.: Towards Information-Centric Networking (ICN) Naming for Internet of Things (IoT): The Case of Smart Campus; Jun 2018; arXiv:1711.10304v1

18.8.4 ICN Security

Das ICN ist ein neues Kommunikationsparadigma, das sich auf das Abrufen von Informationen aus dem Internet bzw. aus einem privaten Netzwerk konzentriert, wobei die zu abzurufenden Informationen mit ihren Namen angegeben werden und nicht – wie im heutigen Internet – mit den Adressen von Rechnern, auf denen sie zu finden sind. Ein derartiges neues Kommunikationsparadigma bringt auch neue Sicherheitsprobleme mit sich. Beim ICN ist die Sicherung der Information, also der Inhalte selbst, auf den ersten Blick viel wichtiger als die Probleme der klassischen Netzwerksicherheit. Um die Sicherheitsziele in diesem neuen ICN-Paradigma zu erreichen, ist ein umfassendes Verständnis der ICN-spezifischen Angriffe, ihrer Klassifizierung und der

Lösungsvorschläge von entscheidender Bedeutung. Es gibt verschiedene Angriffsarten – und zwar solche, die direkt nur ICN betreffen, und solche, die sich indirekt auf ICN auswirken. Für weitere Information über ICN Security ist zu empfehlen:

8.4.a Abdelberi Chaabane, et al.: Privacy in Content-Oriented Networking: Threats and Countermeasures; ACM SIGCOMM Computer Communication Review, Vol. 43(3), Jul 2013; DOI 10.1145/2500098.2500102

8.4.b Nikos Fotiou, et al.: A Framework for Privacy Analysis of ICN Architectures; Proc. Second Annual Privacy Forum (APF), Springer, 2014; DOI 10.1007/978-3-319-06749-0_8

8.4.c Eslam G. AbdAllah, Hossam S. Hassanein, Mohammad Zulkernine: A Survey of Security Attacks in Information-Centric Networking; IEEE Communications Surveys & Tutorials, Vol. 17(3), Jan 2015; DOI: 10.1109/COMST.2015.2392629

8.4.d Muhammad Aamir, Syed Mustafa Ali Zaidi: Denial-of-service in content centric (named data) networking: a tutorial and state-of-the-art survey; Security and Communication Networks, 2015; DOI: 10.1002/sec.1149

8.4.e Mahdi Aiash, Jonathan Loo: An integrated authentication and authorization approach for the network of information architecture; Journal of Network and Computer Applications, Vol. 50, Apr 2015; DOI: 10.1016/j.jnca.2014.06.004

8.4.f Roman Lutz: Security and Privacy in Future Internet Architectures: Benefits and Challenges of Content Centric Networks; Jan 2016; arXiv:1601.01278v2

8.4.g RFC 7945: Information-Centric Networking: Evaluation and Security Considerations; Sep 2016

8.4.h Sabrina Sicari, er al.: A Secure ICN-IoT Architecture; IEEE International Conference on Communications; May 2017; DOI: 10.1109/ICCW.2017.7962667

8.4.i Reza Tourani, Travis Mick, Satyajayant Misra, Gaurav Panwar: Security, Privacy, and Access Control in Information-Centric Networking: A Survey; Jun 2017, arXiv:1603.03409v

18.9 Time-sensitive und Deterministic Networking

IT/OT-Konvergenz

Die Migration zum IoT (insbesondere zum IoT im industriellen Bereich, also zum Industrial IoT (IIoT)) führt dazu, dass zwei Bereiche in industriellen Unternehmen konvergieren: nämlich der Bereich IT (*Information Technology*, Informationstechnologie) und der Bereich OT (*Operational Technology*, Betriebstechnologie/Betriebstechnik). In diesem Zusammenhang und in Verbindung zur Industrie 4.0 spricht man von der *IT/OT-Konvergenz*. Diese neue Konvergenz stellt besondere Anforderungen an Netzwerke und Kommunikationsprotokolle in industriellen Bereichen.

Anforderungen der IT/OT-Konvergenz

In der Vergangenheit war die Trennung der Bereiche IT und OT vor allem sachlich begründet, denn die Bereiche IT und OT hatten vorher getrennte Aufgabenprofile und aus diesem Grund waren in diesen beiden Bereichen verschiedene Netzwerkkonzepte und Kommunikationsprotokolle im Einsatz. In OT-Bereichen wurden lange Zeit in sich

geschlossene, proprietäre Systeme eingesetzt und diese verfügten oft über keine Verbindungen zur Außenwelt. Die IT/OT-Konvergenz, insbesondere die Verwirklichung von IIoT, verlangt aber, dass neue Anforderungen an industrielle Netzwerke gestellt werden müssen. Dabei unterscheidet man zwischen zwei Arten von Anforderungen, und um diese zu erfüllen, wurden diese beiden Arten von Networking konzipiert, und sie werden noch weiter intensiv entwickelt. Es handelt sich um: *Time-Sensitive Networking* (TSN) und *Deterministic Networking* (DetNet). Für weitere Information sei verwiesen auf:

9.a Martin Wollschlaeger, Thilo Sauter, Juergen Jasperneite: The Future of Industrial Communication: Automation Networks in the Era of the Internet of Things and Industry 4.0; IEEE Industrial Electronics Magazine, Vol. 11(1), Mar 2017; DOI: 10.1109/MIE.2017.2649104

9.b Norman Finn: Time-sensitive and Deterministic Networking; Whitepaper; Huawei Technologies Co. Ltd; Jul 2017

9.c Ahmed Nasrallah, et al.: Ultra-Low Latency (ULL) Networks: The IEEE TSN and IETF DetNet Standards and Related 5G ULL Research; Sep 2018; arXiv:1803.07673v3

18.9.1 Time-Sensitive Networking

Time-sensitive Networking (TSN)

Eine Art von Anforderungen an industrielle Netzwerke verlangt von Netzwerken die Unterstützung von zeitsensitiven/zeitsensiblen Prozessen. Um solche Anforderungen zu erfüllen, muss die zeitsensitive Kommunikation in industriellen Netzwerken möglich sein. In diesem Zusammenhang spricht man von *Time-Sensitive Networking* (TSN) bzw. von zeitsensitiven Netzwerken. Von einem *zeitsensitiven Netzwerk* werden u.a. die folgenden Eigenschaften verlangt: minimale Übertragungszeiten auf Links, abschätzbare und garantierte Ende-zu-Ende-Latenzzeiten, minimale Paketverlustraten und hohe Verfügbarkeiten der Verbindungen.

Real-Time Ethernet und IEEE 802.1

Um Ethernet als Netzwerktechnologie für TSN einsetzen zu können, muss die Ethernet-Technologie hierfür entsprechend funktionell so erweitert werden, dass sie einige Echtzeitfähigkeiten besitzt. Daher wird die speziell zur Unterstützung von TNS erweiterte Ethernet-Technologie als *Real-Time Ethernet* (*Echtzeit-Ethernet*) bezeichnet. Ethernet ist somit aus dem industriellen Bereich nicht mehr wegzudenken. Um für industrielle Anwendungen zukünftig eine standardisierte Lösung für Echtzeit-Ethernet bereitzustellen, wurde die Arbeitsgruppe Time-Sensitive Networking (TSN) bei der IEEE 802.1 mit dem Ziel gegründet, Standards für Echtzeit-Ethernet so zu entwickeln, dass industrielle Netzwerke auf der Grundlage von Echtzeit-Ethernet den Anforderungen industrieller Anwendungen gerecht werden. Für Näheres über Time-sensitive Networking, Real-Time Ethernet und die Standards von der IEEE 802.1 siehe:

9.1.a IEEE 802: Time-Sensitive Networking (TSN) Task Group

9.1.b George A. Ditzel III, Paul Didier: Time Sensitive Network (TSN) Protocols and use in EtherNet/IP Systems; ODVA Industry Conference & 17th Annual Meeting, Oct 2015

9.1.c Andre Hennecke, Stephan Weyer: Time-Sensitive Networking in modularen Industrie 4.0 Anlagen; Conference: Automation 2017

9.1.d Levi Pearson: TSN IN LINUX; Nov 2017

9.1.e Norman Finn: Introduction to Time-Sensitive Networking; IEEE Communications Standards Magazine, Vol. 2(2), Jun 2018; DOI: 10.1109/MCOMSTD.2018.1700076

9.1.f Wilfried Steiner, Silviu S. Craciunas, Ramon Serna Oliver: Traffic Planning for Time-Sensitive Communication; IEEE Communications Standards Magazine, Vol. 2(2) , Jun 2018; DOI: 10.1109/MCOMSTD.2018.1700055

9.1.g Bharat Bansal: Divide-and-Conquer Scheduling for Time-sensitive Networks; Master Thesis; University of Stuttgart, Institute of Parallel and Distributed Systems, Mar 2018

9.1.h John L. Messenger: Time-Sensitive Networking: An Introduction; IEEE Communications Standards Magazine, Vol. 2(2), Jun 2018; DOI: 10.1109/MCOMSTD.2018.1700047

9.1.i Janos Farkas, Lucia Lo Bello, Craig Gunther: Time-sensitive networking standards; IEEE Communications Standards Magazine, Vol. 2(2), Jun 2018; DOI: 10.1109/MCOMSTD.2018.8412457

9.1.j Paul Pop, Michael Lander Raagaard, Marina Gutiérrez, Wilfried Steiner: Enabling Fog Computing for Industrial Automation Through Time-Sensitive Networking (TSN); IEEE Communications Standards Magazine, Vol. 2(2), Jun 2018; DOI: 10.1109/MCOMSTD.2018.1700057

9.1.k Soheil Samii, Helge Zinner: Level 5 by Layer 2: Time-Sensitive Networking for Autonomous Vehicles, IEEE Communications Standards Magazine, Vol. 2(2), Jun 2018; DOI: 10.1109/MCOMSTD.2018.1700079

9.1.l Csaba Simon, Markosz Maliosz, Miklós Máté: Design Aspects of Low-Latency Services with Time-Sensitive Networking; IEEE Communications Standards Magazine, Vol. 2(2), Jul 2018; DOI: 10.1109/MCOMSTD.2018.1700081

9.1.m Carlos San Vicente Gutiérrez, Lander Usategui San Juan, Irati Zamalloa Ugarte, Víctor Mayoral Vilches: Time-Sensitive Networking for robotics; Sep 2018, arXiv:1804.07643v2

18.9.2 Deterministic Networking

Deterministic Networking (DetNet)

Infolge der Konvergenz IT und OT hat sich eine besondere Klasse von Anforderungen an heutige industrielle Netzwerke herauskristallisiert. Die Anforderungen dieser Klasse verlangen von industriellen Netzwerken ähnliche Eigenschaften, die man beispielsweise im S/U-Bahnnetz oder im Busnetz einer Stadt erkennen kann. Weil Züge oder Busse nach einem festgelegten Plan fahren, in der Tat nach einem deterministi-

schen Plan also, ist eine derartige Eigenschaft der industriellen Netzwerke zur Unterstützung einiger Produktionsprozesse von fundamentaler Bedeutung. Es handelt sich hier z.B. um Prozesse, die oft auf der Basis einer Vernetzung von Robotern realisiert werden und ständig nach einem von vornherein festgelegten deterministischen Plan verlaufen müssen. Die Netzwerke, mit denen eine deterministische, nach einem Plan verlaufende Kommunikation realisiert werden kann, werden daher als *deterministische Netzwerke* bezeichnet. Im Hinblick darauf wird von *Deterministic Networking* (DetNet) gesprochen. Die Entwicklung der Konzepte für DetNet wird von der gleichnamigen Working Group bei der IETF (*Internet Engineering Task Force*) koordiniert und standardisiert. Für Näheres ist zu empfehlen:

9.2.a Henrik Austad: Deterministic Networking for Real-Time Systems, (Using TSN and DetNet); Cisco Systems, Oct 2017

9.2.b IETF: draft-ietf-detnet-problem-statement-07: Deterministic Networking Problem Statement; Oct 2018

9.2.c IETF: draft-ietf-detnet-architecture-09: Deterministic Networking Architecture; Oct 2018

9.2.d IETF: draft-ietf-detnet-use-cases-19: Deterministic Networking Use Cases; Oct 2018

9.2.e IETF: draft-ietf-detnet-security-03: Deterministic Networking (DetNet) Security Considerations; Oct 2018

18.9.3 6TiSCH Wireless Industrial Networks

Verfahren TSCH

Um hohe QoS-Anforderungen erfüllen zu können, müssen zum Aufbau von IIoT spezielle Netzwerke zur Unterstützung der industriellen Kommunikation eingesetzt werden. Um Netzwerke mit solchen Eigenschaften aufbauen zu können, wurde das Verfahren TSCH (*Time-Slotted Channel Hopping*) konzipiert, sodass man auch von TSCH-Netzwerken spricht. Dieses Verfahren stellt eine spezielle, zeitsynchrone Realisierung des MAC-Protokolls (*Media Access Control*) zum Aufbau drahtloser, industrieller Netzwerke dar. Im Allgemeinen kann man das Verfahren TSCH als eine Kombination von zwei, seit Langem bei der digitalen Datenübertragung gut bekannten Multiplexverfahren ansehen. Es handelt sich hierbei um eine Kombination des *Frequenzmultiplex* und des synchronen *Zeitmultiplex*, also um eine Variante von Frequenz- und Zeitmultiplexverfahren FTDM (*Frequency-Time Division Multiplexing*) – siehe hierfür z.B. [9.3.h], [9.3.i] .

6TiSCH Wireless Industrial Networks

Die Konzepte und Protokolle für den Einsatz von IPv6 in TSCH-Netzwerken werden von der Working Group 6tisch (IPv6 over the TSCH mode of IEEE 802.15.4e) bei der IETF koordiniert. Dementsprechend spricht man im Zusammenhang mit IPv6 over TSCH von (6)TiSCH. Die Idee von TiSCH ermöglicht das Deterministic Networking. Das heißt, sie ermöglicht die Einrichtung einer neuen Generation drahtloser Netzwerke, in denen die Verzögerungs- bzw. Reaktionszeit, allgemein als Latenz(zeit) bezeichnet, mit einer bestimmten Genauigkeit berechenbar ist. Die Kommunikation zwischen Knoten in jedem TSCH-Network, das ein deterministisches Netzwerk darstellt, verläuft nach dem in Form einer TSCH-Schedule-Matrix festgelegten Kom-

munikationsplan. Die Festlegung eines solchen Plans bezeichnet man als Scheduling in TSCH-Netzwerken. Jede TSCH-Schedule-Matrix kann dynamisch aufgebaut und modifiziert werden. Für Näheres über 6TiSCH Wireless Industrial Networks ist zu verweisen auf:

9.3.a i-SCOOP: IT and OT convergence – two worlds converging in Industrial IoT

9.3.b RFC 7554: Using IEEE 802.15.4e Time-Slotted Channel Hopping (TSCH) in the Internet of Things (IoT): Problem Statement; May 2015

9.3.c Pascal Thubert, Maria Rita Palattella, Thomas Engel: 6TiSCH Centralized Scheduling: when SDN Meet IoT; IEEE Conference on Standards for Communications and Networking (CSCN), Oct 2015; DOI: 10.1109/CSCN.2015.7390418

9.3.d Domenico De Guglielmo, Simone Brienza, Giuseppe Anastasi: IEEE 802.15.4e: A survey; Computer Communications, Vol. 88, Aug 2016; DOI: 10.1016/j.comcom.2016.05.004

9.3.e Gopi Garge: Industrial Internet and the need for guarantees (6TiSCH); 2017

9.3.f ETSI GR IP6 009 V1.1.1: IPv6-based Industrial Internet leveraging 6TiSCH technology; Mar 2017

9.3.g RFC 8180: Minimal IPv6 over the TSCH Mode of IEEE 802.15.4e (6TiSCH) Configuration; May 2017

9.3.h Rodrigo Teles Hermeto, Antoine Gallais, Fabrice Theoleyre: Scheduling for IEEE802.15.4-TSCH and slow channel hopping MAC in low power industrial wireless networks: A survey; Computer Communications, Vol. 114, Dec 2017; DOI: 10.1016/j.comcom.2017.10.004

9.3.i Anatol Badach: TSCH – Time-Slotted Channel Hopping; In book: Protokolle und Dienste der Informationstechnologie; WEKA, Ed.: Heinz Schulte; Jan 2018

9.3.j Anatol Badach: 6P - 6top Protocol; In book: Protokolle und Dienste der Informationstechnologie; WEKA, Ed.: Heinz Schulte; Aug 2018

9.3.k RFC 8480: 6TiSCH Operation Sublayer (6top) Protocol (6P); Nov. 2018

18.9.4 Time-Sensitive SDN

Time-Sensitive SDN

In Time-Sensitive Networks (TS-Networks) muss überwiegend die verbindungsorientierte IP-Kommunikation, ähnlich dem Prinzip MPLS (*Multi-Protocol Label Switching*) [Abschnitt 12.2] realisiert werden. Dabei werden virtuelle Ende-zu-Ende-Verbindungen mit bestimmten QoS-Parametern in TS-Networks dynamisch auf- und abgebaut. Für diesen Zweck eignet sich das SDN-Konzept ideal. Beim Einsatz von SDN in TS-Networks spricht man von *Time-Sensitive SDN* (TS SDN). Um virtuelle Ende-zu-Ende-Verbindungen in TS-Networks – nach einem dem Prinzip MPLS ähnelnden Konzept – dynamisch auf- und abzubauen, ist eine spezielle Control Plane nötig. Beim Einsatz von TS SDN würde man SDN Controller innerhalb dieser Control Plane installieren. Für weitere Information über Time-Sensitive SDN ist zu empfehlen:

9.4.a Jiafu Wan, et al.: Software-Defined Industrial Internet of Things in the Context of Industry 4.0; IEEE Sensors Journal, Vol. 16(20), Oct 2016; DOI: 10.1109/JSEN.2016.2565621

9.4.b Daniel Thiele, Rolf Ernst: Formal analysis based evaluation of software defined networking for time-sensitive Ethernet; Design, Automation & Test in Europe Conference & Exhibition (DATE), Mar 2016

9.4.c Dominik Henneke, Lukasz Wisniewski, Jürgen Jasperneite: Analysis of Realizing a Future Industrial Network by Means of Software-Defined Networking (SDN); IEEE World Conference on Factory Communication Systems (WFCS), May 2016; DOI: 10.1109/WFCS.2016.7496525

9.4.d Naresh Ganesh Nayak, Frank Dürr, Kurt Rothermel: Time-sensitive Software-defined Network (TSSDN) for Real-time Applications: Proceeding RTNS '16 Proceedings of the 24th International Conference on Real-Time Networks and Systems, Oct 2016; DOI: 10.1145/2997465.2997487

9.4.e Rakesh Kumar, et al.: End-to-End Network Delay Guarantees for Real-Time Systems using SDN; May 2017, arXiv:1703.01641v2

9.4.f Ben Schneider, Alois Zoitl, Monika Wenger, Jan Olaf Blech: Evaluating Software-defined Networking for Deterministic Communication in Distributed Industrial Automation Systems; 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Sep 2017; DOI: 10.1109/ETFA.2017.8247594

9.4.g Khandakar Ahmed, et al.: Software Defined Industry Automation Networks; 27th International Telecommunication Networks and Applications Conference (ITNAC), Nov. 2017; DOI: 10.1109/ATNAC.2017.8215391

9.4.h Sebastian Schriegel, Thomas Kobzan, Jürgen Jasperneite: Investigation on a distributed SDN control plane architecture for heterogeneous time sensitive networks; 14th IEEE International Workshop on Factory Communication Systems (WFCS); Jun 2018; DOI: 10.1109/WFCS.2018.8402356

9.4.i Khandakar Ahmed, Jan O. Blech, Mark A. Gregory, Heinz W. Schmidt: Software Defined Networks in Industrial Automation; Journal of Actuator Networks, Vol. 7(3), Aug 2018; DOI: 10.3390/jsan7030033

9.4.j Dave Cronberger: The software defined industrial network; Industrial Ethernet Book, Issue 84/5; Nov 2018

18.10 AI-based Networking

Bedeutung von AI-enabled Networking

Da die heutigen Rechnernetze, also IP-Netze, immer flexibler, heterogener und komplexer werden, bietet es sich an, die Künstliche Intelligenz (KI, AI: *Artificial Intelligence*) in Verbindung mit *Maschinellem Lernen* (ML, *Machine Learning*) in technischen Konzepten zur Optimierung und Management von Rechnernetzen einzusetzen. Folglich führt ein aktueller und breiter Trend auf dem Gebiet Networking zur Entwicklung von Ideen und technischen Konzepten, um zukünftig intelligente Vernetzungen von verschiedenen Rechnern bauen zu können. Im Hinblick darauf ist AI in Verbin-

dung mit ML von zukunftsweisender Bedeutung. Beim Einsatz von AI und ML auf dem Gebiet Networking spricht man von *AI-enabled Networking*. Demzufolge werden zukünftige IP-Netze mit Sicherheit die mithilfe von ML gewonnene AI so nutzen, um sich immer an jede neue Situation optimal adaptieren/anpassen und dadurch auch intelligente Services erbringen zu können. Das volle/optimale Nutzungspotenzial zukünftiger IP-Netze lässt sich nur mit der Unterstützung von AI erreichen.

ML-Techniken zur AI-Gewinnung

Um AI-enabled Networking zu realisieren und dabei insbesondere AI aus den gesammelten Daten gewinnen zu können, ist der Einsatz verschiedener ML-Techniken möglich. Dabei handelt es sich insbesondere um *Deep Learning* (DL) [10.3] und *Reinforcement Learning* (RL) [10.f]. Beide ML-Methoden haben in sog. Neuronalen Netzen (*Neural Networks*) fundamentale Bedeutung [10.n].

Als wichtige Trends auf dem Gebiet AI-enabled Networking gelten die folgenden Trends: *AI-enabled SDN*, *Data-Driven Networking*, *Cognitive Networks*, *Intent-based Networking*, *Autonomic Networking* und *AI, IoT and 5G Convergence*. Für weitere, aber immer noch allgemeine Informationen über AI-enabled Networking und die Nutzung von AI und ML sei verwiesen auf:

10.a i-SCOOP: Artificial intelligence (AI) and cognitive computing: what, why and where

10.b i-SCOOP: Big data in action: definition, value, evolutions, benefits and context

10.c Jeffrey O. Kephart, David M. Chess: The vision of autonomic computing; Computer, Vol. 36(1), Jan 2003; DOI:10.1109/MC.2003.1160055

10.d Xiaofei Wang, Xiuhua Li, Vicctor C. M. Leung: Artificial Intelligence-Based Techniques for Emerging Heterogeneous Network: State of the Arts, Opportunities, and Challenges; IEEE Access, Vol. 3, 2015; DOI: 10.1109/ACCESS.2015.2467174

10.e Zubair Md. Fadlullah, et al.: State-of-the-Art Deep Learning: Evolving Machine Intelligence Toward Tomorrow's Intelligent Network Traffic Control Systems; IEEE Commun. Surveys & Tutorials, Vol. 19(4), 2017; DOI: 10.1109/COMST.2017.2707140

10.f Hongzi Mao, Mohammad Alizadeh, Ishai Menachey, Srikanth Kandulay: Resource Management with Deep Reinforcement Learning; Proceedings of the 15th ACM Workshop on Hot Topics in Networks (HotNets '16), Nov 2016; DOI: 10.1145/3005745.3005750

10.g Nei Kato, Zubair Md. Fadlullah, et al.: The Deep Learning Vision for Heterogeneous Network Traffic Control: Proposal, Challenges, and Future Perspective, IEEE Wireless Communications, Jun 2017; DOI: 10.1109/MWC.2016.1600317WC

10.h Roberto Gonzalez, et al.: Net2Vec: Deep Learning for the Network; BigDAMA '17 Proceedings of the Workshop on Big Data Analytics and Machine Learning for Data Communication Networks, Aug 2017; DOI: 10.1145/3098593.3098596

10.i Albert Mestres: Knowledge-Defined Networking: A Machine Learning based approach for network and traffic modeling: Doctoral Thesis, Universitat Politécnica de Catalunya, Sep 2017

10.j Omer Berat Sezer, Erdogan Dogdu, Ahmet Murat Ozbayoglu: Context-Aware Computing, Learning, and Big Data in Internet of Things: A Survey; IEEE Internet of Things Journal, Vol. 5(1), Feb 2018; DOI: 10.1109/JIOT.2017.2773600

10.k Mowei Wang, et al.: Machine Learning for Networking: Workflow, Advances and Opportunities; IEEE Network, Vol. 32(2), Mar – Apr 2018; DOI: 10.1109/MNET.2017.1700200

10.l ITU: Artificial Intelligence (AI) for Development Series - Report on AI and IoT in Security Aspects; Jul 2018

10.m ATIS: Evolution to an Artificial Intelligence-Enabled Network; Sep 2018

10.n Jaspreet Kaur, Prabhpreet Kaur: A Review: Artificial Neural Network; International Journal of Current Engineering and Technology, Vol. 8(4), July/Aug 2018; DOI: 10.14741/ijcet/v.8.4.2

18.10.1 AI-enabled SDN

Die eben genannten ML-Methoden können bei SDN eingesetzt werden. Die mittels dieser ML-Methoden gewonnene AI kann bei SDN dazu genutzt werden, um die zur Konfiguration und zum Management von Netzwerkkomponenten dienenden SDN Controller zu befähigen, die zu steuernden Netzwerkkomponenten in die Lage versetzen, sich an neue Situation im Netzwerk optimal anzupassen. Werden SDN Controller befähigt, die AI so zu nutzen, um verschiedene Netzwerkkomponenten der aktuellen Situation im Netzwerk entsprechend optimal zu steuern und zu managen, kann von AI-enabled SDN gesprochen werden. Für Näheres darüber siehe:

10.1.a Majd Latah, Levent Toker: Artificial Intelligence Enabled Software Defined Networking: A Comprehensive Overview; arXiv:1803.06818

10.1.b Majd Latah, Levent Toker: Application of Artificial Intelligence to Software Defined Networking: A Survey; Indian Journal of Science and Technology, Vol. 9(44), Nov 2016; DOI: 10.17485/ijst/2016/v9i44/89812

10.1.c Albert Mestres, Alberto Rodriguez-Natal, Josep Carner, et al: Knowledge-Defined Networking; ACM SIGCOMM Computer Communication Review, Vol. 47(3), Jul 2017

10.1.d Elisa Rojas: From Software-Defined to Human-Defined Networking: Challenges and Opportunities; IEEE Network, Vol. 32 (1), Jan - Feb 2018

18.10.2 Data-Driven Networking

Notwendigkeit der Erfassung von Messdaten

Menschen erwerben ihr Wissen durch die Beobachtung von Umwelt und Natur, durch Gedanken, Erfahrungen und Lernen. Diese Art und Weise der Erwerbung des menschlichen Wissens muss auch in bestimmtem Grade in komplexen technischen Systemen – insbesondere in großen Netzwerken – nachgebildet werden. Dadurch soll ihnen eine

bestimmte künstliche Intelligenz beigebracht werden, um sie auf diese Art zu befähigen, wichtige Prozesse im Laufe des Netzwerksbetriebs selbst- und eigenständig optimieren zu können. Damit man eine derartige Wunschvorstellung verwirklichen kann, müssen alle relevanten Messwerte im Laufe der Zeit zuerst erfasst, dann auf eine spezielle Art dokumentiert und archiviert werden.

Big Data

Demzufolge müssen die Daten über den Betrieb des Netzwerkes kontinuierlich erfasst werden. Hierdurch wird die Menge von Messdaten im Laufe der Zeit immer größer. Im Hinblick darauf spricht man von *Big Data*, also einer überproportional großen Datenmenge. Diese Daten müssen zuerst analysiert, bevor sie im nächsten Schritt zum maschinellen Lernen in Netzwerken benutzt werden. Eine derartige Analyse von Big Data als eine Art Vorbereitung von großen Daten zu deren Einsatz bei der Unterstützung von Lernprozessen wird als *Big Data Analytics* bezeichnet. Für Näheres darüber siehe [10.2.b], [10.2.c].

(Big-)Data-Driven Networking

Im Zusammenhang mit Big Data Analytics in Netzwerken spricht man von *Data-Driven Networking* (DDN) bzw. von *Big-Data-Driven Networking* (kurz BDDN oder bDDN). (B)DDN ist als ein sehr allgemeines Konzept, als eine Art 'zukünftiges Netzwork-Framework' zu verstehen, in dem die Antworten auf die folgenden vier Fragestellungen spezifiziert werden:

1. Welche Messdaten sollen über den Netzwerkbetrieb gesammelt, dokumentiert und archiviert werden? — Erfassung von Big Data
2. Wozu und wie sollen diese Messdaten verarbeitet werden? — BD Analytics
3. Wie sollen die Ergebnisse der Datenanalyse (Big Data Analytics) zur Befähigung des Netzwerks verwendet werden? — Maschinelles Lernen
4. Wie soll die durch maschinelles Lernen erworbene Intelligenz zur Optimierung bzw. Verbesserung des Systemverhaltens im laufenden Netzwerkbetrieb eingesetzt werden? — Optimierung/ Verbesserung des Systemverhaltens

(B)DDN an andere Arten von Networking

Im Trend zu (B)DDN wird hauptsächlich das Ziel verfolgt, intelligentes, autonomes und optimiertes Netzwerkmanagement sowie selbst adaptierte Sicherheit zu ermöglichen. (B)DDN kann dazu führen, dass man zwischen Netzwerken mit unterschiedlichen Intelligenzarten unterscheiden kann – und zwar zwischen Cognitive Networks, Intent-based Networking und Autonomic Networking. Für weitere Informationen über (B)DDN und über verschiedene intelligente Arten von Networking ist zu empfehlen:

10.2.a Haipeng Yao, Chao Qiu, Chao Fang, Xu Chen, F. Richard Yu: A Novel Framework of Data-Driven Networking; IEEE Access, Vol. 4, Nov 2016; DOI: 10.1109/ACCESS.2016.2624781

10.2.b Junchen Jiang, Vyas Sekar, Ion Stoica, Hui Zhang: Unleashing the potential of data-driven networking; Proceedings of 9th International Conference on COMmunication Systems & NETworkS (COMSNET), 2017.

10.2.c Sachin Katti: Data Driven Networking; Platform Lab Review, Feb 2017

10.2.d Junchen Jiang, Vyas Sekar, Ion Stoica, Hui Zhang: Data-Driven Networking: Harnessing the 'Unreasonable Effectiveness of Data' in Network Design; Carnegie Mellon University, Pittsburgh, Feb 2016

10.2.e ITU-T: Y.3650 - Framework of big-data-driven networking, Jan 2018

18.10.3 Cognitive Networks

Falls die Messdaten in einem Netzwerk zuerst über gewisse Zeit erfasst, gespeichert, analysiert (Big Data Analytics) und danach beim Maschinellen Lernen so verwendet werden, um sog. *kognitiven Fähigkeiten* des Menschen, d.h. Signale aus der Umwelt wahrzunehmen und weiterzuverarbeiten, dann besitzt das Netzwerk auch bestimmte kognitive Fähigkeiten und kann folglich als *kognitives Netzwerk* (*Cognitive Network*) bezeichnet werden.

Kognitive Fähigkeiten

Somit ist ein kognitives Netzwerk ein Netzwerk mit bestimmten kognitiven Fähigkeiten, das durch maschinelles Lernen befähigt wurde, die aktuelle Netzwerksituation wahrzunehmen, diese zu erfassen und zu analysieren, um sich immer an jede neue Netzwerksituation optimal anzupassen, will heißen, einige der wichtigen Konfigurationsparameter des Netzwerks optimal einzustellen. Ein kognitives Verhalten kann in Netzwerken vor allem dazu dienen, die QoS (Quality of Service) und Netzwerksicherheit zu verbessern. Insbesondere die Fähigkeit von Cognitive Networks, verschiedene Anomalien zu erkennen und darauf optimal zu reagieren, kann zur Erhöhung der Netzwerksicherheit beitragen. Für Näheres über Cognitive Networks siehe:

10.3.a Michele Zorzi, et al.: COBANETS: A new paradigm for cognitive communications systems; International Conference on Computing, Networking and Communications, Feb 2016; DOI: 10.1109/ICCNC.2016.7440625

10.3.b Michele Zorzi, Andrea Zanella, Alberto Testolin, Michele De Filippo De Grazia, Marco Zorzi: Cognition-Based Networks: A New Perspective on Network Optimization Using Learning and Distributed Intelligence; IEEE Access, Vol. 3, Aug 2015; DOI: 10.1109/ACCESS.2015.2471178

10.3.c Mario Bkassiny, Yang Li, Sudharman K. Jayaweera: A Survey on Machine-Learning Techniques in Cognitive Radios; IEEE Communications Surveys & Tutorials, Vol. 15(3), 2013; DOI: 10.1109/SURV.2012.100412.00017

10.3.d Marco Levorato: Cognitive Networking with Dynamic Traffic Classification and QoS Constraints; IEEE Wireless Communications and Networking Conference (WCNC); Mar 2017; DOI: 10.1109/WCNC.2017.7925717

10.3.e Ijaz Ahmad: Improving Software Defined Cognitive and Secure Networking; University of Oulu, 2018; ISBN: 978-952-62-1951-6

18.10.4 Intent-based Networking

Ziel von Intent-based (absichtsbasiertes)Networking

Eine besondere Art von von Data-Driven Networking wird als *Intent-based (absichtsbasiertes) Networking* bezeichnet. Man betrachtet absichtsbasiertes Networking als Grundstein für eine neue Ära von Networking. Dem Intent-based Networking liegt das folgende Ziel zugrunde: Ein Netzwerk soll in der Lage sein, sich – der Absicht seines Administrators entsprechend – selbst so zu konfigurieren, um die in Form von sog. Konfigurationsrichtlinien allgemein formulierten Absichten des Administrators in die entsprechenden Konfigurationsparameter umzusetzen.

Idee von Intent-based Networking

Die grundlegende Idee von Intent-based Networking ist, dass das Netzwerk eigenständig die geschäftlichen Absichten/Ziele berücksichtigen muss und diese selbst/automatisch in Konfigurationsparametern seiner Funktionskomponenten (Router, Switches, Firewalls, ...) abbilden soll. Als Folge dessen sollen manuelle Konfigurationseingriffe in Intent-based Networks auf ein Minimum reduziert werden. Darüber hinaus soll jedes Intent-based Network von vornherein ausgewählten Parametern, insbesondere dabei die sog. QoS-Metriken, kontinuierlich überwachen und im Bedarfsfall entsprechende Anpassungen von Konfigurationsparametern vorzunehmen, um sicherzustellen, dass alle ausgewählten Parameter im 'grünen Bereich' sind. Für weitere Informationen über Intent-based Networking ist zu empfehlen:

10.4.a David Lenrow: Intent-Based Networking Seeks Network Effect; Sep 2015

10.4.b Susan Hares: 2015 ONF/SDN Market Opportunities, Tutorial Intent Networking, Open Networking Foundation

10.4.c ONF TR-523: Intent NBI – Definition and Principles; Oct 2016

10.4.d Franco Callegati, Walter Cerroni, Chiara Contoli, Francesco Foresta: Performance of Intent-based Virtualized Network Infrastructure Management; IEEE International Conference on Communications (ICC), May 2017; DOI: 10.1109/ICC.2017.7997431

10.4.e Alexander Thiele: Future Ready Networking Solutions for the Datacenter; Dell EMC Forum, Frankfurt 24.10.2017

10.4.f Carlos Infante: Cisco SD-WAN: Intent-based networking for the branch and WAN; Mar 2018

18.10.5 Autonomic Networking

Seit Beginn der 2000er-Jahre wird der Begriff '*Autonomic Systems*' (*Autonome Systeme*) geprägt [Abschnitt 10.5.1]. Diese Idee von *Autonomic Systems* wird in den letzten 10 Jahren auch auf dem Networking Gebiet übernommen, sodass man von *Autonomic Networking* bzw. *Autonomic Networks* spricht. In RFC 7575 wird 'Autonomic Networking' definiert und dessen Ziele näher dargestellt. Das wesentliche Ziel eines autonomen Netzwerks ist dessen Selbstmanagement, einschließlich Selbstkonfiguration, Selbstoptimierung, Selbstheilung und Selbstschutz. Autonomic Networking kann auch als eine besondere Variante von *Data-Driven Networking* (DDN) betrachtet werden. Daher spielen (Big) Data Analytics und Maschinelles Lernen auch bei Autonomic Networking eine fundamentale Rolle. Um Autonomic Networking zu verwirklichen, werden hierfür verschiedene Architekturmodelle und Protokolle entwickelt. Diese Aktivitäten werden von der Working Group `amina` (*Autonomic Networking Integrated Model and Approach*) bei der IETF koordiniert. Für weitere Informationen siehe:

10.5.a Jeffrey O. Kephart, David M. Chess: The Vision of Autonomic Computing; IEEE Computer, Vol. 36(1), Jan 2003; DOI: 10.1109/MC.2003.1160055

10.5.b EU FP6 IST Project 27489: ANA Project; Autonomic Network Architecture; Feb 2007

10.5.c Zeinab Movahedi, Mouna Ayari, Rami Langar, Guy Pujolle: A Survey of Autonomic Network Architectures and Evaluation Criteria, IEEE Communications Surveys & Tutorials, Vol. 14(2), May 2012; DOI 10.1109/SURV.2011.042711.00078

10.5.d ETSI GS AFI 002 V1.1.1: Autonomic network engineering for the self-managing Future Internet (AFI); Generic Autonomic Network Architecture (An Architectural Reference Model for Autonomic Networking, Cognitive Networking and Self-Management) Disclaimer; Apr 2013

10.5.e CISCO: Autonomic Networking Configuration and Deployment Guide; Dec 2016

10.5.f Zhongliang Zhao, Eryk Schiller, Eirini Kalogeiton, Torsten Braun, Burkhard Stiller, Mevlut Turker Garip, Joshua Joy, Mario Gerla, Nabeel Akhtar, Ibrahim Matta: Autonomic Communications in Software-Driven Networks; IEEE Journal on Selected Areas in Communications, Vol. 35(11), Nov 2017; DOI: 10.1109/JSAC.2017.2760354

10.5.g RFC 7575: Autonomic Networking: Definitions and Design Goals; Jun 2015

10.5.h RFC 8368: Using an Autonomic Control Plane for Stable Connectivity of Network Operations, Administration, and Maintenance (OAM); May 2018

10.5.i draft-ietf-anima-autonomic-control-plane-18: An Autonomic Control Plane (ACP); Jul 2018

10.5.j draft-ietf-anima-reference-model-10: A Reference Model for Autonomic Networking; Nov 2018

18.10.6 AI, IoT and 5G Convergence

Die 5. Generation (5G) der Mobilfunknetze wird eine fundamentale Bedeutung bei der Weiterentwicklung des IoT haben. In Bezug darauf kann man bereits heute von einer Konvergenz der 5G-Mobilfunknetze mit dem IoT sprechen. Die Ideen für die Unterstützung dieser Konvergenz werden intensiv entwickelt. Bei diesen Entwicklungen spielen die Konzepte für (Big) Data Analytics und Maschinelles Lernen eine sehr wichtige Rolle. Für weitere Informationen darüber siehe:

10.6.a Ying Wang, et al.: A Data-Driven Architecture for Personalized QoE Management in 5G Wireless Networks; IEEE Wireless Communications, Vol. 24(1), Feb 2017; DOI: 10.1109/MWC.2016.1500184WC

10.6.b Chunxiao Jiang, et al.: Machine Learning Paradigms for Next-Generation Wireless Networks; IEEE Wireless Communications, Vol. 24(2), Apr 2017; DOI: 10.1109/MWC.2016.1500356WC

10.6.c Zhiyuan Xu, Yanzhi Wang, Jian Tang, Jing Wang, Mustafa Cenk Gursoy: A deep reinforcement learning based framework for power-efficient resource allocation in cloud RANs; IEEE International Conference on Communications (ICC), May 2017; DOI: 10.1109/ICC.2017.7997286

10.6.d Paulo Valente Klaine, et al. A Survey of Machine Learning Techniques Applied to Self-Organizing Cellular Networks; IEEE Communications Surveys and Tutorials, Vol. 19(4), Jul 2017; DOI: 10.1109/COMST.2017.2727878

10.6.e Rongpeng Li, et al.: Intelligent 5G: When Cellular Networks Meet Artificial Intelligence; IEEE Wireless Communications, Vol. 24(5), Oct 2017; DOI: 10.1109/MWC.2017.1600304WC

10.6.f Magnus Malmström: 5G Positioning using Machine Learning; Master of Science Thesis in Applied Mathematics, Department of Electrical Engineering, Linköping University, 2018

10.6.g Yang Yang, et al.: DECCO: Deep-Learning Enabled Coverage and Capacity Optimization for Massive MIMO Systems; IEEE Access, Apr 2018; DOI: 10.1109/ACCESS.2018.2828859

10.6.h Mowei Wang, et al.: Machine learning for networking: Workflow, advances and opportunities; IEEE Network; Vol: 32(2), Mar - Apr 2018; DOI: 10.1109/MNET.2017.1700200

10.6.i Zhifeng Zhao, et al.: Deep Reinforcement Learning for Network Slicing; arXiv:1805.06591v2, May 2018

10.6.j Chaoyun Zhang, Paul Patras, Hamed Haddadi: Deep Learning in Mobile and Wireless Networking: A Survey; arXiv:1803.04311v2, Sep 2018

10.6.k Honggang Zhang: Intelligent 5G/6G: When Wireless Networks Meet Artificial Intelligence (5G+AI=6G)

18.11 Abschließende Bemerkungen

Wissen wird von Menschen durch die Beobachtung von Umwelt und Natur, Gedanken, Erinnerungen, Erfahrungen und Lernen erworben. Die zu diesen Fähigkeiten führenden informationsverarbeitenden Prozesse im menschlichen Gehirn werden als *menschliche Kognition* (Human Congition) bezeichnet. Um technische Systeme zu befähigen, ebenso intelligent wie Menschen zu sein und sich folglich an neue Situationen möglich optimal anzupassen, muss in diesen die menschliche Kognition zu einem bestimmten Grade nachgebildet werden. Demzufolge gilt als wichtiger Trend bei technischer Entwicklung die Erarbeitung von Ideen zur Einrichtung von auf den Prinzipien menschlicher Kognition basierenden intelligenten Systemen. Es wird also versucht, die menschliche Kognition, d.h. kontextbewusstes Denken, in technischen Systemen nachzubilden. Man spricht in diesem Zusammenhang von intelligenten bzw. kognitiven Systemen, die als *kontextbewusste Systeme* (Context-Aware Systems) angesehen werden können.

18.11.1 Vom IoT zum Intelligent Iot

Intelligent IoT

Von enorm großer Bedeutung ist die Nachbildung der menschlichen Kognition im heutigen Internet of Things (IoT). Bestrebungen in diese Richtung führen zur Entstehung von Intelligent IoT (IIoT). Dies bedeutet, dass Komponenten/Systeme im zukünftigen

IIoT eine *künstliche Intelligenz* (KI, *Artificial Intelligence*, AI) besitzen werden. Infolge werden sie in der Lage sein, situationsbewusst zu handeln und somit, wichtige Prozesse im Laufe der Zeit eigenständig zu optimieren und sich an eine neue Situation möglichst optimal anzupassen. Bevor diese Idee im heutigen IoT verwirklicht werden kann, müssen relevante Messwerte und Ereignisse kontinuierlich erfasst und dann auf eine spezielle Art für weitere Analysen bzw. zu Lernzwecken dokumentiert und archiviert werden.

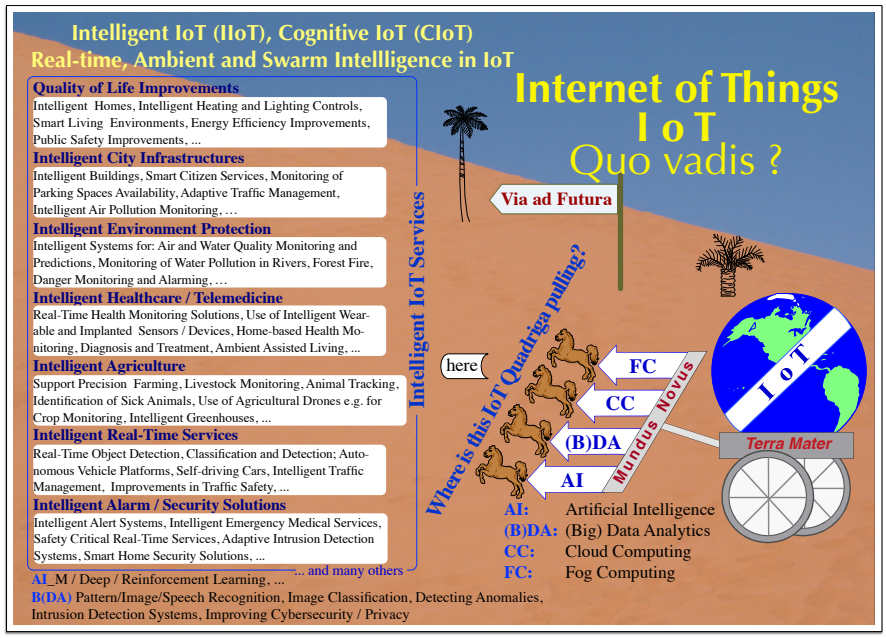


Abb. 18.11-1: Internet: Quo vadis?

Big IoT Data

Durch das kontinuierliche Erfassen verschiedener Arten von Daten im IoT wird ihre Menge jedoch immer größer. Man spricht von *Big Data* bzw. *Big IoT Data*. Bevor sie beim maschinellen Lernen zum Einsatz kommen, müssen diese Daten entsprechend analysiert werden. Die Analyse von Big Data als eine Art 'Vorbereitung auf ihren Einsatz', u.a. zur Unterstützung von Lernprozessen im IoT, wird als *Big Data Analytics* bezeichnet. Das wesentliche Ziel der Analysen von Big Data ist die Entdeckung von versteckten Mustern, unbekannten Korrelationen und anderen nützlichen Informationen in den gesammelten 'riesigen Datenmengen'. Die Ergebnisse dieser Analysen im IoT werden einerseits Menschen angezeigt, damit sie die gewonnenen Informationen nutzen können. Andererseits sollen verschiedene IoT-Komponenten auf der Basis dieser Informationen selbständig lernen und sich dadurch in die Lage versetzen, auf verschiedene Situationen und plötzlich auftretende Ereignisse im IoT fast so intelligent wie Menschen zu reagieren. Sobald das IoT die eben geschilderten Fähigkeiten besitzt, immer kontextbewusst (situationsbewusst) agieren zu können, wird man von IIoT sprechen.

Cognitive IoT

Die Evolution des IoT zum IIoT schreitet unaufhörlich voran und führt zu solchen Erweiterungen des IoT, um den Verlauf der menschlichen Kognition im IoT nachbilden zu können. Die kognitiven Fähigkeiten des Menschen ermöglichen ihm, verschiedene Arten von Informationen aus der Umwelt wahrzunehmen und weiterzuverarbeiten. Da die Fähigkeiten von IIoT im gewissen Grade den kognitiven Fähigkeiten des Menschen entsprechen werden, wird es als Cognitive IoT (CIoT) bezeichnet. Es sei hervorgehoben, dass AI, (Big) Data Analytics, Cloud Computing und Fog Computing als die vier wichtigsten treibenden Kräfte, quasi als 'Zugpferde' einer IoT Quadriga, bei der Weiterentwicklung des IoT zum IIoT angesehen werden können. Abb. 18.11-1 bringt dies illustrativ zum Ausdruck.

Intelligent Internet of Threats

Die Garantie der Sicherheit im IoT ist heutzutage ein enorm breites und wichtiges Thema. Man kann sich heute noch kaum vorstellen, mit welchen bösartigen Angriffen und anderen Unsicherheiten im zukünftigen IIoT gerechnet werden muss. Es bleibt nur zu hoffen, dass statt 'Intelligent Internet of Things (IIoT)' nicht gesagt werden muss: 'Intelligent Internet of Threats (IIoT)'.

18.11.2 Rückblick auf 50 Jahre Rechnerkommunikation

Nachdem die Networking Trends, de facto Internet Trends, und dabei auch die Migration von IoT zum IIoT, also zukünftige Internet-Aspekte, kurz erläutert wurden, stellt sich die Frage: Welche relevanten, technischen sowie organisatorischen Entscheidungen und Entwicklungen gab es auf dem Weg zum heutigen Internet? Eine Antwort darauf soll nun ein kurzer Rückblick auf 50 Jahre Rechnerkommunikation geben.

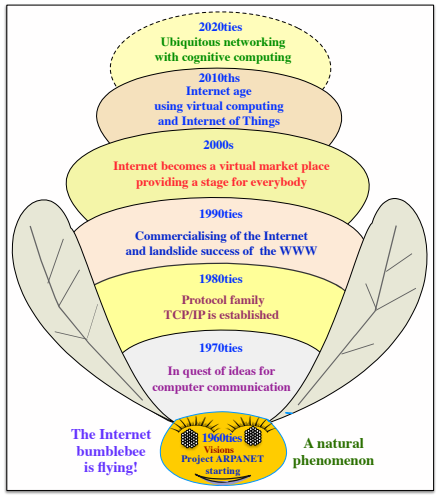


Abb. 18.11-2: Die Internet-Hummel fliegt – ein Naturphänomen

Rückblick auf 50 Jahre der Rechnerkommunikation

Zur Entstehung des Internet in der 'heutigen Form' haben zahlreiche, aus den letzten fünf Jahrzehnten stammende Entwicklungen auf dem Gebiet der Rechnerkommunikation geführt. Heutzutage stellt das Internet – organisatorisch und technisch gesehen – ein immens komplexes, über die ganze Welt verteiltes technisches Gebilde dar. Wegen seiner Komplexität folgt es dem Hummel-Phänomen und funktioniert. Daher soll die

in Abb. 18.11-2 gezeigte Internet-Hummel die fünf Jahrzehnte der Rechnerkommunikation auf dem Weg zum heutigen Internet anschaulich zum Ausdruck bringen.

Um die bedeutenden, zur Entstehung des Internet führenden Ereignisse zu nennen, müssen wir nicht nur die letzten fünf Jahrzehnte nach dem historischen Datum 29. Oktober 1969 betrachten, sondern auch die 60er Jahre. Besonders in den 1960er, 1970er, 1980er und 1990er Jahren gab es eine Vielzahl technischer Ereignisse und Entwicklungen, die als Meilensteine auf dem Weg zum heutigen Internet angesehen werden sollen [Abb.18.11-3]. Im Hinblick hierauf lassen sich die einzelnen Jahrzehnte kurz wie folgt charakterisieren:

Die 1960er Jahre

Das erste experimentelle Rechnernetz namens ARPA(NET) entsteht und die Ära der Rechnerkommunikation beginnt.

1964
- Das Konzept für *Packet Switching Networks* wird von Paul Baran veröffentlicht.

1965
- Ted Nelson prägt die Begriffe *Hypertext* und *Hypermedia* – also jene, die man heute mit dem Begriff WWW verbindet.

1966
- Die Entwicklung des Rechnernetzes namens ARPANET beginnt.

1969
- Im März 1969 ist die Idee entstanden, die ARPANET-betreffenden Entwicklungen als sog. *Request for Comments* (RFC) zu dokumentieren.

Ur-Internet
- Am 29. Oktober 1969 wird das ARPANET mit vier IMPs (*Interface Message Processor*) als Knoten in Betrieb genommen. Somit war das erste Rechnernetz mit Paketvermittlung geboren.
  Die Daten zwischen Hosts werden nach den in RFC 11 (August 1969) spezifizierten *Host-Host Software Procedures* ausgetauscht. Die IMPs, als Vorfahren von heutigen Routern im Internet, wurden damals untereinander über Telefonleitungen verbunden und dienten als Netzknoten mit Paketvermittlung.

Die 1970er Jahre

Die Suche nach Ideen/Konzepten für die Rechnerkommunikation.

1970
- ALOHAnet – das erste Funkrechnernetz – wird in Betrieb genommen.
- Das *Host-to-Host-Protocol* für das ARPANET wird als RFC 33 spezifiziert und im ARPANET vom Januar 1972 bis zur Umstellung auf TCP/IP (1. Januar 1983) verwendet.

1972
- Entwicklung des ersten E-Mail-Programms und Erfindung des `@`-Zeichens durch R. Tomlinson.
- Die Gründung von NIC (*Network Information Center*) u.a. zur Dokumentation und Registrierung von Netzparametern, der Vorläufer von IANA (*Internet Assigned Numbers Authorithy*).

1974
- ITCP und NCP werden im RFC 675 spezifiziert [Abb. 18.11-3].

Internet
- In diesem RFC wird das Wort *Internet* zum ersten Mal offiziell verwendet.

1976
- CCITT-Standard X.25 für *Packet Switching*: Verbindungsorientierte Übermittlung der Datenpakete – die Grundlage für das spätere MPLS.
- Veröffentlichung des Konzepts von *Ethernet*, Robert M. Metcalfe und David R. Boggs.

- Whitfield Diffie und Martin Hellman publizieren die Idee zur Realisierung eines *Schüsseltauschs mittels kryptographischer Funktionen* und somit der Verständigung auf einen gemeinsamem geheimen Schlüssel. DHE
- Das *OSI-Referenzmodell* entsteht und wird später in der CCITT X.200 Empfehlung spezifiziert. 1977
- Das *RSA-Verfahren* (Rivest, Shamir und Adelman) als Grundlage für den Aufbau asymmetrischer Kryptosysteme mit öffentlichen Schlüsseln (*public keys*) wird zum Patent angemeldet.
- Das *Network Control Protocol* (NCP) und das *Internet Transmission Control Program* (ITCP) werden durch IP und TCP (*Transmission Control Protocol*) funktionell ersetzt [Abb. 18.11-3]. Dies ist die Geburtsstunde der Protokolle IP und TCP; eine verbindungslose 'Alternative' zum TCP ist ebenso vorgesehen. 1978
- Das Gremium *Internet Configuration Control Board* (ICCB) wird gegründet, um die Entwicklung des Internets zu koordinieren. 1979

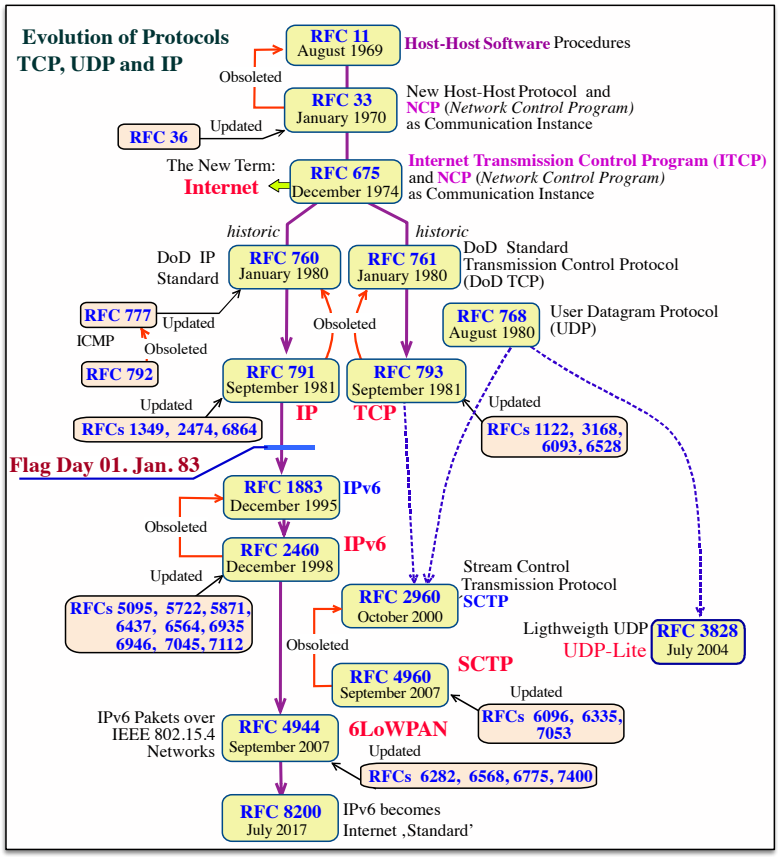


Abb. 18.11-3: Entwicklung der Internet-Standards im Spiegel der RFC

Die 1980er Jahre Die Etablierung des Internets mit der TCP/IP-Protokollfamilie.

1981
- Der Aufbau von CSNET (*Computer Science Network*) wird initiiert und dieses dient u.a. als Vorläufer von NSFNET (*National Science Foundation Network*).
- Der Plan für die ARPANET-Umstellung von ITCP/NCP auf TCP/IP wird entwickelt.

1983
- Am 1. Januar 1983 (sog. Flag Day) wird das ARPANET auf TCP/IP umgestellt und damit hat die Ära von TCP/IP begonnen.
- Aus dem ARPANET entsteht das MILNET (*Military Network*) und der restliche Teil bildet das wissenschaftliche, zivile ARPANET.
- Der Aufbau von DFN (*Deutsches Forschungsnetz*) beginnt und damit auch die Einführung des Internets in Deutschland.

1984
- Die Kopplung des DFN mit dem CSNET führt dazu, dass die erste, über das Internet transportierte E-Mail in Deutschland an der Universität Karlsruhe am 3. August 1984 empfangen werden kann [Zor14].

1986
- Das NSFNET in einer modernen hierarchischen Struktur (*Backbone-Konzept*) wird etabliert (eine Art Internet-Backbone) und die große Akzeptanz von NSFNET führt zur Stilllegung des ARPANET.

TLD `.de`
- Am 5. November 1986 wird die Top-Level-Domain `.de` für Deutschland im DNS eingetragen.

Die 1990er Jahre Die Kommerzialisierung des Internets und der Siegeszug des WWW.

1990
- Am 28. Februar 1990 wird das alte, ausgediente ARPANET offiziell stillgelegt. Seine Rolle übernimmt das moderne, in den 80er Jahren eingerichtete, Backbone-basierte NSFNET.

1991
- Das NSFNET wird für kommerzielle Nutzung als eine Art Internet-Backbone freigegeben. Die Folge ist die Kommerzialisierung des Internets und der Internet-basierten Technologien.

1995
- Der Backbone-Teil im NSFNET wird auf die neue Struktur mit NAPs (*Network Access Point*) umgestellt. Diese sind Vorläufer moderner IXPs (*Internet eXchange Point*) und führen zur Entstehung von ISPs (*Internet Service Provide*r).

Die 2000er Jahre Das Internet als Information Delivery Network, als virtueller Markt und der Beginn von Internet of Things.

Die 2010er Jahre Das Internet im Zeitalter von Virtual Computing und Internet of Things.

Für eine Vertiefung der chronologischen Entwicklung des Internets sei verwiesen auf:

- Meilensteine in der Entwicklung des Internets
- Evolution of the Internet – Significant Technical Events

Dank an Alle! Das Internet ist eine der größten technischen Schöpfungen des 20ten Jahrhunderts mit der Beteiligung einer enormen Vielzahl der Menschen auf der ganzen Welt. Dafür geht unseren Dank an alle, die mit ihrem organisatorischen und technischen Engagement, Wissen und Können zur Entstehung des Internets beigetragen haben.

Abbildungsverzeichnis

Stichwortverzeichnis

TECHNIK DER IP-NETZE //

- **Umfassende Informationen – Das lückenlose Standardwerk zu den Prinzipien der Kommunikation im Internet mit über 700 Bildern.**
- **Ausgewogenheit von Theorie und Praxis – Technische Aspekte der IP-Netze werden detailliert und zugleich praxisorientiert dargestellt.**
- **Aktuell und zukunftsweisend – Die präsentierten Themen vertiefen die aktuellen Trends der Internettechnik und -sicherheit sowie zukünftige Anforderungen.**

In IP-Netzen laufen komplexe Vorgänge bei der Übermittlung von Daten in Form von IP-Paketen ab. Das massive Internet-Wachstum und dabei entstandene Anforderungen haben zu zahlreichen Entwicklungen geführt – hervorzuheben sind u. a. das Internetprotokoll IPv6, die Techniken MPLS und GMPLS, mehrere Arten von Virtual Networks, Distributed Layer 2/3 Switching und »Internet of Things«.

Dieses Buch enthält eine systematische Darstellung der TCP/IP-Protokollfamilie, von Routing-Prinzipien in klassischen IP- wie auch in IPv6-Netzen. Es erläutert außerdem die Konzepte zum Aufbau von IP-Netzen auf der Basis unterschiedlicher Netztechnologien speziell im Hinblick auf Virtual Networks sowie der Unterstützung der Mobilität. Zudem enthält diese Auflage eine umfangreiche Darstellung der technologischen Grundlagen des »Internet of Things«.

Das Buch eignet sich nicht nur als Lehrbuch für Studierende unterschiedlicher Fachrichtungen sowie für Neueinsteiger, sondern auch als Nachschlagewerk für den Praktiker. Im Buch sind die relevanten Quellen ins Internet verlinkt, sodass es sich auch als »Informations-Hub« für das Selbststudium einsetzen lässt.

Prof. Dr.-Ing. Anatol **BADACH**
ist ehemaliger Professor im FB Angewandte Informatik der Hochschule Fulda. Seine Schwerpunkte sind Netzwerktechnologien und -protokolle, VoIP, Next Generation Networking sowie Internet of Things.

Prof. Dr. Erwin **HOFFMANN**
ist Vertretungs-Professor für Informatik an der Frankfurt University of Applied Sciences. Schwerpunkte sind Netzwerke, verteilte Systeme sowie IT-Security und ihre Implementierung.

AUS DEM INHALT //

- **Grundlagen der IP-Netze, Security; Protokolle: IPv4, ICMP, IGMP, TCP, UDP und SCTP**
- **Protokolle: DNS, DHCP, NAT, IPsec, TLS (1.3) und Multipath TCP**
- **Protokolle für Echtzeitkommunikation: RTP, RTCP, SIP und SDP**
- **IPv6 und Support-Protokolle: ICMPv6, NDP und DHCPv6**
- **Migration zu IPv6: 6to4, 6rd, ISATAP, Teredo, NAT64 und Translation IPv4&IPv6**
- **Routing-Protokolle: RIP, OSPF und BGP-4; Multicast-Routing nach PIM und MSDP**
- **IP-Netze mit MPLS, GMPLS und Traffic Engineering**
- **VPNs, Mobility Support und Distributed Layer 2/3 Switching**
- **Internet of Things mit 6loWPAN, RPL und CoAP**

HANSER

€ 42,99 [D] | € 42,99 [A]
ISBN 978-3-446-45511-5

9 783446 455115

www.hanser-fachbuch.de/computer