Kommunikation zwischen Rechnern

Rechnernetze

Ein ***Rechnernetz*** ist ein räumlich verteiltes System von Rechnern, die miteinander verbunden sind. Dadurch wird ein Austausch von Daten (Informationen, Dokumenten, E-Mails, …) möglich.

Topologie (Rechnernetz)

Der Begriff ***Topologie*** beschreibt die Struktur bzw. den Aufbau eines Netzwerks. Man unterscheidet verschiedene Topologien:

→ Siehe Präsentation Topologie

IP- und MAC-Adressen, Routing

Damit sich verschiedene Rechner in einem Rechner ansprechen können sind Adress-Schemata nötig. Hierfür gibt es IP- und MAC-Adressen.

IP-Adressen

Um ein Computer-Netzwerk zu definieren bzw. weltweit sehr viele solcher Netzwerke unterscheiden zu können braucht man ***IP-Adressen***. Diese haben die Form ***91.66.185.199*** . Es handelt sich um vier durch Punkte getrennte Bytes (je 8 bit = 32 bit) in Dezimaldarstellung. Jedes Byte kann also Werte von 0 bis 255 annehmen ($2^8 = 256$). Den Werten 0 und 255 können dabei Sonderrollen zukommen (Die „erste“ IP-Adresse (0) bezeichnet das Netz selbst und die „letzte“ (255) ist für den Broadcast (engl. „Rundruf“) vorbehalten.).

Konventionen zur Vergabe von IP-Adressen im Netzwerk: Die ***Router*** bekommen die ***ersten freien*** (nicht die Netz- und Broadcast-) ***Adressen*** im Netzwerk. Die ***Server*** belegen die ***letzten freien Adressen***. Die „normalen“ Rechner bekommen beliebige freie IP-Adressen in der Mitte.

Zusätzlich zur IP-Adresse benötigt man heutzutage auch noch eine ***Subnetzmaske***, die ebenfalls aus vier durch Punkte getrennte Bytes im Dezimalsystem dargestellt werden. Allerdings sind hier „im Hausgebrauch“ nur die Werte 0 und 255 üblich. Subnetzmasken („im Hausgebrauch“) beginnen immer mit 255 und enden immer mit 0. Übliche werte für die Subnetzmaske sind also ***255.255.255.0*** (wird auch mit ***/24*** nach der IP-Adresse angegeben, ehemals Klasse C), ***255.255.0.0*** (***/16***, ehemals Klasse B) und ***255.0.0.0*** (***/8***,

ehemals Klasse A). Damit teilt die Subnetzmaske die IP-Adresse in einen ***Netzanteil*** (255) und einen ***Host-Anteil*** (0). Die verschiedenen Subnetze unterscheiden sich in der Menge der an das Netz angeschlossenen Endgeräte (Hosts). An ein Netz mit Subnetzmaske /24 können z. B. nur 254 (x.x.x.1 bis x.x.x.254, da die 0 und 255 schon vergeben sind) Endgeräte anschlossen sein.

91 . 66 . 185 . 199
255 . 255 . 0 . 0

Netzanteil: 91.66 Hostanteil: 185.199

Alle Rechner mit demselben Netzanteil befinden sich in demselben Netz (***Adresse des Netzwerks: 91.66.0.0***) und können direkt miteinander (über einen Switch – und damit über die ***MAC-Adressen***) kommunizieren.

MAC-Adressen

Jede Netzwerk-Karte verfügt vom Hersteller über eine (eigentlich weltweit eindeutige) ***MAC-Adresse***. Diese hat die Form ***90:e6:ba:c1:76:fe***. Es handelt sich um 6 mal 2 Hexadezimal-Zeichen (0, 1, 2, … , 9, a, b, … , f) bzw. um 6 Bytes. In einem lokalen Netz wird ausschließlich über diese MAC-Adressen kommuniziert. Die MAC-Adressen der Rechner sind es auch, die sich ein Switch „merkt“, um Datensendungen nur an den entsprechenden Rechner und nicht an alle anderen zu schicken.

Routing

Wenn Rechner aus verschiedenen Netzwerken miteinander kommunizieren wollen, dann müssen diese Netzwerke über ***Router*** miteinander verbunden sein und die Router müssen Informationen bereit halten, wo man welches Netzwerk findet und über welchen Weg die Daten dorthin zu versenden sind. Genauso müssen auch Rechner in einem Netzwerk wissen, wie sie aus dem lokalen Netzwerk hinaus (***Gateway*** („Tor nach außen“) über den Router) kommunizieren können.

Rechner in privaten Netzen (z.B. bei jedem von uns daheim) sind über das Internet in der Regel nicht erreichbar, da sie sich hinter einem ***NAT-Router*** (Network Address Translation Router) befinden, der ein privates Netzwerk (mit „privaten“ IP-Adressen) aufspannt. Des Weiteren verhindert dieser NAT-Router ungewollte Kommunikation von außen auf interne Rechner (***Firewall***). Ein NAT-Router lässt allerdings Antworten von außen stehenden Rechnern auf Anfragen von internen Rechnern durch, sonst könnte man ja z. B. nicht im Internet surfen. Zudem stellt dieser Router uns WLAN bereit.

Ein weiterer besonderer Router ist der ***ISP-Router*** (Internet Service Provider). Sie stellen uns eine Internetverbindung gegen Bezahlung zur Verfügung z. B. Telekom, Vodafone, M-net usw.

Kommunikation zwischen Rechnern

In der heutigen Zeit kommunizieren Rechner nicht einfach irgendwie (unkontrolliert) miteinander. Kommunikation findet nach dem ***Client-Server-Prinzip*** statt. Das bedeutet, dass ein Rechner im Netzwerk einen Dienst zur Verfügung stellt und die anderen Rechner greifen darauf zu. Beispiele:

- Ein ***Webserver*** stellt Webseiten zur Verfügung. Andere Rechner sehen sich die Seiten mit Web-Browsern an.
- Ein ***Dateiserver*** stellt Speicherplatz zur Verfügung. Andere Rechner können Dateien darauf speichern oder von dort herunter laden.
- Ein ***Datenbankserver*** stellt Datensätze zur Verfügung. Andere Rechner können darauf zugreifen, die Daten abfragen, sie verändern, löschen oder neue Daten einfügen.
- Ein ***E-Mail-Server*** stellt die Möglichkeit des Versendens von Nachrichten bzw. das Abrufen von empfangenen Nachrichten zur Verfügung.
- Ein ***DHCP-Server*** teilt anderen Rechnern eindeutige IP-Adressen und Informationen bzgl. des Internet-Zugangs zu. Die anderen Rechner im selben Teilnetz bitten den dhcp-Server beim Hochfahren um eine IP-Adresse.
- Ein ***DNS-Server*** (Domain Name System) übersetzt Domain-Namen in IP-Adressen (z.B. www.joachimhofmann.org in 188.138.127.119) und umgekehrt. Andere Rechner im Internet bitten ihn um eine entsprechende Übersetzung, wann immer sie im Internet surfen.

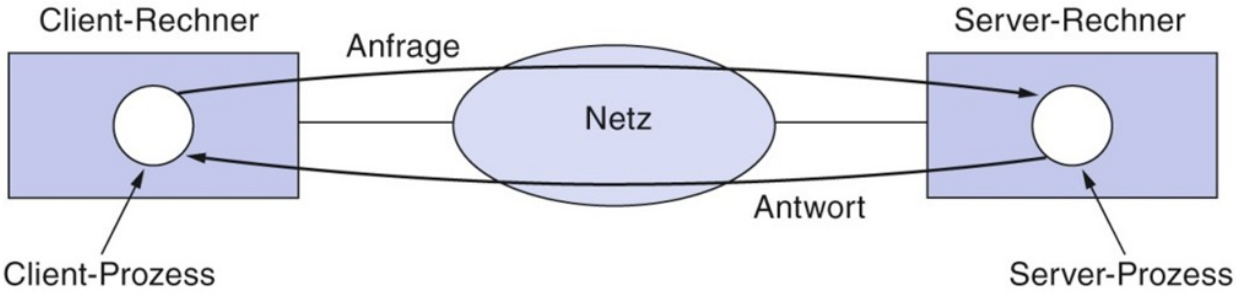


Abbildung 1.2: Das Client-Server-Modell enthält Anforderungen und Antworten.

Aufgabe 1:

Bearbeite das Arbeitsskript „Rechnernetze mit Filius“.

Ports

Ein Server muss nicht immer ein eigenständigen Rechner sein. Es ist aber auch möglich, dass ein ***ganzer Rechnerverbund (Serverfarm) einen Dienst anbietet*** (z. B. Google, Facebook usw.). Andererseits ist aber auch möglich, dass ***mehrere Server-Dienste auf einem einzelnen Rechner laufen***. So ist z. B. der Server in unserem Computerraum gleichzeitig DHCP-Server und Datei-Server.

Wie kann aber nun dieser Rechner unterscheiden, was ein Client von ihm will?

Hierfür hat man sog. ***Ports*** eingeführt. Das sind Nummern zwischen ***0 und 65535***, die man sich vorstellen kann wie die Zimmernummern in einer großen Behörde. Im Rathaus gibt es z. B. auf Zimmer 115 KFZ-Zulassungen, auf Zimmer 116 Führerscheine, auf Zimmer 124 – 126 Ausweis-Dokumente… . Allerdings können wir für unsere Zwecke nicht alle Port-Nummer nutzen, weil viele Port-Nummer schon von anderen Diensten belegt sind (Liste der standardisierten/well-known Ports). Daher benutzen unsere Programme nur Port-Nummern, die größer als 1024 sind.

Protokolle

Unter einem ***Protokoll*** versteht man in der Informatik einen Satz von Regeln, welche die Kommunikation eines Server-Dienstes exakt regeln. So gibt es z.B. folgende gut bekannten Protokolle und diese bekannten Dienste liegen auf wohl bekannten (wellknown) Ports:

- ***http*** (hypertext transfer protocol) regelt die Kommunikation zwischen Webserver und Web-Client und benutzt Port 80
- ***ftp*** (file transfer protocol) regelt die Kommunikation zwischen einem Datei-Server und Datei-Client und benutzt Port 20 (data transfer) und 21 (command)
- Da es viele verschiedene Datenbankprodukte gibt, gibt es leider nicht DAS Datenbankprotokoll
- ***smtp*** (simple mail transfer protocol), ***imap*** (internet message access protocol) und ***pop*** (post office protocol) regeln die Kommunikation zwischen E-Mail-Server und E-Mail-Client. (***smtp*** nutzt Port 25, ***imap*** nutzt Port 143, ***pop*** nutzt Port 110)
- ***dhcp*** (dynamic host configuration protocol) regelt die Kommunikation zwischen dhcp-Server und den dhcp-Clients und nutzt Port 68
- ***dns*** (domain name system) regelt die Kommunikation zwischen Name-Server und Name-Client und nutzt Port 53

Ports aus den nicht reservierten Bereichen stehen dann z. B. den Clients zur Verfügung um auf Antworten der Server zu lauschen. Diese Port-Nummern können dann bei jeder einzelnen Verbindung wechseln und sind nicht lange gebunden.

Port-Weiterleitung

Sitzt ein Server hinter einer Firewall oder einem NAT-Router, so ist er zunächst durch dieses Gerät geschützt, also nicht über das Netzwerk erreichbar. Möchte man dennoch seinen eigenen Rechner als Server (z.B. Spiele-Server bei Minecraft hosten) nutzen, sodass Rechner aus anderen Netzwerken auf diesen Server zugreifen können, so muss man eine sog. Port-Weiterleitung schalten. Das bedeutet, dass man auf dem Router eine Regel definiert, die in etwa allgemein so formuliert werden kann:

Leite Anfragen, die an die IP-Adresse des Routers und an einen bestimmten Port gerichtet sind an eine andere IP und ggf. einen anderen Port im internen Netzwerk weiter.

Nun kann man eine Anfrage, die eigentlich an einen Server hinter dem Router gerichtet ist, einfach direkt an den Router schicken. Er wird sie passend „zustellen“.

Aufgabe 2:

Stelle eine Verbindung zu einem Mailserver her und wende das SMTP-Protokoll an.

https://de.wikipedia.org/wiki/Simple_Mail_Transfer_Protocol

Datenübermittlung mittels der Transportprotokollen TCP und UDP

Bei Datenübertragungen über ein Netzwerk unterscheidet man prinzipiell zwischen ***TCP*** (***transmission control protocol***) und ***UDP*** (***user datagram protocol***).

- ***TCP*** ist eine verlustsichere Datenübertragung. Sicher in dem Sinne, dass für jedes abgeschickte Datenpaket von der Gegenstelle eine Empfangsbestätigung verlangt wird. Erfolgt diese nicht in einer bestimmten Zeit, so wird das entsprechende Paket erneut gesendet. TCP verwendet man immer dann, wenn eine unvollständige Datensendung keinen Sinn macht, z. B. also bei Webseiten (http), Datenbank-Anfragen, E-Mails (smtp, imap, pop), ...
- ***UDP*** ist eine verlustunsichere Verbindung. Unsicher in dem Sinne, dass die Datenpakete einfach versendet werden ohne sich Gedanken darüber zu machen, ob sie auch bei der Gegenstelle ankommen. UDP verwendet man häufig bei zeitkritischen Übertragungen, bei denen man auch mal auf einen kleinen Teil der Information verzichten kann, z. B. beim Streamen von Audio- oder Video-Dateien. Dort fällt es am Client nicht auf, wenn einmal eins von 24 Bildern pro Sekunde des Films nicht ankommt. Ein Nachreichen eines im Netz verunglückten Bildes gäbe hier wenig Sinn.

Schichtenmodelle

Bisher musste man aber nie genauer definieren, ob jetzt TCP oder UDP verwendet werden soll. Das liegt daran, dass der Aufbau der Rechnernetze in ***einzelnen übereinanderliegenden Schichten bzw. Ebenen*** unterteilt ist, um die Komplexität zu verringern. Die Funktion und Anzahl der Schichten können variieren, aber das Prinzip ist dabei immer gleich. Es sollen Dienste an höhere Schichten bereitgestellt werden. Dabei wird die jeweilige Implementierung der jeweiligen Schicht verborgen, um die Komplexität zu verringern. Der Austausch erfolgt über festgelegte Protokolle und Informationen.

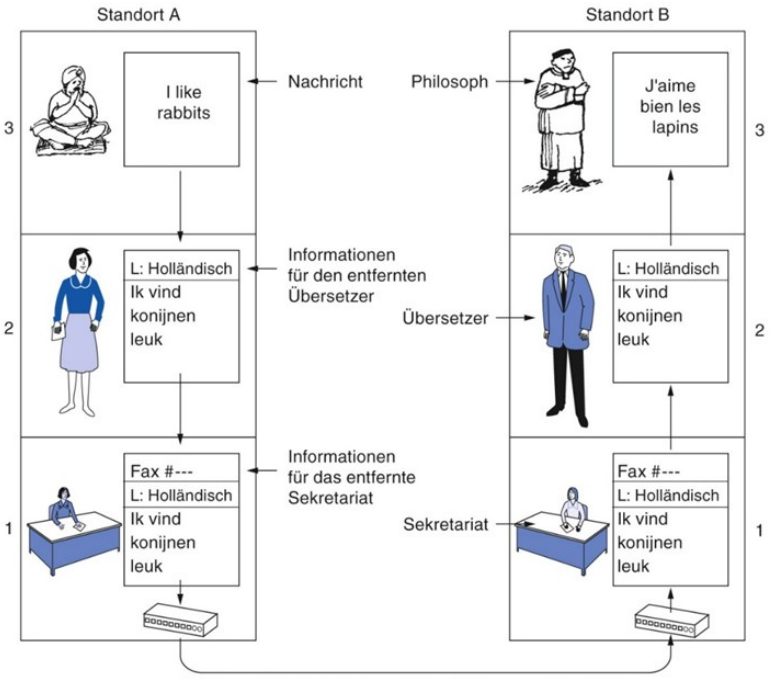


Abbildung 1.14: Die Architektur „Philosoph-Übersetzer-Sekretärin".

TCP/IP-Modell und Kommunikation

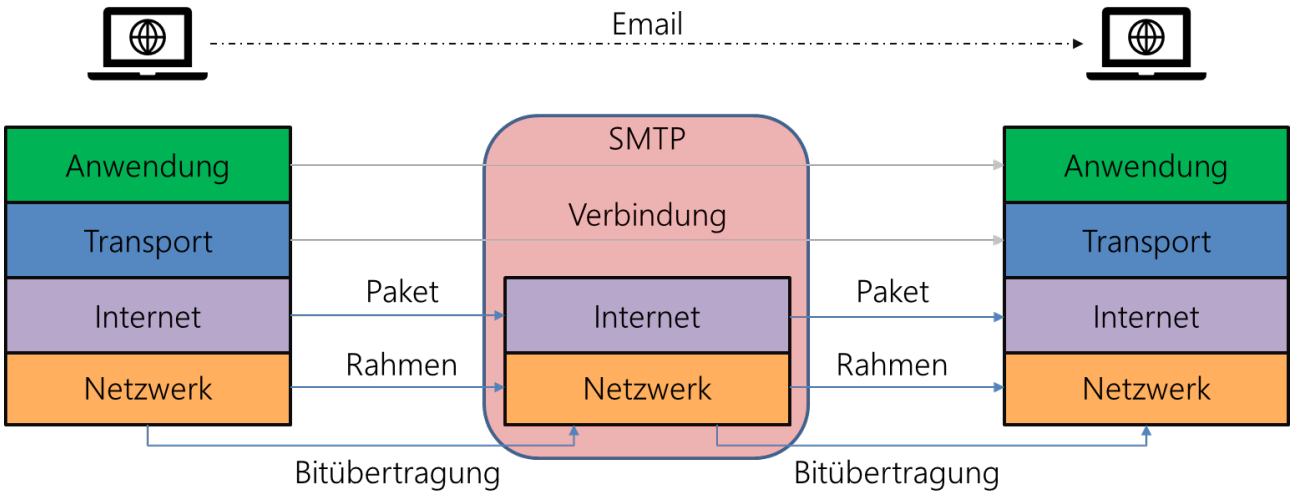
- Anwendungsschicht (Application Layer)
- Transportschicht (Transport Layer)
- Internetschicht (Internet Layer)
- Netzwerkschicht (Network Layer)

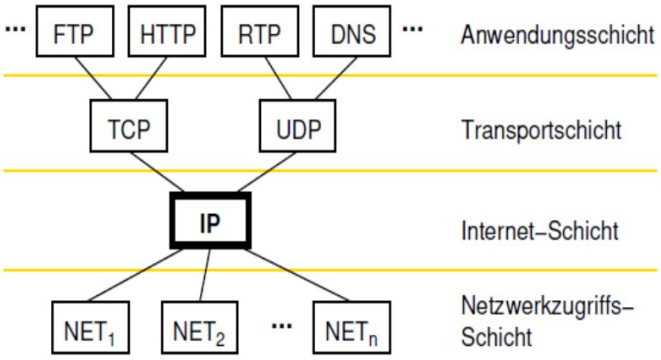
Applikationsdienst-Elemente abhängig von den ablaufenden Anwendungen
z.B. Dateizugriff (FTP), Fernverarbeitung (Telnet), Elektronische Post (SMTP), Name Service, WWW (HTTP)

netzunabhängiger Transport von Nachrichten zwischen zwei Endsystemen; Anpassung der Übertragungsqualität an Transportnetz
Flusssteuerung

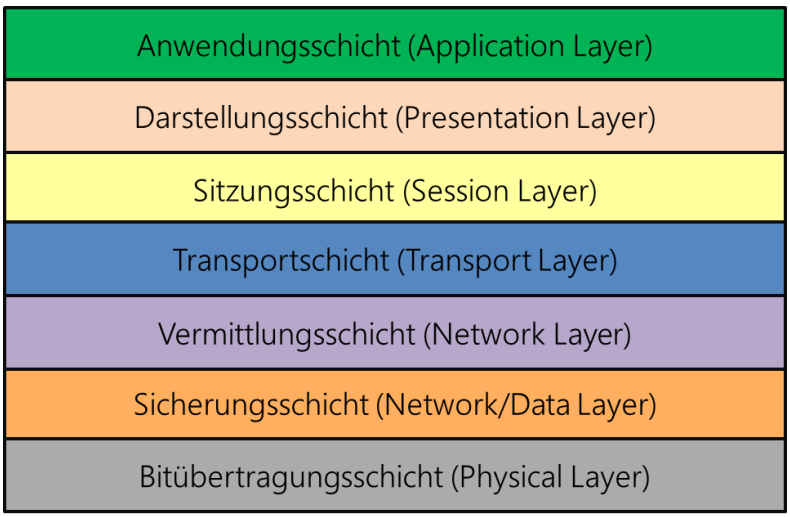
Zusammenschaltung von Teilstrecken zu einer End-zu-End-Verbindung, Wegewahl und Vermittlung
ohne Garantie auf korrekte Reihenfolge

Übertragungsmöglichkeit einzelner Dateneinheiten (Bits) unter bestimmten Zeitbedingungen an;
Nachrichtenübertragung zwischen zwei benachbarten Rechnern.

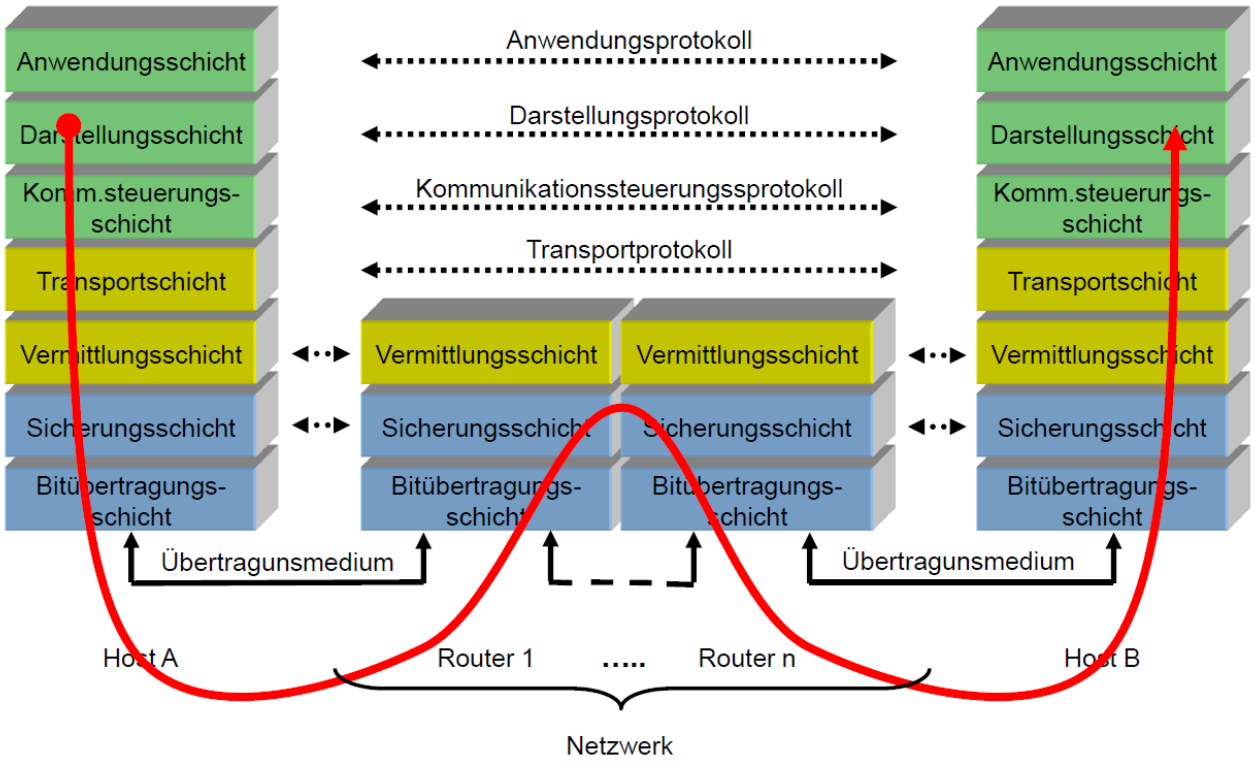




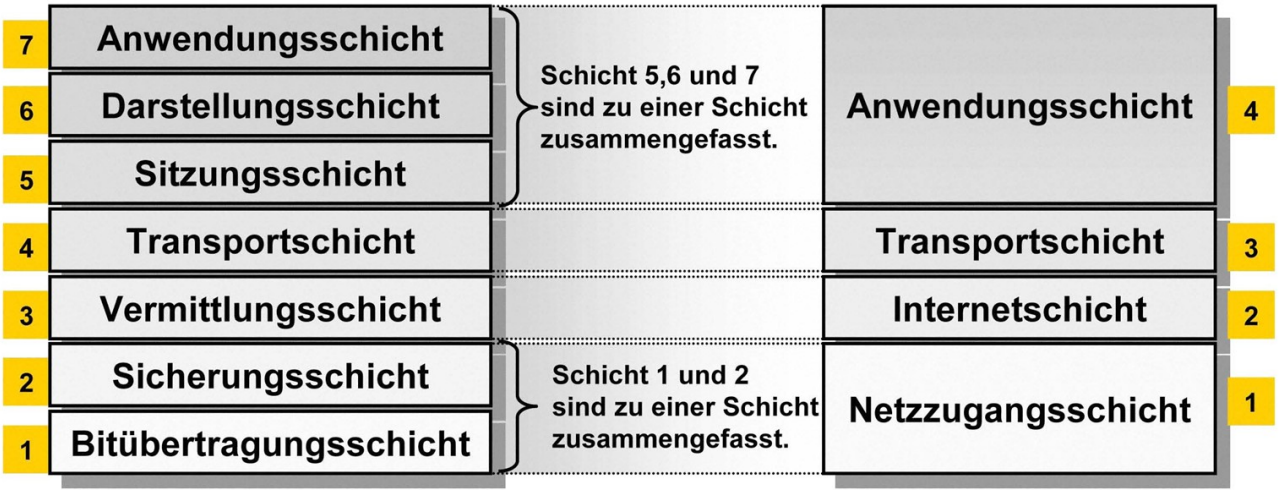
ISO/OSI-Modell und Kommunikation



- **Anwendungsschicht**: Bereitstellen von Diensten http, ftp, smtp, …
- **Darstellungsschicht**: prüft Daten, um sicherzustellen, dass diese Daten von der Anwendungsschicht mit den unteren Ebenen kompatibel sind (html, doc, jpeg, mp3, avi, Sockets, …)
- **Sitzungsschicht**:: Bereitstellen von Sitzungen, also verwaltet sie die Verbindung zwischen lokaler und entfernter Anwendung her; übernimmt auch Authentifizierungs- und Autorisierungsfunktionen
- **Transportschicht**: Übertragungskanal als Bytestrom mit Reihenfolge in einem fehlerfreien Punkt-zu-Punkt-Kanal
- **Vermittlungsschicht** ist hauptsächlich zuständig für die Auswahl der Routen zwischen den Rechnern
- **Sicherungsschicht** stellt eine Leitung ohne Übertragungsfehler bereit (Erkennen von Fehlern und Korrektur von Fehlern) und regelt die Zugriffssteuerung auf die Leitung
- **Bitübertragungsschicht**: Aufbau der Verbindung für eine physikalische (Elektrische / Optische Verbindung / Funk) Übertragung über eine 1:1-Verbindung



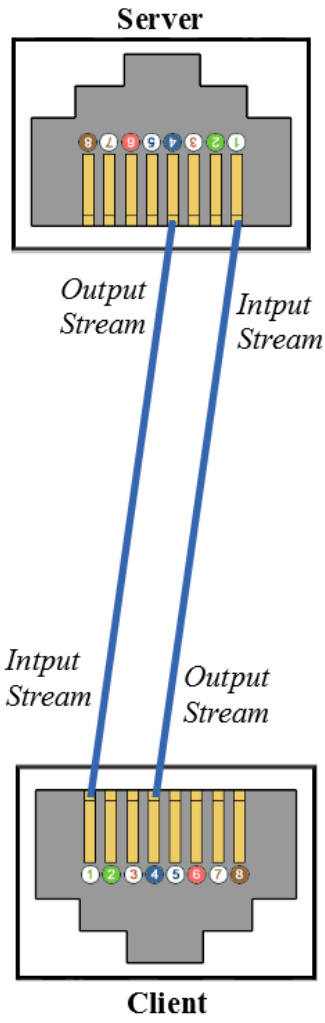
ISO/OSI-Modell vs. TCP/IP-Modell



Client-Server-Kommunikation in Java mit Sockets

Ein ***Socket*** ist ein Objekt, welches ein Programm nutzen kann um mit anderen entferntem Programm eine TCP-Verbindung aufzubauen und Daten auszutauschen. Diese Verbindungsstelle (Socket) wird durch eine Kombination aus ***IP-Adresse und Port*** repräsentiert wird. Mit einem Socket wird festgelegt, mit welchem Rechner und welchem Dienst man sich verbinden möchte. Für eine Kommunikation zwischen zwei Rechnern ist also ein Socket-Paar nötig: IP-Adresse und Port für den Server-Rechner als auch IP-Adresse und Port für den Client-Rechner.

Aufbau einer Client-Server-Verbindung mit Sockets



1. ***Der Server*** erzeugt ein ***ServerSocket***, welches am eigenen Netzwerkinterface an einem bestimmten Port – den man selbst angeben muss (>1024) – auf eine Verbindungsanfrage lauscht.
2. ***Der Client*** erstellt sich nun einen „gewöhnlichen“ Kommunikations-***Socket*** und gibt die entsprechende IP-Adresse und den (lauschenden) Port des Servers an und sendet darüber eine Verbindungsanfrage an den Server-Socket.
   Worum man sich beim Programmieren nicht kümmern muss: Der Client sendet dabei automatisch seine IP-Adresse und seinen Port mit.
3. Erhält ***der Server*** diese Anfrage, so erzeugt er ebenfalls ein „gewöhnliches“ Kommunikations-Socket, mit welchem er nun mit dem Client kommunizieren kann. Der ServerSocket kann nun erneut lauschen und weitere Verbindungen annehmen oder geschlossen werden.

Client und Server haben sich jetzt technisch abgestimmt. Jeder der beiden besitzt nun einen Sendekanal (***OutputStream***) und einen Empfangskanal (***InputStream***). Was die eine Seite sendet, landet bei der anderen Seite „im Briefkasten“. Dort kann es dann herausgenommen und verarbeitet werden.

Den Rest regelt dann ein speziell für diesen Dienst angefertigtes (oder für eigene Programme ein selbst geschriebenes) ***Protokoll***. Dies ist ein „Dialog“ zwischen Rechnern basierend auf einer formalen Sprache. So kann man dann in einem genau definierten Rahmen (gemäß dem Protokoll) dem anderen Rechner sagen, was genau man möchte und dieser weiß auch umgekehrt immer, was die gerade empfangenen Daten bedeuten sollen.

Aufgabe 3:

Bearbeite die Präsentation zur „Client-Server-Programmierung“.