

## KOMMUNIKATION – Lernziele

### (1) INTERNET

Die wichtigsten Kommunikationskonzepte, die dem Internet zugrunde liegen, werden verstanden

### (2) TCP

Das Transportprotokoll Transmission Control Protocol (TCP) kann in die Kommunikationsarchitektur eingeordnet werden und der TCP-Service kann mittels Java-Socketprogrammierung genutzt werden

### (3) HTTP

Die wichtigsten dem Anwendungsprotokoll HTTP zugrundeliegenden Konzepte sind bekannt und die Funktionsweise eines einfachen Web-Client-Programm und Web-Server-Programms werden verstanden

- (1) Dieses Lernziel behandelt den prinzipiellen Aufbau des Internet und stellt die den Internet-Protollen zugrundeliegenden Protokollsichten dar.
- (2) Zunächst werden Eigenschaften des durch TCP bereitgestellten Transportservices behandelt und anschließend werden die Konzepte zur Nutzung des TCP-Services in der Programmierung behandelt, was zur Socketprogrammierung führt.
- (3) Die grundlegenden Prinzipien (z.B. Anfrage-Antwort, nicht persistente Verbindungen) und Methoden (insbes. GET, POST) von HTTP werden soweit vorgestellt, dass ein einfacher Web-Client und Web-Server programmiert werden kann, der die im vorhergehenden Lernziel behandelte Socketprogrammierung nutzt.

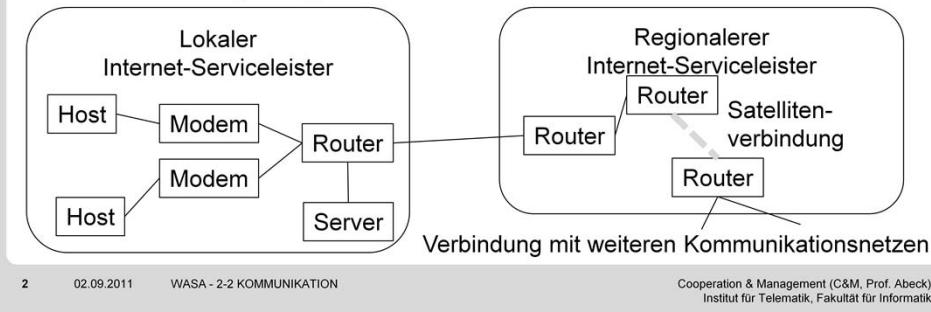
HTTP	HyperText Transfer Protocol
TCP	Transmission Control Protocol

Hauptquellen:

[KR03] James F. Kurose, Keith W. Ross: Computer Networking – A Top-down Approach Featuring the Internet, 2nd Edition, Addison Wesley, 2003.

## Das Internet

- (1) Das öffentliche Internet ist ein weltweites Rechnernetz, das Millionen von elektronischen Geräten (engl. computing devices) verbindet
  - (1) Unterscheidung von traditionellen und nicht-traditionellen Geräten
  - (2) Die Geräte werden als Hosts oder Endsysteme bezeichnet
- (2) Zwischen Hosts bestehen Kommunikationsverbindungen
  - (1) Verschiedene Typen von physikalischen Medien
  - (2) Verbindung erfolgt "indirekt" über Router



2 02.09.2011

WASA - 2-2 KOMMUNIKATION

Cooperation & Management (C&M, Prof. Abeck)  
Institut für Telematik, Fakultät für Informatik

[KR03:2]

Aufgrund der inhärenten Komplexität des Internet fällt es schwer, dieses Kommunikationsnetz in einem Satz erklären zu wollen.

(1) Der Beschreibung liegt der Ansatz zugrunde, das Internet von seinen Bestandteilen (Hardware- und Softwarekomponenten) zu erklären (engl. "nuts-and-bolts", Nüsse und Bolzen -> praktische Details).

Januar 2002: zwischen 100 und 500 Millionen Endsysteme hingen am Internet

(1.1) Traditionell: PCs, Unix-Workstations, Web-/Mail-Serversysteme

Nicht-traditionell: PDAs, mobile Rechner, Autos, Haushaltsgeräte (Toaster)

(1.2) Host heißt wörtlich übersetzt Wirt oder Wirtsrechner, d.h. es handelt sich um eine Rechner-Hardware, die die "Endsystem-Funktionalität beherbergt".

(2) Die Kommunikationsverbindung kann sich aus einem oder mehreren Übertragungsstrecken (engl. communication link) zusammensetzen.

(2.1) Kabelgebunden (Koaxial, Kupfer, Fiberglas -> optisch) oder Funkverbindung (engl. radio spectrum).

Eine wichtige mit dem Medium unmittelbar verbundene Eigenschaft ist die Übertragungsrate, genannt die Bandbreite (engl. bandwidth) der Verbindung.

(2.2) Router erhalten Informationsstücke (engl. information chunk), genannt Pakete, an einer eingehenden Kommunikationsverbindung und leiten (engl. forward) diese an einer ausgehenden Verbindung weiter.

(Internet-Serviceleister) Jedes Endsystem greift auf das Internet über einen Internet-Serviceerbringer (Internet Service Provider ISP) zu. ISPs können auf die Wohnung (engl. residential) bezogen (z.B. AOL) sein, universitär oder unternehmensbezogen (engl. corporate) sein [4].

(Modem) Verschiedene Arten lassen sich unterscheiden, wie z.B.:

(i) Einwähl (engl. dial-up, 56 kbps)

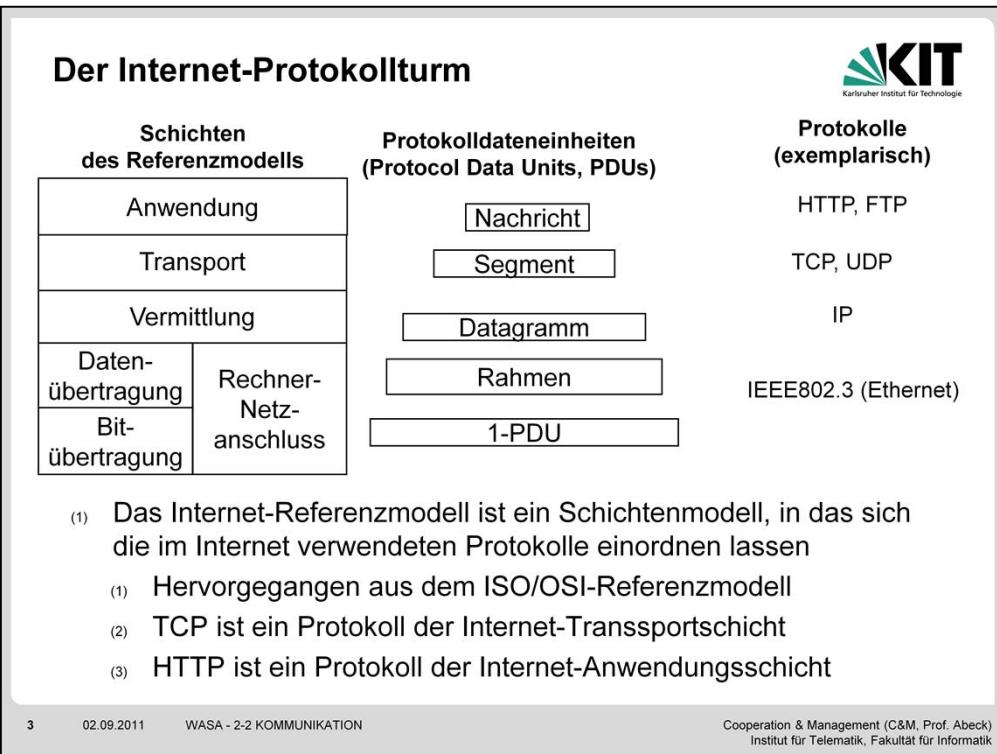
(ii) Digital Subscriber Line (DSL, Standard zur Übertragung hoher Datenraten über einfache Kupferleitungen, insbes. Telefonanschlussleitungen)

(iii) FunK-LAN (engl. wireless)

DSL Digital Subscriber Line

ISP Internet Service Provider

Addison Wesley, 2003.



[KR03:54]

(Schichten) Der Internet-Turm (engl. stack) besteht aus 5 Schichten (bzw. 4 im Falle des Zusammenziehens der unteren beiden Schichten) [:54].

Das dem Internet zugrunde liegende Referenzmodell geht aus dem sieben Schichten umfassenden ISO/OSI-Referenzmodell hervor, wobei die drei oberen Schichten (Session, Presentation, Application) zu nur einer Schicht zusammengefasst wurden.

(PDUs) Außer für die ersten Schicht (engl. physical layer, Bitübertragungsschicht) bestehen für alle Schichten spezifische Bezeichnungen für die darin auftretenden PDUs.

(Anwendung) Die für die Unterstützung von Netzanwendungen (engl. network application) verantwortliche Schicht.

(Transport) Erbringt den Service zum Transport von Nachrichten der Anwendungsschicht zwischen der Client- und Server-Seite einer Anwendung.

(Vermittlung, engl. network layer) Routen von Diagrammen von einem Host (=Endsystem) zu einem anderen.

(Datenübertragung, engl. Link) Zuverlässige (engl. reliable) Übertragung von einem Knoten zu einem über ein Medium direkt verbundenen Nachbarknoten.

(Bitübertragung, engl. physical) Übertragung der einzelnen Bits, die innerhalb eines Rahmens (Frame, 2-PDU) auftreten.

(Protokolle) Die im Zusammenhang mit dem Internet entwickelten Protokolle werden als TCP/IP-Protokolle bezeichnet und umfassen neben dem Transportprotokoll TCP (Transmission Control Protocol) und dem Vermittlungsprotokoll IP (Internet Protocol) unter anderem auch Protokolle der Anwendungsschicht (z.B. HyperText Transfer Protocol HTTP oder File Transfer Protocol FTP).

(1.1) Das TCP/IP-Modell ist als Modell weniger relevant. An dieser Stelle hat die ISO die wesentlichen Grundlagen erarbeitet, die z.T. auch Eingang in das TCP/IP-Referenzmodell gefunden haben. Interessant im Zusammenhang mit TCP/IP sind vielmehr die konkreten Protokolle, die das Modell auffüllen. Hier besteht ein genau umgekehrtes Verhältnis zu der ISO, deren Protokolldefinitionen zu keinem Zeitpunkt die hohe Akzeptanz der TCP/IP-Protokolle erreicht hat. Während die ISO die modelltechnischen Grundlagen für Kommunikationssysteme lieferte, resultierten aus den Internet-Aktivitäten allgemein akzeptierte Protokollstandards in Form der TCP/IP-Protokolfamilie.

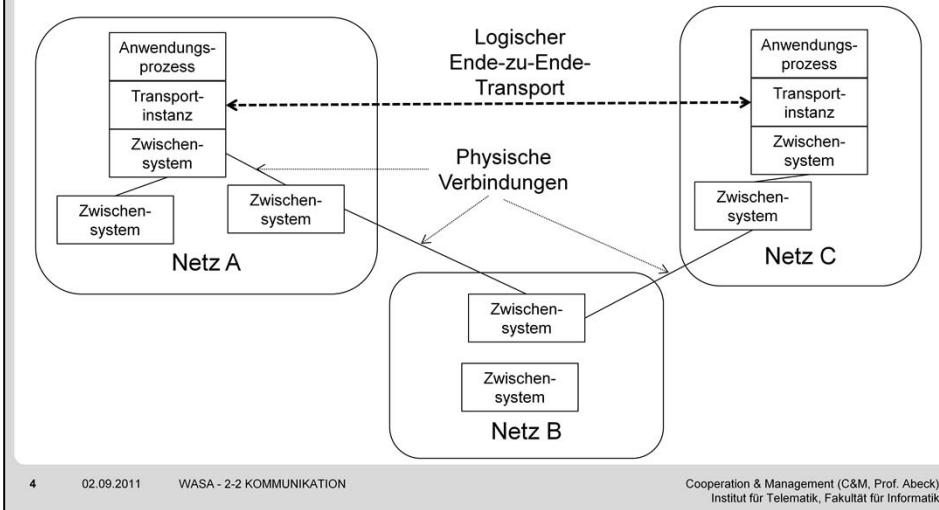
(1.2) (1.3) Diese beiden Protokolle werden im Folgenden näher behandelt.

FTP	File Transfer Protocol
HTTP	HyperText Transfer Protocol
IP	Internet Protocol
PDU	Protocol Data Unit
TCP	Transmission Control Protocol
UDP	User Datagram Protocol

[KR03] James F. Kurose, Keith W. Ross: Computer Networking – A Top-down Approach Featuring the Internet, 2nd Edition, Addison Wesley, 2003.

## Transportschicht

- (1) Das Transportprotokoll erbringt eine logische Kommunikation zwischen Anwendungsprozessen



4

02.09.2011 WASA - 2-2 KOMMUNIKATION

Cooperation & Management (C&M, Prof. Abeck)  
Institut für Telematik, Fakultät für Informatik

[KR03:80, 185]

(1) Die Anwendungsprozesse laufen üblicherweise auf unterschiedlichen Hostsystemen.

Logische Kommunikation bedeutet hier, dass die Anwendungsprozesse (bzw. die Hosts, auf denen sie laufen) sich direkt miteinander verbunden fühlen (obwohl diese real nicht direkt, sondern nur über ein oder mehrere Zwischenstationen verbunden sind).

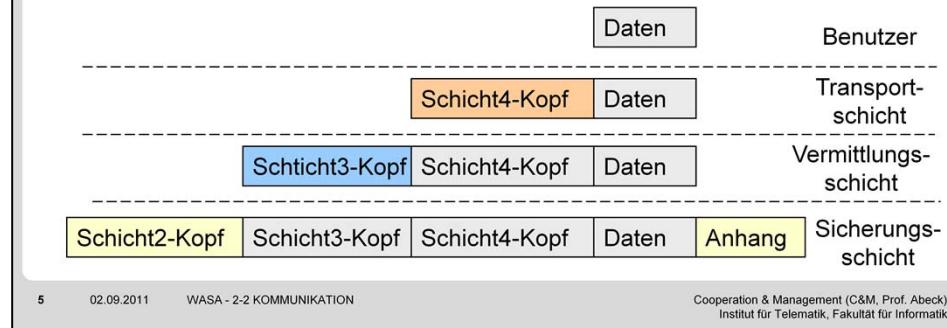
(Physische Verbindungen) Die logische Kommunikation läuft über eine Kette von physischen Verbindungen.

(Zwischensysteme) Sind Router bzw. Vermittlungsstellen, die die Instanzen der Vermittlungsschicht (engl. network layer, Schicht 3) und die darunterliegende Sicherungsschicht (engl. data link layer, Schicht 2) und Bitübertragungsschicht (engl. physical layer, Schicht 1) beinhalten.

[KR03] James F. Kurose, Keith W. Ross: Computer Networking – A Top-down Approach Featuring the Internet, 2nd Edition, Addison Wesley, 2003.

## Zusammenspiel der Schichten (allgemein)

- (1) Sicherungsschicht-Instanzen (Schicht 2) sichern die Übertragung einer physischen Verbindung (engl. link)
- (2) Vermittlungsschicht-Instanzen (Schicht 3) leiten Dateneinheiten durch das Netz zum Empfänger
- (3) Transportschicht-Instanzen (Schicht 4) fügen Anwendungsadressierung zur Vermittlungsschicht hinzu



[KR03:186]

(1) Die Sicherungsschicht setzt voraus, dass zwischen den beiden Schicht2-Instanzen eine physische Verbindung besteht.

Zum Begriff "Link": In der englischen Bezeichnung "Data Link Layer" tritt der Begriff explizit auf.

(2) Auf der Vermittlungsschicht werden die Daten (= Benutzerdaten+Schicht4-Steuerinformation) von Zwischensystem (reales System, auch als Host bezeichnet) zu Zwischensystem bis zum Zielsystem transportiert.

Allgemeiner Zusammenhang: Steuerdaten der Schicht N sind transparent zu übertragene (Benutzer-) Daten auf der Schicht N-1

im Beispiel: Schicht3-Kopf (= Schicht3-Steuerdaten, d.h. N=3) sind für die Schicht 2 (=N-1) Daten.

(3) Wie die Abbildung zeigt, wird der Schicht3-Kopf (= Steuerinformation) an den Schicht4-Kopf angefügt.

Allgemein sind die Anwendungsadressen TSAPs (Transport Service Access Points); im TCP/IP-Fall heißen diese Adressen Ports, wie auf einer späteren Seite noch verdeutlicht wird.

Die Transportschicht-Instanz stellt darüberhinaus einen zuverlässigen Service bereit.

TSAP              Transport Service Access Point

[KR03] James F. Kurose, Keith W. Ross: Computer Networking – A Top-down Approach Featuring the Internet, 2nd Edition, Addison Wesley, 2003.

- (1) Zum Internet-Referenzmodel sind jeweils eine Schicht und ein darin auftretendes Protokoll zu nennen
- (2) Durch welche Schicht wird eine logische Verbindung geschaffen?
- (3) Das Transportprotokoll stellt einen Übertragungsdienst bereit für
  - (1) Zwischensysteme
  - (2) Anwendungsprozesse
  - (3) Transportprotokollinstanzen
- (4) Welche Protokolle stehen mit TCP in einer Nutzungsbeziehung?
  - (1) UDP
  - (2) HTTP
  - (3) IP
  - (4) SMTP
  - (5) SNMP

## Transportprotokoll aus der Sicht einer Anwendung

- (1) Ein Anwendungsentwickler muss sich für ein Transportprotokoll entscheiden, das den Anforderungen der Anwendung am nächsten kommt
- (2) Klassifikation der Serviceanforderungen einer Anwendung entlang der folgenden drei Dimensionen
  - (1) Datenverlust
  - (2) Bandbreite
  - (3) Zeitsensitivität
- (3) Beispiele
  - (1) Dateitransfer, E-Mail: kein Datenverlust, elastische Bandbreite, nicht zeitsensitiv
  - (2) Audio/Video, interaktive Spiele: Datenverlust-tolerant, Mindestbandbreite, harte Zeitanforderungen

[KR03:83]

(1) Das entspricht ungefähr der Situation, sich bei einer Reise zwischen zwei Städten für den Zug oder das Flugzeug zu entscheiden.

(2.1) Während eine Multimedia-Anwendung einen gewissen Datenverlust (u.a. vom Kodierungsschema abhängig) tolerieren kann, gilt das für eine Finanztransaktion nicht.

(2.2) Internet-Telefonie ist ein Beispiel einer Bandbreiten-sensitiven Anwendung, da sie eine gewisse Mindestbandbreite (Kodierung üblicherweise von 32 kbps) erfordert.

Anwendungen wie E-Mail oder Web-Übertragung sind Beispiele für elastische Anwendungen.

(2.3) Interaktive Realzeit-Anwendungen wie Internet-Telefonie, Teleconferencing oder Multiplayer-Spiele erfordern harte Zeitbeschränkungen, z.B. wenige hundert Millisekunden Ende-zu-Ende-Verzögerung.

(3.1) Diese Klasse von Anwendungen (hierzu gehört auch die Übertragung von Web-Dokumenten) haben lediglich die Anforderung, dass keine Daten verloren gehen dürfen, sind hinsichtlich Bandbreite und Zeitanforderungen unkritisch [84].

(3.2) Bandbreite Audio: wenige kbps bis 1 Mbps, Video: 10 kbps bis 5 Mbps

Zeitanforderungen: mehrere 100 msec, bei gespeichertem (stored) Audio/Video erhöht sich die Zeitgrenze auf einige Sekunden

[KR03] James F. Kurose, Keith W. Ross: Computer Networking – A Top-down Approach Featuring the Internet, 2nd Edition, Addison Wesley, 2003.

- (1) Eine Anwendung erhält die folgenden zwei Services von TCP
  - (1) Verbindungsorientierter Service
    - (1) Voll-Duplex-Verbindung nach Durchführung eines Handshaking-Verfahrens
  - (2) Zuverlässiger Transportservice
    - (1) Übertragung der Daten ohne Fehler und in der richtigen Reihenfolge
  - (2) Staukontroll-Mechanismus zur Vermeidung der Überlastung des Netzes
    - (1) Hieraus resultiert die UDP-Verwendung von Realzeitanwendungen
  - (3) Service-Eigenschaften, die TCP nicht bereitstellt
    - (1) Garantie einer minimalen Übertragungsrate
    - (2) Garantie von Verzögerungszeiten

[KR03:85]

Wenn ein Entwickler eine Anwendung für das Internet erzeugt, ist eine seiner ersten Entscheidungen, ob er TCP oder UDP verwendet, da beide Protokolle der aufrufenden Anwendung unterschiedliche Services bereitstellen.

- (1.1) Die Verbindung existiert zwischen den Sockets der beiden Prozesse.
- (1.2) Der auf der einen Seite in den Socket geleiteten Bytestrom kommt in der gleichen Weise (ohne fehlendes oder dupliziertes Byte) auf der anderen Seite wieder heraus.
- (2) Liefert weniger einen direkten Vorteil für die kommunizierende Prozesse als vielmehr für das gesamte Internet.
  - (2.1) Ein zu starkes Drosseln (engl. throttle) der Übertragungsrate kann zu einer Unterschreitung der erforderlichen Bandbreite führen.
- (3.1) Die Senderate wird durch den Staukontroll-Mechanismus vorgegeben.
- (3.2) Es wird nur garantiert, dass die Daten beim Empfänger ankommen - aber nicht, bis wann das erfolgt. Ein Symptom dieser fehlenden Service-Eigenschaft ist das "world wide wait".

[KR03] James F. Kurose, Keith W. Ross: Computer Networking – A Top-down Approach Featuring the Internet, 2nd Edition, Addison Wesley, 2003.

- (1) Schlichtes leichtgewichtiges Transportprotokoll
- (2) Minimalistisches Servicemodell
- (3) Verbindungslos
- (4) Keine Fehler- oder Reihenfolgebehandlung
- (5) Keine Unterstützung des Staukontroll-Mechanismus

[KR03:86]

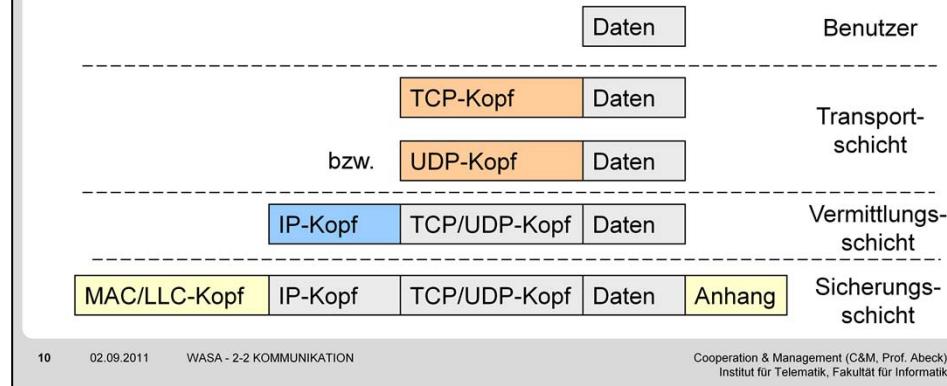
Alle Eigenschaften des UDP-Services verdeutlichen den minimalistischen Ansatz, der diesem Protokoll zugrunde liegt.

- (1) Einfach zu implementieren
- (2) Wenige Services mit den nötigsten Parametern
- (3) Es wird kein Handshake-Verfahren zum Aufbau einer Verbindung durchgeführt
- (4) Daher gibt UDP diesbezüglich auch keinerlei Garantie.
- (5) Aus diesem Grund kann ein Prozess Daten mit beliebiger Rate in den UDP-Socket pumpen, weshalb Entwickler von Realzeitanwendungen den UDP-Service wählen.

[KR03] James F. Kurose, Keith W. Ross: Computer Networking – A Top-down Approach Featuring the Internet, 2nd Edition, Addison Wesley, 2003.

## Zusammenspiel der Schichten (TCP/IP)

- (1) IP leitet Dateneinheiten durch das Netz zum Empfänger
- (2) TCP/UDP fügen Anwendungsadressierung zu IP hinzu
- (3) TCP stellt darüberhinaus zuverlässigen Service bereit



[KR03:185]

(1) (IP-Kopf) Wichtige Felder im Kopf (engl. header) eines IP-Pakets sind [KR05:326]:

- Quell- und Zieladresse (engl. source and destination addresses): Im Falle von IPv4 handelt es sich um eine 32-bit Adresse (IPv6: 128 bit).
- Länge: Gesamtlänge des IP-Datagramms inklusive Header und Daten (max. 65.535 Bytes, üblicherweise nicht länger als 1500 Bytes)
- Time-to-live (TTL): Zähler, der beim Durchlauf des Datagramms durch einen Router heruntergezählt wird. Bei 0 wird das Datagramm vernichtet, wodurch ein endloses Zirkulieren verhindert wird.

(2) (TCP-Kopf) Wichtige Felder im Kopf eines TCP-Segments [:231]:

- Quell-Portnummern, Ziel-Portnummer: 16-bit lange Nummer (Wertebereich von 0 bis 65535), durch die die Anwendung identifiziert wird (Well-known Port Numbers, z.B. 80: HTTP, 21: FTP) und die Voraussetzung für das Multiplexing ist [:190]. Diese Felder befinden sich auch im UDP-Kopf.
- Sequenznummer, Bestätigungsnummer: 32-bit lange Nummern, die sich auf die Bytenummer im Übertragungsstrom beziehen und zur Umsetzung einer zuverlässigen Datenübertragung genutzt werden.

TTL                  Time-to-live (IP)

[KR03] James F. Kurose, Keith W. Ross: Computer Networking – A Top-down Approach Featuring the Internet, 3rd Edition, Addison Wesley, 2005.

Anwendung	Anwendungs-protokoll	Verwendetes Transportprotokoll
E-Mail	SMTP	TCP
Remote Terminal Access	Telnet	TCP
Web	HTTP	TCP
Dateitransfer	FTP	TCP
Entfernter Fileserver	NFS	TCP, urspr. UDP
Streaming Multimedia	RTSP, ...	i.d.R. UDP
Internet-Telefonie	SIP/H.323 & RTP	i.d.R. UDP
Netzmanagement	SNMP	i.d.R. UDP
Intra-Domain-Routing	RIP	i.d.R. UDP
Inter-Domain-Routing	BGP	TCP
Namensübersetzung	DNS	i.d.R. UDP

[ITM-EiR:Transportschicht-13] , [KR03:87]

(SMTP) Simple Mail Transfer Protocol: Da E-Mails sehr lang sein können und deren fehlerfreie Übertragung erforderlich ist, wird das verbindungsorientierte Transportprotokoll TCP gewählt.

(NFS) Network File System: Von SUN für das UNIX-Betriebssystem entwickeltes Protokoll, das den Zugriff von Dateien über ein Netzwerk ermöglicht. Das Analogon heißt beim Windows-Betriebssystem Server Message Block (SMB). Ursprünglich wurde nur UDP, ab der Version 4 wird insbesondere wegen Netzwerksicherheit TCP unterstützt.

(RTSP) Real Time Streaming Protocol: Zur Steuerung der Session zur kontinuierlichen Übertragung von audiovisuellen Daten (Streams). Textbasiertes Protokoll, das im Aufbau und Verhalten dem HTTP ähnelt und über TCP oder UDP übertragen werden kann.

(SIP) Session Initiation Protocol: Zum Aufbau, Steuerung und Abbau einer Kommunikationssitzung.

H323 ist das von der ITU standardisierte Pendant und ist vereinfacht ein "ISDN über IP".

(RTP) Real-Time Transport Protocol: Protokoll zur Übertragung von Daten in Echtzeit.

(SNMP) Simple Mail Transfer Protocol: Verbindungslose Kommunikation mittels UDP ist insbesondere im Fall eines instabilen Netzes (aufgrund eines Fehlerfalls) gegenüber der Verbindungsorientierung (TCP) zu bevorzugen, da Daten größere Chancen haben, übertragen zu werden.

(RIP) Routing Information Protocol: Interior Gateway Protocol (eingesetzt innerhalb eines autonomen Systems -> LAN), das auf der Basis des Distanzvektoralgorithmus (Dijkstra) arbeitet und die Routingtabellen von Routern automatisch erstellt. Wegen der Verwendung von UDP ist nicht garantiert, dass die zu benachbarten Routern versendeten Informationen auch tatsächlich ankommen.

(BGP) Border Gateway Protocol: Providerübergreifendes Routingprotokoll, das beschreibt, wie Router untereinander die Verfügbarkeit von Verbindungswegen zwischen den Netzen autonomer Systeme (= einzelnes Internetprovidernetz) weitergeben.

(DNS) Domain Name Service: Einer der wichtigsten Services im Netz, der die Anfragen zur Namensauflösung beantwortet. Mittels eines auf weltweit verteilten Servern verteilten hierarchisch organisierten Verzeichnisses wird der Hostname (Domainname, z.B. www.example.org) auf eine IP-Adresse (IPv4 oder IPv6) abgebildet.

DNS-Anfragen werden über UDP-Port 53 zum Nameserver gesendet; der DNS-Standard erlaubt aber auch die Verwendung von TCP.

- (1) Es ist eine Anwendung mit weichen und harten Zeitanforderungen an den Transportdienst zu nennen
- (2) Wodurch kann das "World Wide Wait" im Internet überhaupt entstehen?
- (3) Wozu dient die Adressierung in der Transportschicht und wie sehen die Adressen bei TCP/UDP aus?
- (4) Aus welchem Grund ist UDP für die Unterstützung von Realzeitanwendungen geeigneter als TCP?
- (5) Warum wird im Netzmanagement UDP genutzt?

## Nutzung des Transportservices durch eine Netzanwendung

- (1) Netzanwendungen setzen sich üblicherweise aus zwei verteilten (Client- und Server-) Prozessen zusammen
  - (1) Die Kommunikation erfolgt durch Austausch von Nachrichten
- (2) Das Senden und Empfangen von Nachrichten erfolgt über Sockets
  - (1) Bilden die Schnittstelle zwischen der Anwendungsschicht und der Transportschicht
- (3) Ein Anwendungsentwickler hat nur wenig Einfluss auf die Seite der Transportschicht des Sockets
  - (1) Wahl des Transportprotokolls
  - (2) Bestimmung gewisser Transportschicht-Parameter
- (4) Adressierung des empfangenen Prozesses durch zwei Informationen
  - (1) Name oder Adresse des Hosts
  - (2) Identifikator des empfangenen Prozesses im Ziel-Host

[KR03:79]

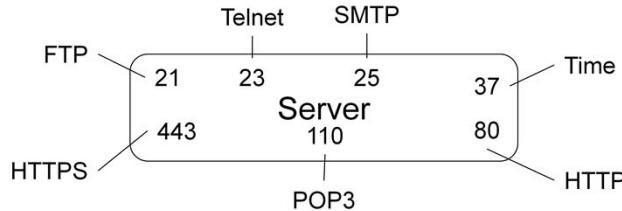
- (1) Der Client-Prozess (z.B. ein Web-Browser) und der Server-Prozess (Web-Server) laufen jeweils in unterschiedlichen Endsystemen.
  - (1.1) Der Aufbau der Nachrichten und die Reihenfolge des Austauschs wird durch das Anwendungsschicht-Protokoll festgelegt [:79].
- (2) Ein Socket entspricht einer Tür, die die Anwendung zum darunterliegenden Transportinfrastruktur öffnen kann.
  - (2.1) Da Sockets die Programmierschnittstellen darstellen, mit denen die Netzanwendungen gebildet werden, heißen die Schnittstellen auch APIs (Application Programming Interfaces) [:81].
- (3) Die Anwendungsschicht-Seite wird hingegen vollständig vom Entwickler kontrolliert.
  - (3.1) Falls überhaupt mehr als ein Protokoll zur Verfügung steht.
  - (3.2) Z.B. maximale Puffergröße oder maximale Segmentgröße
    - (4.1) Die Host-Adresse ist in Internet-Anwendungen die IP-Adresse.
    - (4.2) Dieser Identifikator ist die sog. Portnummer, die den Typ der Anwendung eindeutig identifiziert.

API Application Programming Interface

[KR03] James F. Kurose, Keith W. Ross: Computer Networking – A Top-down Approach Featuring the Internet, 2nd Edition, Addison Wesley, 2003.

## Ports

- (1) Eigenschaften eines TCP-Port
  - (1) 16-Bit-Nummer (Port 0 bis 65535)
  - (2) Repräsentieren eindeutig eine logische Verbindung zu einem spezifischen Software-Programm auf dem Server
  - (3) Wird vom Client gewählt, um sich mit einer bestimmten Anwendung zu verbinden
- (2) Wohlbekannte TCP-Port-Nummern
  - (1) Port 0 bis 1023 sind reserviert



14 02.09.2011

WASA - 2-2 KOMMUNIKATION

Cooperation & Management (C&M, Prof. Abeck)  
Institut für Telematik, Fakultät für Informatik

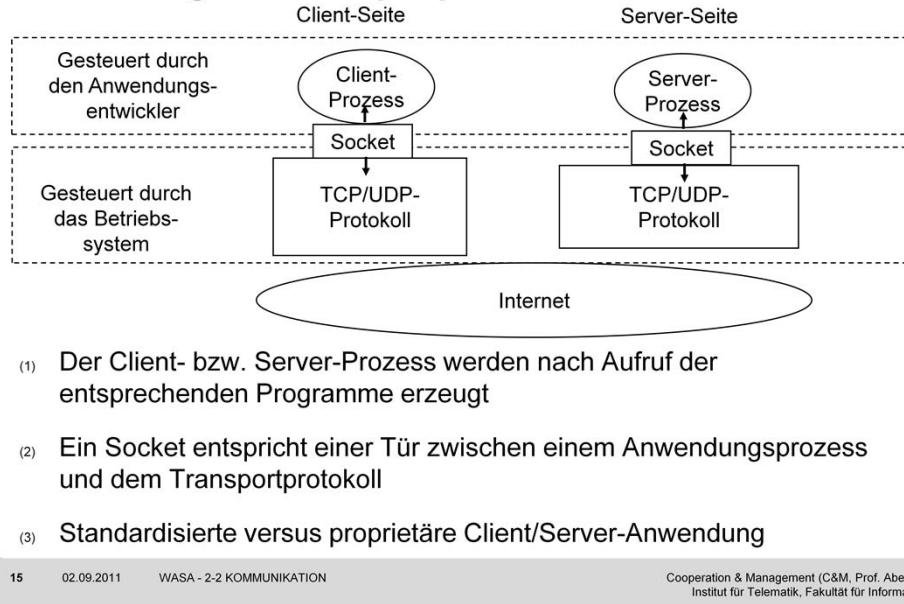
[BS+08:21]

- (1) Ports sind Adressen auf der Ebene von TCP und somit auf der Transportschicht.
  - (1.1) 0 bis  $(2^{16}) - 1$
  - (1.2) Sie repräsentieren hingegen nicht einen Platz auf der Server-Hardware, in den man physische Geräte einstecken (engl. plug in) kann.
  - (1.3) Ohne Port-Nummern hätte der Server keine Möglichkeit herauszufinden, mit welcher Anwendung sich der Client verbinden möchte.
- (2) Wohlbekannt (engl. well-known) heißt hier, dass an diesen Portnummern durch den Standard festgelegte und damit für ein Client-Programm wohlbekannte Server-Programme (Kommunikationsanwendungen wie z.B. WWW oder E-Mail) erreichbar sind.
  - (2.1) Diese Port-Nummen sollten daher nicht für die eigenen (proprietären) Server-Programme genutzt werden.

(POP3, HTP) Durch die eindeutigen, vorab vergebenen Port-Nummern wird z.B. verhindert, dass ein Web-Browser irrtümlich bei einem POP3-Mail-Server (Port 110) anstelle des gewünschten Web-Servers (Port 80) anfragt.

[BS+08] Bryan Basham, Kathy Sierra, Bert Bates: Head First Servlets & JSP, Second Edition, O'Reilly, 2008.

## Anwendungsprozesse, Sockets und das darunterliegende Transportprotokoll



[KR03:82,:146]

(1) Eine Client-Server-basierte Web-Anwendung besteht aus einem Paar von Prozessen bzw. Programmen.

(2) Die Anwendungsprozesse laufen auf verschiedenen Rechnern und senden bzw. empfangen Nachrichten, indem sie Ein-/Ausgabeoperationen auf dem ihnen zur Verfügung stehenden Socket durchführen.

(3) Das hängt davon ab, ob sich das Client- und Server-Programm an einen Standard hält (z.B. einen RFC wie z.B. FTP oder HTTP) oder nicht.

Die nachfolgend entwickelte Anwendung ist zunächst proprietär, da sie sich an keinen Standard hält. Mit der Einführung eines minimalen Web-Servers wird ein Schritt in Richtung einer "Standard-Web-Anwendung" vorgenommen.

(Gesteuert durch ...) Zu beachten ist, dass ein Anwendungsentwickler unterhalb der Schnittstelle, also auf der Ebene des Transportprotokolls (TCP/UDP) höchstens gewisse Transportparameter ändern kann (bei TCP z.B. die maximale Puffergröße) und somit nur sehr eingeschränkte Einflussmöglichkeiten besitzt.

RFC

Request for Comments

[KR03] James F. Kurose, Keith W. Ross: Computer Networking – A Top-down Approach Featuring the Internet, 2nd Edition, Addison Wesley, 2003.

## Sockets

- (1) Konzept zur Unterstützung der Programmierung von netzbasierten Anwendungen
- (2) Stellt in einem lokalen Betriebssystem die Abstraktion des Kommunikationsendpunktes eines bestimmten Transportprotokolls dar
- (3) Im Internet-Umfeld ist zu wählen zwischen
  - (1) Socket-Programmierung mit TCP
    - (1) Zuverlässiger Bytestrom-Kanal
  - (2) Socket-Programmierung mit UDP
    - (1) Unabhängig voneinander gesendete Datenpakete ohne Auslieferungsgarantie

[Ta08:166]

(1) Durch Sockets wird eine nachrichtenorientierte Kommunikation unterstützt. Die alternative (gegensätzliche) Kommunikationsform ist der entfernte Prozedurauftrag (Remote Procedure Call, RPC) [Ta08:166].

(2) Es bestehen verschiedene Beispiele von Socket-Schnittstellen, die im Wesentlichen auf dem gleichen Kommunikationsmodell basieren, aber sich in den an der Programmierschnittstelle angebotenen Socket-Befehle unterscheiden. Beispielhaft sei das Betriebssystem Berkley Unix genannt, in das eine Socket-Schnittstelle ("Berkley-Sockets") im Jahr 1970 eingefügt wurde.

(3.1) Meist werden aufgrund der höheren Zuverlässigkeit die TCP-basierten Stream-Sockets eingesetzt. Hierauf wird im Folgenden näher eingegangen.

RPC

Remote Procedure Call

[Ta08] Andrew S. Tanenbaum, Marten van Stehen: Verteilte Systeme – Prinzipien und Paradigmen, Pearson Studium, 2008.

- (1) Start der die Netzanwendung bildenden Client- und Server- Programme erzeugt entsprechende Prozesse auf den Hosts
- (2) Client/Server-Netzanwendungen können konform zu einem RFC- Standard oder proprietär sein
- (3) Der Entwickler muss das von der Anwendung zu nutzende Transportprotokoll festlegen
  - (1) TCP: verbindungsorientiert, zuverlässiger Bytestrom-Kanal
  - (2) UDP: verbindungslos, voneinander unabhängige Pakete
- (4) Zur Erstellung der Client-/Server-Programme kann eine höhere Programmiersprache wie C, C++ oder Java genutzt werden

[KR03:133]

- (1) Diese Client- und Server-Prozesse kommunizieren über Sockets [:133].
- (2) Im Falle einer standardkonformen Anwendung können verschiedene Entwickler Client-Programme (z.B. ein Netscape-Browser) und Server-Programme (z.B. ein Apache Web-Server) entwickeln, die miteinander auf der Grundlage des verwendeten RFC (Request for Comment) kommunizieren können.  
Im Falle einer proprietären Anwendung dürfen die Entwickler keinen der in den RFCs beschriebenen Well-known Ports verwenden.
- (3) Das ist eine der ersten Entscheidungen im Entwicklungsprozess der Netz-Anwendung.
- (4) Im Folgenden wird Java genutzt, da hiermit die Programme besonders sauber und kurz gestaltet werden können.

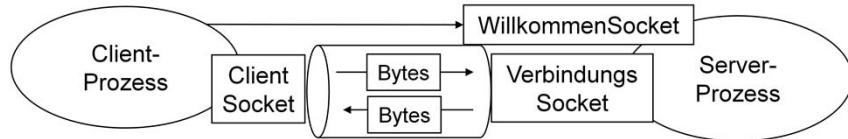
RFC

Request for Comments

[KR03] James F. Kurose, Keith W. Ross: Computer Networking – A Top-down Approach Featuring the Internet, 2nd Edition, Addison Wesley, 2003.

## Aufbau einer TCP-Verbindung

- (1) Client initiiert den Kontakt mit dem Server
  - (1) Ein Server-Prozess muss bereits auf dem Server laufen und über einen definierten Socket ansprechbar sein
- (2) Client initiiert eine TCP-Verbindung durch Erzeugung eines Client-Socket
- (3) Aufbau der TCP-Verbindung durch ein 3-Wege-Handshake
  - (1) Erzeugung eines neuen Server-seitigen Verbindungs-Socket
- (4) Ergebnis: TCP-Verbindung zwischen Client-Socket und Server-seitigem Verbindungs-Socket



18

02.09.2011

WASA - 2-2 KOMMUNIKATION

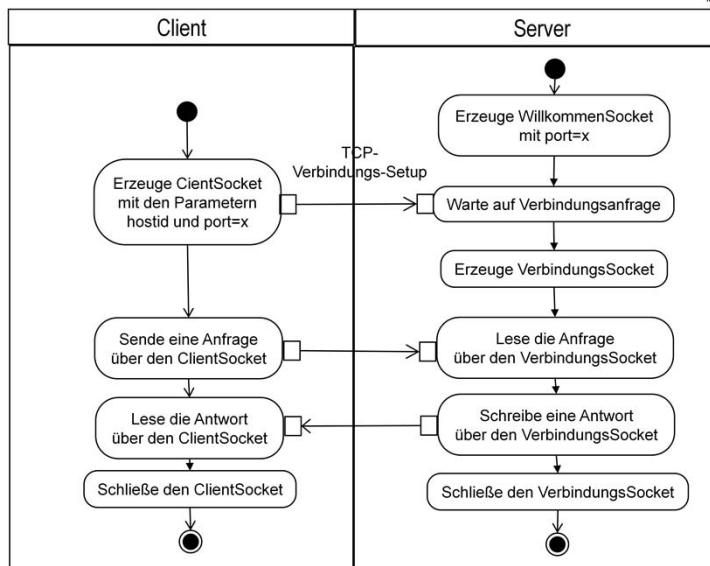
Cooperation & Management (C&M, Prof. Abeck)  
Institut für Telematik, Fakultät für Informatik

[KR03:133]

- (1) Ein definierter Socket, der sog. "WillkommenSocket", dient dem Client dazu, an die Tür des Server-Prozesses anzuklopfen.
- (2) Angabe der Adresse des Server-Prozesses (IP-Adresse) und der Portnummer.
- (3) Der 3-Wege-Handshake wird Client-seitig initiiert und ist vollständig transparent für den Client und den Server.
  - (i) Der Client-Prozess klopft am "WillkommenSocket" des Server-Prozesses an, woraufhin der Server einen "VerbindungsSocket" erzeugt.
  - (ii) Der Client-Prozess schickt eine Anfrage an den "VerbindungsSocket".
  - (iii) Der Server-Prozess schickt eine Antwort an den "ClientSocket".
- (4) Aus der Sicht der Anwendung stellt die TCP-Verbindung eine direkte virtuelle Pipe zwischen dem Client-Socket und dem Verbindungs-Socket dar.  
Es wird die Reihenfolge der übertragenen Bytes erhalten und Übertragungsfehler werden behandelt (zuverlässiger Bytestrom-Service).

[KR03] James F. Kurose, Keith W. Ross: Computer Networking – A Top-down Approach Featuring the Internet, 2nd Edition, Addison Wesley, 2003.

## Ablauf als Sequenzdiagramm



19 02.09.2011

WASA - 2-2 KOMMUNIKATION

Cooperation & Management (C&M, Prof. Abeck)  
Institut für Telematik, Fakultät für Informatik

[Ke06]

Der Ablauf wird in Form eines UML-Aktivitätsdiagramms [Ke06:215] dargestellt.

Der Client und der Server umfassen jeweils einen Aktivitätsbereich (wird auch als Verantwortungsbereich bezeichnet) [:219].

Die Pfeile mit offener Pfeilspitze geben den Kontrollfluss an und repräsentieren die Ausführungsreihenfolge der Aktivitäten, die sie verbinden [:218].

[Ke06] Christoph Kecher: UML 2.0 – Das umfassende Handbuch, Galileo Press, 2006.

- (1) Welcher Zusammenhang besteht zwischen einem Socket und
  - (1) einer Netzanwendung
  - (2) dem Betriebssystem
  - (3) einem Anwendungsentwickler
  - (4) einem Port
- (2) Welche Sockets werden beim Aufbau einer TCP-Verbindung benötigt?
- (3) Warum werden auf Server-Seite zwei Sockets benötigt?
- (4) Warum wird das Verfahren zum Aufbau einer TCP-Verbindung als 3-Wege-Handshake bezeichnet?

## Java-Sockets

- (1) Zur Socket-Programmierung stehen zwei Java-Klassen zur Verfügung
  - (1) Klasse "ServerSocket" zur Erzeugung des WillkommenSocket
  - (2) Klasse "Socket" zur Erzeugung des ClientSocket und VerbindungsSocket
- (2) ServerSocket
  - (1) Bei der Erzeugung kann der Port angegeben werden, den der Server bzgl. einer Verbindungsanfrage abhört

```
ServerSocket serverSocket = new ServerSocket(port);
```
- (3) Socket
  - (1) Wird vom Server zum Abhören von Verbindungsanfragen genutzt

```
Socket socket = serverSocket.accept();
```
  - (2) Wird vom Client zur Anfrage einer Verbindung genutzt

```
Socket socket = new Socket(serverName, port);
```

21

02.09.2011

WASA - 2-2 KOMMUNIKATION

Cooperation & Management (C&M, Prof. Abeck)  
Institut für Telematik, Fakultät für Informatik

[Li05:1092]

Zwei Arten von Sockets werden in Java unterstützt: (i) Stream-Sockets, die TCP nutzen und (ii) Datagramm-Sockets, die UDP nutzen

Aufgrund der zuverlässigen Übertragung werden Stream-Sockets in den meisten Gebieten der Java-Programmierung verwendet [:1092].

(1) Die Klassen sind im Paket "java.net" enthalten [:1097].

(2) Der "ServerSocket" bzw. das zu der Klasse erzeugte "serverSocket"-Objekt entspricht dem "WillkommenSocket".

(2.1) Der Port identifiziert den TCP-Service am Socket. Er ist ein Wert zwischen 0 und 65536, wobei die Portnummern 0 bis 1024 für privilegierte Services reserviert sind (z.B. Port 80 für Web-Server und Port 25 für E-Mail-Server).

Der Versuch, einen Server-Socket an einem bereits genutzten Port zu erzeugen, führt zur Ausnahme "java.net.BindException".

(3.1) Die von der "ServerSocket"-Klasse angebotene "accept()"-Methode führt dazu, dass der Server-Prozess solange wartet, bis sich ein Client mit dem Server-Socket verbindet.

Die von der "ServerSocket"-Klasse angebotene "accept()"-Methode entspricht dem Hören eines Anklopftens durch einen Client-Prozess (was der Client-Prozess durch das Erzeugen eines Socket-Objekts mit der entsprechenden Adresse (Host+Port) durchführt).

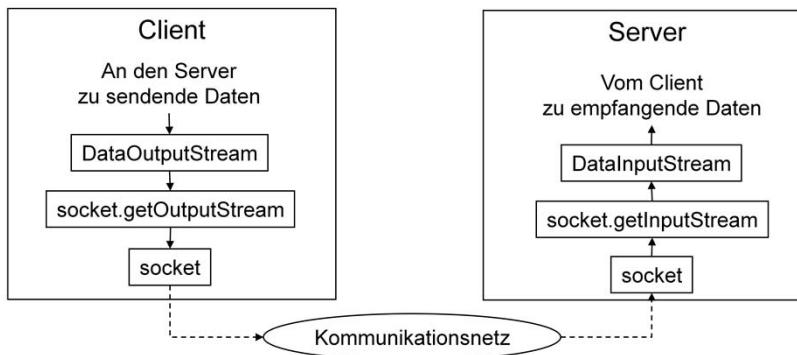
(3.2) Die Erzeugung eines Objekts der Klasse "Socket" öffnet einen Socket, so dass der Client mit dem Server kommunizieren kann.

"serverName" ist der Internet-Hostname des Servers oder die IP-Adresse [:1093].

[Li05] Y. Daniel Liang: Introduction to Java Programming; Companion Website: [www.prenhall.com/liang](http://www.prenhall.com/liang), Pearson Prentice Hall, 2005.



## Ein-Ausgabe-Ströme



- (1) Die Datenübertragung über Sockets erfolgt analog zur Ein-/Ausgabe
- (2) Daten werden zwischen Sockets aus Effizienzgründen grundsätzlich binär und nicht als Text übertragen

[Li05:1095]

(1) Aus der Sicht eines Java-Programms sind Sockets eine bestimmte Ausprägung eines Ein-Ausgabegeräts.

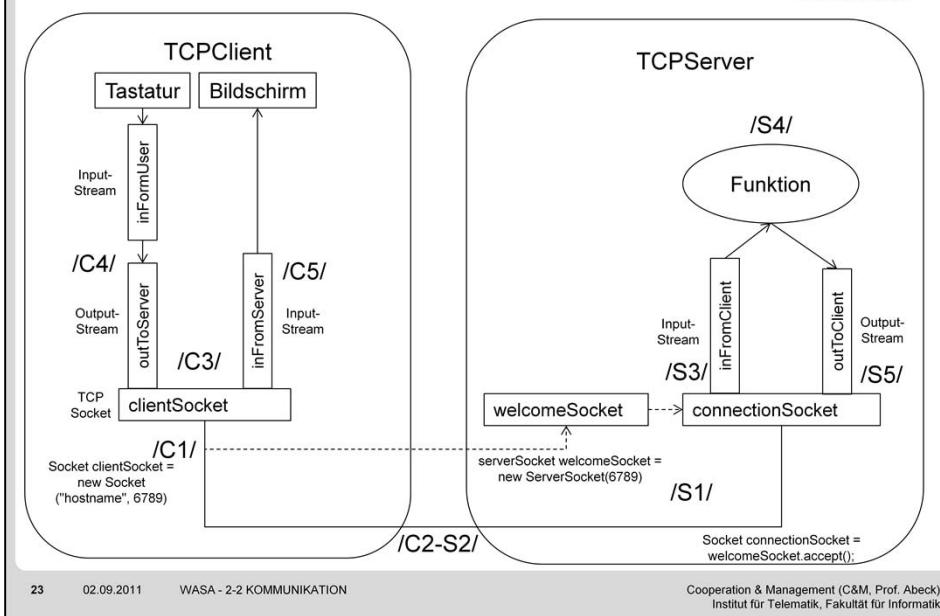
Die Übertragung von Daten in die andere Richtung, also vom Server zum Client, verläuft analog (d.h., der Server nutzt Output-Klassen und der Client nutzt Input-Klassen).

(2) Der Grund für die Ineffizienz ist die notwendige Kodierung und Dekodierung von Textdaten.

(DataOutputStream, DataInputStream) Zu den verschiedenen Arten von Streams: Durch "InputStream" und "OutputStream" lassen sich nur Bytes lesen und schreiben. Sollen Daten z.B. der Typen int, double oder String übertragen werden, bietet sich die Umwandlung in einen "DataInputStream" bzw. "DataOutputStream" an [:1094].

[Li05] Y. Daniel Liang: Introduction to Java Programming; Companion Website: [www.prenhall.com/liang](http://www.prenhall.com/liang), Pearson Prentice Hall, 2005.

## Auftretende Sockets und Ströme



[KR03:136]

In einer einfachen Java-basierten Client-Server-Anwendung besteht der durch die Abbildung skizzierte Zusammenhang zwischen Sockets und Streams.

Hinweis: Eine solche (minimale) Anwendung soll in der nächsten Praktischen Aufgabe implementiert werden.

(TCPClient, TCPServer) Stellen jeweils ein Java-Programme dar (z.B. "TCPServer.java" und "TCPClient.java") [:136].

(Funktion) Das ist die eigentliche Geschäftslogik der Anwendung.

Nachfolgend wird das Zusammenspiel der einzelnen Schritte auf Server-Seite /Sn/ und Client-Seite /Cn/ beschrieben:

/S1/ In der Initialisierungsphase werden auf der Server-Seite der "welcomeSocket" und der "connectionSocket" (mittels "accept()"-Methode) erzeugt.

Der Server-Programm (genauer: der "welcomeSocket") wartet auf das Anklopfen eines (beliebigen, zum jetzigen Zeitpunkt noch nicht bekannten) Client-Programms.

/C1/ Mit dem Erzeugen des "clientSocket" klopft der Client bei dem durch die bei der Erzeugung angegebenen Parameter (hostId, portnr) adressierten "welcomeSocket" an.

/C2-S2/ Es wird - für die beiden Prozesse transparent - eine TCP-Verbindung zwischen "clientSocket" und "connectionSocket" aufgebaut.

/C3/ Es werden ein Output-Stream "outToServer" und ein Input-Stream "inFromServer" an den "clientSocket" angebracht (engl. attach).

/C4/ Daten (Text) werden von der Tastatur eingelesen und über den "clientSocket" versendet.

/S3/ Die Daten werden über den "connectionSocket" empfangen und über einen an den Socket angebrachten Input-Stream einer Funktion zur Verarbeitung bereitgestellt.

/S4/ Die Daten werden durch eine Funktion verarbeitet.

/S5/ Die verarbeiteten Daten werden über einen Output-Stream und den "connectionSocket" an den "clientSocket" gesendet.

/C5/ Die Daten werden vom "clientSocket" über den angebrachten Input-Stream auf dem Bildschirm ausgegeben.

[KR03] James F. Kurose, Keith W. Ross: Computer Networking – A Top-down Approach Featuring the Internet, 2nd Edition, Addison Wesley, 2003.

## LZ TCP – ÜA JAVA-SOCKETS

- (1) Wie heißen das Java-Paket und darin enthaltene Java-Klassen zur Socket-Programmierung?
- (2) Wie wird der Zusammenhang zwischen den beiden Socket-Typen auf der Server-Seite hergestellt?
- (3) Welches Java- Konzept wird genutzt, um einem Socket zu sendende Daten zu übergeben bzw. von diesem empfangene Daten zu erhalten?

- (1) Es ist mittels Eclipse eine Client/Server-Anwendung zu schreiben, die auf Sockets beruht. Der Client schickt eine Tastatur-Eingabe über eine TCP-Verbindung an den Server. Der Server wandelt den Text in Großbuchstaben um und schickt ihn an den Client zurück, der den Text auf dem Monitor ausgibt.

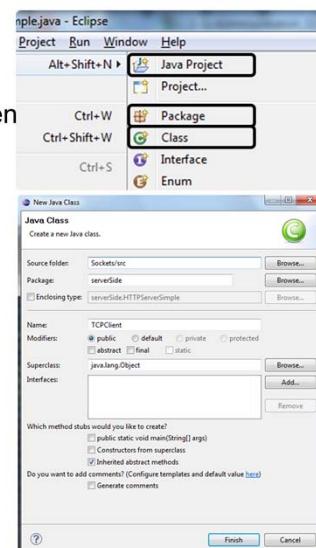
- (1) Die geforderte Anwendung erfordert keine über das Java Development Toolkit (JDT, Teil des Standard-Eclipse) hinausgehenden Plug-ins.

JDT

Java Development Toolkit

## PA SOCKETPROGRAMMIERUNG – Anlegen eines neuen Projekts

- (1) File → New
  - (1) Java-Projekt mit dem Namen "2-2\_SOCKETPROGRAMMIERUNG" anlegen
- (2) Pakete anlegen mit Rechtsklick auf das Projekt
  - (1) Zwei Pakete (Package) "clientSide" und "serverSide" anlegen
- (3) Klassen erstellen
  - (1) Klasse "TCPClient" im "clientSide"-Paket
  - (2) Klasse TCPServer im "serverSide"-Paket



(1) Für eine bessere Übersicht im Eclipse-Workspace ist für jede Praktische Aufgabe ein neues Projekt anzulegen.

Die Konvention für Projektnamen lautet:

<Nummer der WASA-Kurseinheit\_<Name der Praktischen Aufgabe>

(2) Die Programme in Java werden als Mengen von Paketen organisiert. Jedes Paket hat seine eigene Reihe von Namen für Typen (z.B. Klassen), um Namenskonflikte zu vermeiden. Durch Pakete lassen sich Klassen organisieren, die der gleichen Kategorie angehören oder eine zusammengehörige Funktionalität haben. Ein Paket kann in einem Dateisystem abgelegt werden (als Ordner) oder in einer Datenbank. Pakete werden in der Regel mit Hilfe eines hierarchischen Benennungsmusters definiert, dessen Hierarchieebenen durch Punkte (.) (sprich "dot") getrennt sind [JLS-C7].

(2.1) Die Benutzung des Standardpaketes ist nicht zu empfehlen, deshalb ist eine gute Herangehensweise das Erstellen von separaten Paketen, eines für die Client - und ein anderes für die Server-Seite. Da dies eine verteilte Anwendung wird, wirkt die Einteilung in Pakete natürlich.

(3.1) Im "clientSide"-Paket liegt der Code, der auf der Seite des Client ausgeführt wird. Dieser Code dient zur Erfassung der Eingaben des Benutzers, deren eventuellen Validierung und dem Anzeigen der Antwort vom Server.

(3.2) Im "serverSide"-Paket liegt der Code, der auf der Seite des Servers ausgeführt wird. Dieser Code dient zur Bearbeitung und Beantwortung der Anfragen vom Client.

[JLS-C7] Java Language Specification Chapter 7, [http://java.sun.com/docs/books/jls/third\\_edition/html/packages.html](http://java.sun.com/docs/books/jls/third_edition/html/packages.html)

## PA SOCKETPROGRAMMIERUNG – Client-Programm "TCPClient.java" (1)

```
1.  Package clientSide;
2.  import java.io.*;
3.  import java.net.*;
4.  class TCPClient {
5.  public static void main(String argv[]) throws Exception {
6.  String sentence;
7.  String modifiedSentence;
8.  BufferedReader inFromUser =
9.  new BufferedReader(new InputStreamReader(System.in));
10. Socket clientSocket = new Socket("localhost", 8000);
11. DataOutputStream outToServer =
12. new DataOutputStream(clientSocket.getOutputStream());
13. BufferedReader inFromServer =
14. new BufferedReader
15. (new InputStreamReader(clientSocket.getInputStream()));
16. sentence = inFromUser.readLine();
17. outToServer.writeBytes(sentence + '\n');
18. modifiedSentence = inFromServer.readLine();
19. System.out.println("Antwort vom Server: " + modifiedSentence);
20. clientSocket.close();
21. } //main
22. } // TCPClient
```

27

02.09.2011 WASA - 2-2 KOMMUNIKATION

Cooperation & Management (C&M, Prof. Abeck)  
Institut für Telematik, Fakultät für Informatik

[KR03:138] bzw. [KR10:191]

Dieser Quelltext zeigt ein Client-Programm, welches die Eingabe des Benutzers von der Konsole liest, an den Server schickt und anschließend die Antwort des Servers auf der Konsole ausgibt.

(Package clientSide 1.) Deklaration des Paketes, zu dem die Klasse gehört.

(import java.io 2.) Das "java.io"-Paket enthält Klassen für Eingabe- und Ausgabe-Ströme. Hierzu gehören auch die im Programm verwendeten Klassen "BufferedReader" und "DataOutputStream" [:138].

(import java.net 3.) Das "java.net"-Paket stellt Klassen für die Netz-Unterstützung bereit, wozu auch die Klassen "Socket" und "ServerSocket" gehören.

(class ... 4.) Anfang des die Klasse "TCPClient" definierenden Blocks. Der Block wird durch eine geschweifte Klammer (engl. curly bracket) geöffnet und enthält Variablen und Methoden.

(public ... 5.) Die Klasse "TCPClient" besitzt keine Klassenvariablen und genau eine Methode, die "main()"-Methode.

"main()" ist die Methode, die vom Java-Interpreter aufgerufen wird, wenn die Anwendung gestartet wird (analog zur "main"-Funktion in C bzw. C++).

(string sentence 6.) Das Objekt "sentence" ist der vom Benutzer eingegebene String, der zum Server gesendet wird.

(string modifiedSentence 7.) Das Objekt "modifiedSentence" ist der String, der vom Server erhalten wird und an die Standard-Ausgabe (Bildschirm) des Benutzers gesendet wird.

(BufferedReader ... 8., 9.) Erzeugung des Stream-Objekts "inFromUser" vom Typ "BufferedReader".

Der Eingabestrom wird durch "System.in" initialisiert, wodurch der Strom "inFromUser" an die Standard-Eingabe angehängt wird.

[KR03] James F. Kurose, Keith W. Ross: Computer Networking – A Top-down Approach Featuring the Internet, 2nd Edition, Addison Wesley, 2003.

[KR10] James F. Kurose, Keith W. Ross: Computer Networking – A Top-down Approach, 5<sup>th</sup> Edition, Pearson, 2010.

[Li05] Y. Daniel Liang: Introduction to Java Programming; Companion Website: [www.prenhall.com/liang](http://www.prenhall.com/liang), Pearson Prentice

Hall, 2005.

## PA SOCKETPROGRAMMIERUNG – Client-Programm "TCPClient.java" (2)

```
1.  Package clientSide;
2.  import java.io.*;
3.  import java.net.*;
4.  class TCPClient {
5.  public static void main(String argv[]) throws Exception {
6.  String sentence;
7.  String modifiedSentence;
8.  BufferedReader inFromUser =
9.  new BufferedReader(new InputStreamReader(System.in));
10. Socket clientSocket = new Socket("localhost", 8000);
11. DataOutputStream outToServer =
12. new DataOutputStream(clientSocket.getOutputStream());
13. BufferedReader inFromServer =
14. new BufferedReader
15. (new InputStreamReader(clientSocket.getInputStream()));
16. sentence = inFromUser.readLine();
17. outToServer.writeBytes(sentence + '\n');
18. modifiedSentence = inFromServer.readLine();
19. System.out.println("Antwort vom Server: " + modifiedSentence);
20. clientSocket.close();
21. } //main
22. } //TCPClient
```

28 02.09.2011

WASA - 2-2 KOMMUNIKATION

Cooperation & Management (C&M, Prof. Abeck)  
Institut für Telematik, Fakultät für Informatik

[KR03:138] bzw. [KR10:191]

(Socket clientSocket ... 10.) Erzeugung des Objekts "clientSocket" vom Typ "Socket" durch Aufruf einer Konstruktor-Methode der Klasse "Socket".

Gleichzeitig wird die TCP-Verbindung zwischen Client und Server initiiert.

Hinweis: in [KR03:138] ist der String "hostname" angegeben, der gemäß einer Erklärung in [:140] zu ersetzen ist gegen den Hostnamen oder die IP-Adresse des Servers.

Hier wird im Beispielprogramm "localhost" verwendet, was ausdrückt, dass sich der Server auf dem gleichen Rechner befindet wie der Client [Li:1098].

(DataOutputStream outToServer = ... 11.-15.) Durch diese Anweisungen werden die Stream-Objekte "outToServer" und "inFromServer" erzeugt, die an den Socket "clientSocket" angehängt werden.

(sentence = ... 16.) Die vom Benutzer eingegebene, durch ein "\n" (Line Feed, LF) abgeschlossene Zeile (engl. Line) wird von der Standardeingabe durch den Strom "inFromUser" dem String "sentence" zugewiesen.

(outToServer... 17.) Der um ein "\n" (LF) erweiterte String "sentence" fließt in den Strom "outToServer". Da dieser Strom an den Socket "clientSocket" gebunden ist, fließt er durch den Socket des Clients in die TCP-Pipe.

Der Client wartet dann, um Zeichen vom Server zu erhalten.

(modifiedSentence 18.) Die vom Server gesendeten Zeichen fließen durch den Strom "inFromServer" und werden in den String "modifiedSentence" gesetzt, bis die Zeile mit einem LF endet.

(System.out.println(...) 19.) Der String "modifiedSentence" wird zum Ausdruck an die Standard-Ausgabe (Bildschirm) geleitet.

(clientSocket.close() 20.) Schließen des Sockets und damit der TCP-Verbindung.

[KR03] James F. Kurose, Keith W. Ross: Computer Networking – A Top-down Approach Featuring the Internet, 2nd Edition, Addison Wesley, 2003.

[KR10] James F. Kurose, Keith W. Ross: Computer Networking – A Top-down Approach, 5<sup>th</sup> Edition, Pearson, 2010.

[Li05] Y. Daniel Liang: Introduction to Java Programming; Companion Website: [www.prenhall.com/liang](http://www.prenhall.com/liang), Pearson Prentice Hall, 2005.

## PA SOCKETPROGRAMMIERUNG – Server-Programm "TCPserver.java"

```
1. package serverSide;  
2. import java.io.*;  
3. import java.net.*;  
4.  
5. class TCPserver {  
6. public static void main(String argv[]) throws Exception {  
7. String clientSentence;  
8. String capitalizedSentence;  
9. ServerSocket welcomeSocket = new ServerSocket(8000);  
10. while(true) {  
11. Socket connectionSocket = welcomeSocket.accept();  
12. BufferedReader inFromClient =  
13. new BufferedReader  
14. (new InputStreamReader(connectionSocket.getInputStream()));  
15. DataOutputStream outToClient =  
16. new DataOutputStream(  
17. connectionSocket.getOutputStream());  
18. clientSentence = inFromClient.readLine();  
19. capitalizedSentence = clientSentence.toUpperCase() + '\n';  
20. outToClient.writeBytes(capitalizedSentence);  
21. connectionSocket.close(); } //while - listen loop  
22. } //main  
22. } //TCPserver
```

[KR03:141] bzw. [KR10:194]

Dieser Quelltext zeigt ein Server-Programm, welches eine Anfrage von einem Client entgegennimmt, eine Ausgabe ermittelt und sie dann dem Client als Antwort zurückschickt.

Das Programm "TCPserver" hat viele Ähnlichkeiten mit dem Programm "TCPClient". Zeilen, die identisch sind, werden hier nicht mehr kommentiert.

(ServerSocket welcomeSocket ... 8.) An dieser Stelle ist "TCPserver" fundamental verschieden zu "TCPClient", da hier der Welcome-Socket, also die Tür erzeugt wird, an dem ein beliebiges Client-Programm durch Angabe des Servers und der Portnummer anklopfen kann. Der Welcome-Socket (d.h. das Objekt "welcomeSocket" der Klasse "ServerSocket") hört(lauscht) (listen) am Port 8000.

(Socket connectionSocket = ... 10.) Durch die in der Klasse "ServerSocket" angebotene "accept()"-Methode wird ein neuer Socket "connectionSocket" erzeugt mit der gleichen Portnummer (8000) wie der Welcome-Socket.

TCP erzeugt dann (mittels eines aus der Sicht der beiden Programm transparenten 3-Wege-Handshake-Verfahrens) eine direkte virtuelle Pipe zwischen dem "clientSocket" auf der Client-Seite und dem "connectionSocket" auf der Server-Seite.

Der "welcomeSocket" hört weiterhin, ob andere Clients einen Verbindungsaufbau wünschen. In diesem Fall würden Threads erzeugt, wodurch beliebig viele TCP-Verbindungen von dem Server parallel über den Port bedient werden könnten.

(BufferedReader ... 11.-16.) Erzeugung mehrerer Stream-Objekte analog zu den Stream-Objekten, die zum "clientSocket" erzeugt wurden.

(capitalizedSentence = ... 18.) Diese Zeile ist das Herz (d.h. die eigentliche Funktionalität oder Geschäftslogik) der Anwendung, da es die Großschreibung des Satzes vornimmt.

Alle übrigen Befehle des Programms dienen ausschließlich der Kommunikation mit dem Client.

(connectionSocket.close() 20.) Der "connectionSocket" wird geschlossen da die Anfrage abgearbeitet, und beantwortet wurde. Der "welcomeSocket" ist immer noch offen, so dass der Server weitere Clients akzeptieren kann.

[KR03] James F. Kurose, Keith W. Ross: Computer Networking – A Top-down Approach Featuring the Internet, 2nd Edition, Addison Wesley, 2003.

[KR10] James F. Kurose, Keith W. Ross: Computer Networking – A Top-down Approach, 5<sup>th</sup> Edition, Pearson, 2010.

## PA SOCKETPROGRAMMIERUNG – Starten der Anwendung aus Eclipse

- (1) Mit Rechtsklick auf die Klassen, die die "main"-Methode besitzen

- (1) Run As → Java Application
- (2) Den Server zuerst starten
- (3) Beliebige Clients starten

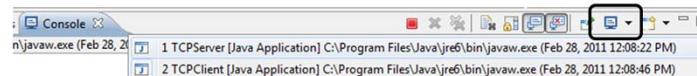
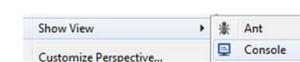


- (2) Im Konsolenfenster des Client den Text eingeben und "Enter" drücken

- (3) Die Antwort vom Server wird im Konsolenfenster des Client angezeigt

```
<terminated> TCPClient [Java Application] C:\Program Files  
Hallo Welt!  
Antwort vom Server: HALLO WELT!
```

- (4) Anzeigen der Konsole über Window → Show View → Console



30 02.09.2011 WASA - 2-2 KOMMUNIKATION

Cooperation & Management (C&M, Prof. Abeck)  
Institut für Telematik, Fakultät für Informatik

Es ist zu beachten, dass die Ausnahmen (engl. exceptions), die auftreten können, nicht von dem hier aufgeführten Code behandelt werden. Ausnahmen treten z.B. in den folgenden Fällen auf:

- (i) Verbindungsversuch des Client zu einem nicht laufenden Server
- (ii) Unerwartete Unterbrechung der Verbindung.

Wenn dies der Fall ist, wird eine Fehlermeldung (in roter Schrift) ausgegeben und das Programm bricht ab. Die Anwendung kann nach einem Absturz wieder normal gestartet werden.

(1) Eclipse führt die Übersetzung und das Ausführen von Java-Klassen transparent für den Benutzer durch. Bei jedem Speichern in Eclipse wird die Klasse neu kompiliert. Der Java-Byte-Code, der daraus entsteht, wird in die \*.class Dateien im Ordner "<Workspace>\<Projektnamen>\bin\" gespeichert.

(1.2) Damit der Client eine Verbindung aufbauen kann, muss der Server zuerst laufen und am entsprechenden Port lauschen.

(1.3) Da der Server in einer Dauerschleife lauscht, können mehrere Client darauf zugreifen, bis der Server beendet wird.

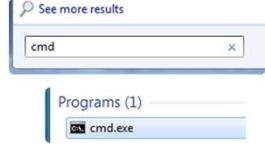
(2) Es kann ein beliebiger Text eingegeben werden.

(3) Vom Server wird der eingegebene Text in Großbuchstaben ausgegeben.

(4) Die Konsole erscheint standardmäßig in der unteren Sicht. Jedes ausgeführte Java-Programm bekommt sein eigenes Konsolenfenster, das man in der Konsolensicht durch Klicken auf dem blauen Bildschirmsymbol auswählen kann (das Bildschirmsymbol ist in dem oberen Screenshot mit einem schwarzen Kästchen markiert).

## PA SOCKETPROGRAMMIERUNG – Windows-Eingabeaufforderung

- (1) Eingabeaufforderung (cmd) starten
  - (1) Start → cmd eingeben
- (2) Ausführung des Client
  - (1) Der Server muss beim Starten des Client bereits laufen
  - (2) In der Eingabeaufforderung zu "<Workspace>\<Projekt-Name>\src" navigieren
  - (3) Kompilieren des Client mit „javac“
  - (4) Starten des Client mit „java“
- (3) Wenn die Java-Befehle nicht erkannt werden, muss die Path-Variable gesetzt werden



```
C:\Users\Andrei\workspaceBA\Sockets\src>javac clientSide/TCPClient.java
C:\Users\Andrei\workspaceBA\Sockets\src>java clientSide/TCPClient
Hallo Welt!
Antwort vom Server: HALLO WELT!
C:\Users\Andrei\workspaceBA\Sockets\src>
```

(1) Die Windows-Eingabeaufforderung bietet die Möglichkeit, die Java-Funktionalitäten des JDK aufzurufen. Um die Eingabeaufforderung zu starten, kann man (ab Windows Vista) im Suchfeld des Start-Fensters (Windows-Symbol) das Wort "cmd" eingeben. Eine andere Möglichkeit, die in älteren Versionen von Windows verwendet werden kann, ist durch den "Ausführen"-Dialog gegeben (Start -> Ausführen oder Windows-Taste + R). Auch hier wird mit der Eingabe des Wortes "cmd" die Eingabeaufforderung gestartet. Bei Linux und OS X sind die Java Funktionalitäten durch den Terminal aufzurufen.

(2) Die Verbindung zum Server wird wie zuvor aufgebaut, auch wenn der Client aus einer anderen Umgebung gestartet wird.

(2.1) Der Server muss laufen, damit sich der Client verbinden und die Anfrage schicken kann. Der Server kann entweder in Eclipse gestartet werden oder aus der Eingabeaufforderung (analoges Vorgehen wie beim Client).

(2.2) Im "src"-Ordner (sources) befinden sich die ".java"-Dateien. Diese sind dieselben Dateien, die unsere Klassen beinhalten und wir in Eclipse bearbeiten. Die Navigation in der Eingabeaufforderung wird mit dem Befehl "cd" (change directory) veranlasst, z.B.: "cd myWorkspace\Sockets"

(2.3) Mit dem Befehl "javac" (Java Compile) wird aus den Klassen, dem Java-Code, Bytecode generiert, der in "\*.class"-Dateien, mit dem Namen der Klasse abgelegt werden. Die Struktur des "javac"-Befehls lautet "javac <Klassename.java>". Da verschiedene Pakete bestehen, muss man beim Kompilieren und Ausführen auf die entsprechenden Pakete verweisen. Ein Aufruf sieht dann folgendermaßen aus: "javac <Paket>/<Klassename>.java"

(2.4) Mit dem Befehl "java" wird der Bytecode, der in den "\*.class"-Dateien gespeichert ist, von der Java Virtual Machine (JVM) ausgeführt. Die Struktur des "java"-Befehls lautet "java <Klassename>". Hier ist zu beachten, dass die Endung „.class“ weggelassen wird.

Man kann auch die von Eclipse kompilierten .class"-Dateien mit dem "java"-Kommando starten, dazu muss man in der Eingabeaufforderung zum Ordner "<Workspace>\<Projekt-Name>\bin" (Binaries) navigieren und sie von dort aus starten.

(3) Setzen der Umgebungsvariable "PATH"; in Windows 7 mittels "Start -> Systemsteuerung -> System und Sicherheit -> System -> Erweiterte Systemeinstellungen -> Umgebungsvariablen,). Das Verzeichnis der Java-Installation(z.B. "C:\Programme\Java\jdk1.6.0\_22\bin") ist an den existierenden Wert anzuhängen.

## Folie 31

---

AM1 java Befehl notizen?

Durch Befehle direkt in der Konsole(achtung - nur bei einer Session)

Andrei Miclaus; 03.05.2011

- (1) Erst das (World Wide) Web öffnete das Internet auch für Nutzer, die nicht im akademischen Bereich tätig sind
  - (1) Bis zu diesem Zeitpunkt (Anfang der 90er Jahre) war das Internet nur eines von vielen vorhandenen Datennetzen
- (2) Das Web ist der dritte große Schritt im Bereich der elektronischen Kommunikation nach Telefon (1870) und Radio/Fernsehen (1920)
- (3) Das HyperText Transfer Protocol (HTTP) bildet das Herz des Web
  - (1) Implementiert in einem Client-Programm und einem Server-Programm
- (4) Eine Web-Seite besteht aus Objekten
  - (1) Jedes Objekt wird durch eine einzelne URL adressiert
  - (2) Beispiele für Objekte: HTML-Datei, JPEG-Bild, Java-Applet
  - (3) Meist eine Basis-HTML-Datei und mehrere referenzierte Objekte

[KR03:88]

- (1) Bis Anfang der 90er Jahre wurde das Internet fast ausschließlich im universitären Bereich zum Zugriff auf entfernte Rechner und zum Austausch von Mails genutzt.
  - (1.1) Beispiele anderer Netze sind: Prodigy, America Online, CompServe, Minitel/Transpac in Frankreich, X.25 Datex-P-Netz in Deutschland
- (2) Eine besondere Attraktivität des Web im Vergleich zum Fernsehen besteht darin, dass es auf Anfrage (engl. on demand) arbeitet [:89].
- (3) HTTP ist in den zwei RFCs 1945 und 2616 definiert.
  - (3.1) Diese Programme laufen auf unterschiedlichen Systemen und tauschen HTTP-Nachrichten aus.
- (4) Eine Web-Seite wird auch als Dokument bezeichnet. Ein Objekt ist einfach eine Datei.
  - (4.1) Uniform Resource Locator
  - (4.2) Auch Bilder in anderen Formaten wie z.B. GIF, Audio/Video-Clip
  - (4.3) Z.B. einen HTML-Text (das ist die Basis-HTML-Datei), in dem 5 Bilder auftreten, womit die Webseite aus 6 Objekten besteht.

[KR03] James F. Kurose, Keith W. Ross: Computer Networking – A Top-down Approach Featuring the Internet, 2nd Edition, Addison Wesley, 2003.

- (1) Wesentliche Eigenschaften
  - (1) ASCII-Anwendungsprotokoll
  - (2) Setzt auf eine (sichere) TCP-Verbindung auf
  - (3) Default-Port: 80
  - (4) Unterstützt nicht-persistente und persistente Verbindungen
- (2) Wichtige HTTP-Operationen
  - (1) GET: Anfordern eines bestimmten Dokuments
  - (2) HEAD: Anfordern von Informationen über ein Dokument
  - (3) POST: Senden von Daten für die weitere Bearbeitung durch den Server

Wie alle Internet-Protokolle ist auch das HTTP-Protokoll in einem Request for Comments (RFC1945, 2068) beschrieben.

(1.1) ASCII-Protokoll besagt konkret, dass die Anfragen durch den Client und die Antworten durch den Server als ASCII-Strings über das Netz übertragen werden. D.h., eine Beispiel-Anfrage "GET /anschrift/inf.htm" wird als ASCII-Zeichenfolge übergeben, woraufhin der Server den Inhalt dieser Seite ebenfalls als ASCII-Zeichenfolge überträgt.

(1.2) Bevor wir näher auf die Befehle eingehen, sollen zunächst die wesentlichen Eigenschaften des Protokolls beschrieben werden.

(1.3) HTTP setzt auf eine sichere TCP-Verbindung auf, wobei der HTTP-Server auf Port 80 hört, ob eine Anfrage eines HTTP-Clients vorliegt.

(1.4) Nicht-persistent bedeutet, dass der HTTP-Server die Verbindung nach einer Anfrage sofort schließt. In diesem Fall ist HTTP statuslos, was die Entwicklung von HTTP-Clients und HTTP-Server vereinfacht, aber ein ineffizientes Protokollverhalten bewirkt.

Ab der Version HTTP/1.1 werden persistente Verbindungen unterstützt, wie später noch ausführlicher erklärt wird.

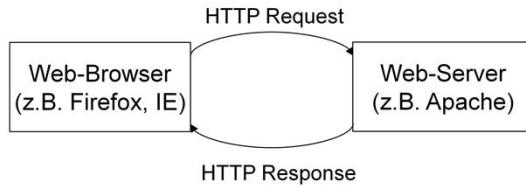
(2) Die von HTTP unterstützten Befehle können als Methoden auf dem Objekt "Web-Seite" angesehen werden.

(2.1) In den Versionen vor 1.0 gab es ausschließlich den Befehl GET, der daher auch gar nicht angegeben werden musste. Weitere Befehle neben GET sind:

(2.2) Statt der ganzen Seite kann durch den Befehl HEAD auch nur die Steuerinformation einer Web-Seite abgefragt werden.

(2.3) Ein Beispiel, wann der POST-Befehl vom Client benötigt wird, ist das Senden von der von einem Benutzer über ein Formular eingegebenen Information an den Server. Daneben kann durch den Befehl PUT auch eine ganze Web-Seite im Server gespeichert werden, sofern der Client die Rechte dazu besitzt. Der zu PUT entgegengesetzte HTTP-Befehl ist DELETE, durch den die als URL angegebene Seite gelöscht werden soll.

## Anfrage-Antwort-Prinzip von HTTP



- (1) Die Anfrage ist eine HTTP-Methode
  - (1) Ein Parameter ist die Seite, auf die zugegriffen werden soll
  - (2) Eventuell zusätzliche Parameter zur
- (2) Die Antwort liefert das Ergebnis zur Anfrage
  - (1) Status-Code
  - (2) Inhalte-Typ
  - (3) Inhalt

[BS+08:10]

Die Struktur einer HTTP-Konversation entspricht dem Anfrage-Antwort-Prinzip (engl. request-response), wobei der Transport einer vollständigen Anfrage bzw. Antwort zwischen den beteiligten Systemen durch TCP/IP in Form eines "Request Stream" bzw. "Response Stream" durchgeführt wird [:10].

(1) Die Methode beschreibt die durchzuführende Aktion.

Hinweis: Anstelle von HTTP-Methoden wird in der Literatur teilweise auch der Begriff der HTTP-Operationen verwendet.

(1.1) Die Seite wird durch die Angabe des Uniform Resource Locator (URL) bestimmt.

(1.2) Die Parameter entsprechen den Argumenten zu der Methode.

Neben POST unterstützt auch GET im eingeschränkten Umfang die Übertragung zusätzlicher Parameter, wie später verdeutlicht wird.

(2) Die Antwort besteht aus dem Inhalt, der HTML sein kann, und zusätzlicher Steuerinformation, die den HTTP-Header bildet.

(2.1) Gibt an, ob die Anfrage erfolgreich war oder nicht.

(2.2) Beispiele sind reiner Text, ein Bild oder HTML.

URL                    Uniform Resource Locator

[BS+08] Bryan Basham, Kathy Sierra, Bert Bates: Head First Servlets & JSP, Second Edition, O'Reilly, 2008.

## Nicht-persistente Verbindung

- (1) Eine nicht-persistente HTTP-Verbindung ermöglicht die Übertragung nur eines einzigen Web-Objekts über eine TCP-Verbindung
  - (1) Wird in HTTP/1.0 verwendet
- (2) Ablauf bei der Abfrage einer Web-Seite mit einem Basis-HTML-Datei und 10 JPEG-Bildern
  - (1) HTTP-Client initiiert eine TCP-Verbindung zu dem Web-Server
  - (2) HTTP-Client sendet eine HTTP-Anfrage-Nachricht an den Server
  - (3) HTTP-Server empfängt die Anfrage-Nachricht, ruft das Objekt ab und kapselt es in einem Objekt in einer HTTP-Antwort-Nachricht
  - (4) HTTP-Server fordert TCP zum Abbau der Transportverbindung auf
  - (5) HTTP-Client erhält die Antwort-Nachricht und extrahiert das HTML-Datei, in der die Referenzen auf die 10 JPEG-Bilder enthalten sind
  - (6) Wiederholung der Schritte (1) bis (4) für jedes JPEG-Bild

[KR03:91]

Auf die Eigenschaften von persistenten HTTP-Verbindungen wird später näher eingegangen.

(1) Nicht-persistent heißt also, dass die Verbindung über die Übertragung dieses einen Objekts nicht dauerhaft bestehen bleibt [:91].

(2) Die Web-Seite besteht also aus insgesamt 11 Objekten. Der URL sei "www.kit.edu/cm.tm/home.index"

(2.1) Angabe der Adresse bzw. des (durch DNS aufzulösenden) Namens des Web-Servers ("www.kit.edu") und der Portnummer 80.

(2.2) Die Übertragung der Nachricht erfolgt über den Socket, der mit der TCP-Verbindung assoziiert ist [:92].

Die Anfrage-Nachricht beinhaltet den Pfadnamen der Basis-HTML-Datei ("/cm.tm/home.index").

(2.3) Das Objekt liegt in einem Speicher (Platte oder RAM) auf dem Server.

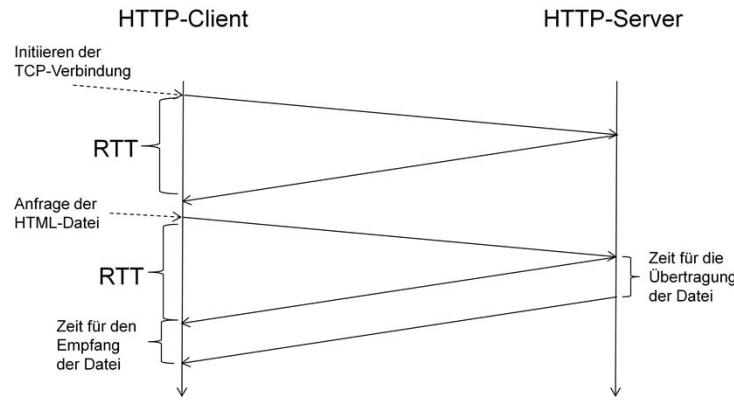
(2.4) Die TCP-Instanzen bauen die Verbindung erst dann ab, wenn sie sicher wissen, dass der Web-Client die Antwort-Nachricht korrekt erhalten hat.

(2.5) An dieser Stelle terminiert die TCP-Verbindung.

(2.6) Verschiedene Browser ermöglichen den Aufbau von parallelen TCP-Verbindungen, so dass dieser Vorgang nicht notwendigerweise streng sequentiell erfolgen muss.

[KR03] James F. Kurose, Keith W. Ross: Computer Networking – A Top-down Approach Featuring the Internet, 2nd Edition, Addison Wesley, 2003.

## Abschätzung der für die Anfrage einer HTML-Datei benötigten Zeit



- (1) Die Umlaufzeit (engl. round trip time, RTT) ist die Zeit, die ein kleines Paket vom Client zum Server und wieder zurück benötigt

[KR03:92]

Bei der betrachteten Datei handelt es sich z.B. um die Basis-HTML-Datei des auf der vorhergehenden Seite beschriebenen Ablaufs.

(1) Zur Abschätzung der Anfrage-Zeit wird die Umlaufzeit (engl. round trip time, RTT) benötigt. Die RTT beinhaltet (i) die Verzögerungszeit aufgrund der Paket-Verbreitung (engl. propagation), (ii) Zeit, die das Paket in Warteschlangen (engl. queuing) in Zwischensystemen verbringt, (iii) Paketverarbeitungs-Verzögerungen (engl. processing delays).

(Initiieren der TCP-Verbindung) Der Aufbau erfolgt durch den 3-Wege-Handshake.

Die erste Übertragung vom HTTP-Client zum HTTP-Server ist der Verbindungsauftaktwunsch (ein kleines TCP-Segment), der von der Server-Seite mit einem kleinen TCP-Segment beantwortet wird. Diese ersten zwei Teile des 3-Wege-Handshake-Verfahrens erfordert eine RTT.

(Anfrage der HTML-Seite) Das Senden der HTTP-Anfrage-Nachricht wird mit dem dritten Teil des 3-Wege-Handshake kombiniert [:93].

Der HTTP-Server sendet aufgrund der Anfrage die gewünschte HTML-Datei, was insgesamt (Anfrage und Antwort) eine zweite RTT dauert.

(Zeit für die Übertragung der Datei) Neben den zwei RTTs ist diese am Server entstehende Zeitdauer in die Berechnung der Gesamtzeit einzubeziehen.

RTT                  Round Trip Time

[KR03] James F. Kurose, Keith W. Ross: Computer Networking – A Top-down Approach Featuring the Internet, 2nd Edition, Addison Wesley, 2003.

- (1) Nachteil der nicht-persistenten Verbindung ist der hohe Ressourcen- und Zeitverbrauch
- (2) Eine persistente Verbindung ermöglicht die Übertragung beliebig vieler Objekte und damit auch beliebig vieler Web-Seiten über eine TCP-Verbindung
- (3) Effizienzsteigerung durch Pipelining
  - (1) Es können Anfragen vor Eintreffen der Antwort eines vorher angefragten Objekts an den Server geschickt werden
  - (2) Die TCP-Verbindung wird hierdurch besser ausgelastet

[KR03:93]

(1) Es wird für jedes anzufragende Objekt eine TCP-Verbindung aufgebaut.

- Ressourcenverbrauch: Allokation von TCP-Puffern und Halten von TCP-Variablen in den Clients und insbesondere in den Servern, was bei hunderten von simultanen Anfragen im Server zu gravierenden Engpässen führen kann.

- Zeitverbrauch: Eine zusätzliche RTT aufgrund der jeweils immer wieder neu aufzubauenden TCP-Verbindung.

(2) Die TCP-Verbindung wird üblicherweise nach einer (konfigurierbaren) Zeitdauer, in der keine Aktivität herrscht, geschlossen.

(3) Eine persistente Verbindung mit Pipelining (sinngemäß Parallelverarbeitung) ist der voreingestellte Modus in HTTP/1.1.

(3.1) Ohne Pipelining muss der Client die Anfragen streng sequentiell verschicken, da er grundsätzlich zuerst auf die Antwort der aktuellen Objektanfrage warten muss.

(3.2) Weil die gepipelinetete TCP-Verbindung für eine kürzeren Teil der Zeit frei (engl. idle) bleibt.

[KR03] James F. Kurose, Keith W. Ross: Computer Networking – A Top-down Approach Featuring the Internet, 2nd Edition, Addison Wesley, 2003.

- (1) Woraus besteht eine Web-Seite?
- (2) Was ist ein Basis-HTML-Datei?
- (3) Es sind einige wichtige Eigenschaften von HTTP zu nennen
- (4) Was ist eine nicht-persistente HTTP-Verbindung?
- (5) Wer initiiert den Aufbau und den Abbau der TCP-Verbindung bei einer HTTP-Anfrage?
- (6) Aus welchen Zeiten setzt sich die Dauer einer Anfrage einer HTML-Datei zusammen?
- (7) Welcher Zeitgewinn wird durch eine persistente HTTP-Verbindung erzielt?

## Überblick über die von HTTP/1.1 unterstützten Methoden

### (1) GET

- (1) Abfrage des Inhalts eines Objekts
- (2) Beschränkte Übertragung von zusätzlichen Informationen

### (2) POST

- (1) Zusätzlich zur Objektabfrage können umfangreiche und gesicherte Informationen geschickt werden

### (3) HEAD

- (1) Abfrage der Kopfinformation zu einem Objekt

### (4) PUT

- (1) Hochladen eines Objekts

### (5) DELETE

- (1) Löschen eines Objekts

[KR03:96]

Die Vorgänger-Spezifikation HTTP/1.0 hat lediglich die ersten drei genannten Methoden GET, POST und HEAD unterstützt.

(1) GET ist die bei weitem am häufigsten verwendete HTTP-Operation.

(1.1) Durch Angabe des URL und des Servers, auf dem das Objekt liegt.

(1.2) Die zusätzliche Information wird bei einer GET-Anfrage und GET-Antwort durch eine Ergänzung des URL übertragen.

(2) POST wird häufig zum Ausfüllen eines Formulars genutzt.

(2.1) Die Anfrage (engl. request message) geht vom Client zum Server, d.h., der Client kann neben der Festlegung des abzufragenden Objekts zusätzliche Information (im Hauptteil, engl. entity body) mitliefern.

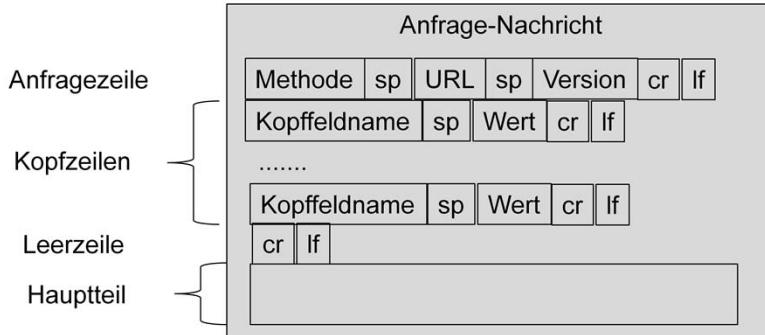
(3) Die HEAD-Methode entspricht der GET-Methode mit dem Unterschied, dass der eigentliche Inhalt (z.B. HTML-Text) des Objekts nicht vom Server an den Client geschickt wird.

(4) Die PUT-Methode erlaubt das Hochladen eines Objekts an eine bestimmte Stelle (Verzeichnispfad) eines Web-Servers, weshalb diese Methode häufig im Bereich des Web-Publishing eingesetzt wird.

(5) Durch die DELETE-Operation kann ein Benutzer oder eine Anwendung ein Objekt vom Web-Server löschen [96].

[KR03] James F. Kurose, Keith W. Ross: Computer Networking – A Top-down Approach Featuring the Internet, 2nd Edition, Addison Wesley, 2003.

## Aufbau einer Anfrage-Nachricht



- (1) Aufteilung in einen Kopf (engl. header) und einen Körper (engl. body)
- (2) Die erste Angabe in der Anfragezeile gibt die Methode an (z.B. GET)
- (3) Parameterangabe durch Name-Wert-Paare
- (4) Der Hauptteil (engl. entity body) ist bei einer GET-Anfrage leer

[KR03:94]

(1) Der Aufbau der HTTP-Nachrichten, die zwischen Client und Server ausgetauscht werden, orientiert sich an den bekannten Konzepten des E-Mail-Transportes. Das wird insbesondere an der Trennung einer HTTP-Nachricht in Kopf (engl. header) und Hauptteil (engl. entity body) deutlich. Die weiter unten in allgemeiner Form angegebenen Formate werden auf dieser Folie durch eine konkrete Anfrage bzw. Antwort verdeutlicht. Wir gehen bei diesem Beispiel implizit davon aus, dass die Verbindung zwischen Client und Server bereits besteht (also eine TCP-Verbindung auf Port 80). Es handelt sich im Beispiel um eine volle Anfrage, was an der angegebenen HTTP-Version ersichtlich ist.

(2) HTTP unterscheidet fünf Methoden: GET, POST, HEAD, PUT, DELETE

(3) (Kopffeldname sp Wert cr lf) Die Parameter der Anfrage werden durch Name-Wert-Paare angegeben, wobei Name und Wert durch ein Leerzeichen (engl. space sp) getrennt sind und jedes Name-Wert-Paar durch einen Zeilenumbruch (carriage return line feed cr lf).

Ein Beispiel eines Namens ist "Accept language" und mögliche Werte sind "en" (für englisch) oder "dt" (für deutsch).

(4) (Hauptteil) Dieser Teil der HTTP-Nachricht ist bei der GET-Methode leer und wird aber von der POST-Methode zur Übertragung von Informationen (die z.B. in ein Formular eingetragen wurden) vom Client zum Server genutzt.

Hinweis: Auch mittels GET können begrenzt Informationen übertragen werden, indem diese durch Sonderzeichen getrennt an den URL gehängt werden.

Beispiel: [www.some-site.com/animalsearch?monkeys&bananas](http://www.some-site.com/animalsearch?monkeys&bananas) [KR03:96].

[BS+08] Bryan Basham, Kathy Sierra, Bert Bates: Head First Servlets & JSP, Second Edition, O'Reilly, 2008.

[KR03] James F. Kurose, Keith W. Ross: Computer Networking – A Top-down Approach Featuring the Internet, 2nd Edition, Addison Wesley, 2003.

- (1) Die Nachricht ist eine ASCII-Zeichenkette
- (2) Die erste Angabe in der Anfragezeile besagt, dass es um ein GET (HTTP GET Request) handelt
- (3) "/select/selectPOI" gibt den URL der zu holenden (engl. get) Seite an
- (4) Nach der URL können optional Parameter angehängt werden
  - (1) Erster Parameter ist durch ein "?" und die weiteren sind durch jeweils ein "&" getrennt

- ```
1. GET /select/selectPOI.jsp?id=4711 HTTP/1.1
2. Host: www.kit.edu
3. Connection: close
4. User-agent: Mozilla/4.0 ...
5. Accept-language: de, en
```

[KR03:94]

(1) Zunächst fällt auf, dass die Nachrichten in normalem ASCII-Code geschrieben sind, weshalb das HTTP-Protokoll auch als ASCII-Protokoll bezeichnet wird [95].

In HTTP lassen sich durch eine Reihe von Parametern die Anfragen bzw. Antworten präzisieren. So können in der Anfrage die vom Client akzeptierte Dokumententypen bzw. Sprachen eingeschränkt werden oder die genutzte Client-Software dem Server mitgeteilt werden.

(2) (1. GET ...) Die erste Zeile einer HTTP-Anfrage-Nachricht heißt Anfragezeile (engl. request line). Die nachfolgenden Zeilen heißen Kopfzeilen (engl. header lines) [95].

(<Methode><URL><Version>) Die Anfragezeile besteht aus den drei genannten Feldern.

Die GET-Methode ermöglicht die Abfrage des durch den URL angegebenen Objekts und ist die am häufigsten genutzte Methode. Beispiele für andere Methoden sind POST und HEAD.

(3) (1. ... /select/selectPOI.jsp ...) Gibt den URL des angefragten Objekts an [95].

(3.1) Nach der Ressourcen-Angabe können - durch ein Fragezeichen getrennt – Parameter angegeben werden, die ihrerseits durch ein Ampersand (&) getrennt sind.; Beispiel: /select/selectBeerTaste.jsp?color=dark&taste=malty [BS+08:15].

(2. Host www.kit.edu) Spezifiziert den Host, auf dem das Objekt liegt.

(3. Connection: close) Die Angabe "close" drückt aus, dass der Server die Verbindung nach dem Senden des angefragten Objekts die Verbindung schließen soll, d.h. der Browser wünscht keine persistente HTTP-Verbindung.

(4. User-agent: Mozilla/4.0 ...) Diese Kopfzeile spezifiziert den Browser-Typ, durch den die Anfrage erfolgt.

(5. Accept-language: de, en) Hierdurch kann der Benutzer festlegen, welche Sprachversion des angefragten Objekts er bevorzugt (im Beispiel deutsch oder englisch).

Beispiele für weitere Parameter, die als Kopfzeilen in HTTP-Anfrage-Nachrichten auftreten können:

(Accept: image/gif, image/jpeg, \*/\*) Angaben zum gewünschten Inhalt

(Pragma no-cache) In der Anfrage des Client treten neben den zuvor eingeführten Parameter "Accept", "Accept-Language" oder "User-Agent" noch zwei weitere Parameter auf. Durch den Parameter FROM kann die E-Mail-Adresse der anfragenden Person mit übergeben werden. Im Zusammenhang mit dem Parameter PRAGMA ist folgender Sachverhalt wichtig: Der HTTP-Client kann abgefragte Web-Seiten im Cache halten und dadurch eine wiederholte Übertragung ggf. vermeiden. Durch die Angabe "no-cache" wird der Client aufgefordert, die Seite vom Server in jedem Fall zu fordern, selbst wenn sie sich im Cache befindet und seit diesem Zeitpunkt auch nicht verändert wurde. Der Parameter PRAGMA (pragmatic) dient allgemein dazu, nicht standardisierte und möglicherweise Software-spezifische Aspekte zwischen Client und Server auszutauschen ([TJE 97] - 932).

[KR03] James F. Kurose, Keith W. Ross: Computer Networking – A Top-down Approach Featuring the Internet, 2nd Edition, Addison Wesley, 2003.

## Uniform Resource Locator (URL)

- (1) Jede Ressource im Web hat eine eindeutige Adresse
  - (1) URL gibt das Format dieser Adresse vor
- (2) Aufbau eines URL
  - (1) Protokoll
    - (1) http://
  - (2) Sever
    - (1) www.kit.edu
  - (3) Port
    - (1) :80
  - (4) Pfad
    - (1) /select
  - (5) Anfrage-String (optional)
  - (6) Ressource
    - (1) selectPOI.jsp

[BS+08:20]

(1.1) Ein URL ist eine Form eines sog. Uniform Resource Identifier (URI). Eine zweite Form eines URI ist der Uniform Resource Name (URN), dessen wesentlicher Unterschied zum URL darin besteht, dass er ortsunabhängig und damit die Eigenschaft der Persistenz (d.h., bleibt bei einem Umzug der Ressource erhalten) aufweist.

|     |                             |
|-----|-----------------------------|
| URI | Uniform Resource Identifier |
| URL | Uniform Resource Locator    |
| URN | Uniform Resource Name       |

[BS+08] Bryan Basham, Kathy Sierra, Bert Bates: Head First Servlets & JSP, Second Edition, O'Reilly, 2008.

## HTTP GET: Antwort-Nachricht

(1) Beispiel einer HTTP-Antwort (vom Server zum Client)

```

1. HTTP/1.1 200 OK           Statuszeile
2. Connection: close         Erste Kopfzeile
3. Date: Tue, 15 Mar 2011 17:23:06 GMT
4. Server: Apache/1.3.0 (Unix)
5. Last-Modified: Thu, 06 Aug 1998 12:00:15 GMT
6. Content-Length: 6821
7. Content-Type: text/html

8. <HTML>
9. Gemäß HTML-Konventionen
10. strukturiertes Dokument
11. </HTML>

```

(2) Aufbau

Statuszeile

Kopfzeilen

Leerzeile

Entity Body

Hauptteil (Entity)



[KR03:97]

Eine Antwort-Nachricht besteht aus drei Abschnitten (engl. sections): (i) Statuszeile, (ii) Kopfzeilen, (iii) Entity Body (dt. Nutzdaten)

(200 OK) Der Code "200" bedeutet, dass die Anfrage erfolgreich bearbeitet werden konnte. Zu jedem Antwort-Code wird auch ein erläuternder Text mit angegeben; bei dem Code "200" ist das "OK".

Vor dem eigentlichen Inhalt des HTML-Dokuments kann auch der Server noch Parameter nutzen, um dem Client weitere Angaben zu machen. Hierzu zählen die Informationen zu dem Server selbst (z.B. verwendete Software) und insbesondere das in der Antwort mitgelieferte Dokument geben.

(Connection) Durch "close" teilt der Server dem Client mit, dass er die TCP-Verbindung nach Senden der Nachricht schließt.

(Date) Angabe der Zeit und des Datums, wann die HTTP-Antwort erzeugt wurde.

(Server) Gibt an, dass die Antwort vom Apache WebServer erzeugt wurde (entspricht dem Parameter "User-agent" in der Anfrage-Nachricht).

(Last-Modified) Diese Angabe ist relevant für das Cachen von Objekten.

(Content-Length) Anzahl der Bytes im Objekt, die gesendet wurden.

(Content-Type) Zeigt an, dass das Objekt im Entity-Body HTML-Text ist.

(<HTML>) Nach den Parametern folgt der eigentliche Inhalt der HTML-Seite, der durch zwei sog. HTML-Tags, <HTML> und </HTML> begrenzt ist (Näheres zu HTML siehe in der Kurseinheit INFORMATION).

(2) Das allgemeine Format entspricht bis auf die erste Zeile weitestgehend dem Aufbau der Anfrage-Nachricht.

(Status, Phrase) Die wichtigsten Statuscodes und deren Erklärung eines kurzen String-Wertes sind [:97]:

- 200 OK: Erfolgreiche Ausführung der Anfrage
- 301 Moved Permanently: Dauerhafter Umzug des angefragten Objekts auf die im Kopffeld "Location" angegebene URL.
- 400 Bad Request: Allgemeiner Fehler in der Anfrage, so dass der Server diese bearbeiten konnte.
- 404 Not Found: Das angefragte Dokument existiert nicht unter der angegebenen URL auf dem Server.
- 505 HTTP Version Not Supported: Vom Server nicht unterstützte HTTP-Version

Addison Wesley, 2003.

## HTTP POST zur Übertragung von Benutzerdaten

- (1) Zur Übertragung von Benutzerdaten dient die HTTP-Methode POST
  - (1) GET-Methode kann zur Übertragung kurzer Benutzerdaten in der Anfragezeile genutzt werden
  - (2) Die an den Server gesendeten Benutzerdaten werden auch als Payload bezeichnet

|                                         |                                                 |
|-----------------------------------------|-------------------------------------------------|
| 1. POST /select/selectPOI.jsp HTTP/1.1  | Angabe der Ressource auf dem Web-Server         |
| 2. Host: ...                            | Anfang des Abfrage-Kopfes                       |
| 3. ...                                  |                                                 |
| 4. Accept: ...                          |                                                 |
| 5. ...                                  |                                                 |
| 6. Connection: keep-alive               | Ende des Anfrage-Kopfes                         |
| 7. longitude=49.011924&latitude=8.41678 | Benutzerdaten als Payload im Nachrichten-Körper |

[BS+08:14]

(1) POST ist im Vergleich zu GET (das im Wesentlichen zum Abholen (engl. get) einer Ressource vom Server dient) eine mächtigere Anfrage (GET plus plus), da durch diese Methode neben der Anfrage einer Ressource zusätzlich Benutzerdaten an den Server übermittelt werden können [BS+08:13].

(1.1) Die Benutzerdaten werden nach der Angabe der Ressource durch ein Fragezeichen getrennt als einfache Namen-Werte-Paare ergänzt.

Beispiel: "GET /select/selectPOI.jsp?longitude=47&latitude=11 HTTP/1.1"

Zu beachten: Die Benutzerdaten sind begrenzt und werden in der Browser-Titelleiste (bar) für jeden sichtbar angezeigt [BS+08:14].

(2) Eine andere Bezeichnung ist "Message Body" (Nachrichten-Körper).

(7. longitude) Die geographische Länge (engl. longitude) und Breite (engl. latitude) sind hier in der Winkeleinheit Grad in Dezimalschreibweise (z. B. 66,34°).

[BS+08] Bryan Basham, Kathy Sierra, Bert Bates: Head First Servlets & JSP, Second Edition, O'Reilly, 2008.

## Ablauf einer HTTP-Bearbeitung

- (1) Der Benutzer tippt in seinen Web-Browser eine URL ein
- (2) Der Browser erzeugt daraus ein HTTP Get Request und schickt dieses zum Server
- (3) Der Server sucht die im URL enthaltene Seite im Dateisystem
- (4) Falls vorhanden, erzeugt der Server eine HTTP-Antwort, die er an den Browser schickt
- (5) Der Browser wandelt das HTML in die entsprechende graphische Darstellung um und zeigt diese dem Benutzer an



45

02.09.2011 WASA - 2-2 KOMMUNIKATION

Cooperation & Management (C&M, Prof. Abeck)  
Institut für Telematik, Fakultät für Informatik

[BS+08:18]

Der Ablauf soll an einem konkreten Beispiel verdeutlicht werden. Die aufgerufene Web-Seite soll die Abfrage ermöglichen, ob es sich bei einem Campus-Besucher um einen KIT-Angehörigen (Studierender, Mitarbeiter) oder einen KIT-Externen handelt.

(1) URL: "http://www.wasa.edu/test/Visitor.html"

(2) Die Anfragenachricht lautet:

GET /test/Visitor.html HTTP/1.1

HOST: www.wasa.edu

User-Agent: Mozilla/5.0

...

(3, 4) Die HTML-Datei hat den folgenden Inhalt:

```
<html><body>
<h1 align=center>KITCampusGuide-Anmeldeseite</h1>
<form>
Welchen Status haben Sie hinsichtlich des KIT?<p>
<input type=radio name=besuchertyp value=KIT-intern> KIT-Angehöriger<br>
<input type=radio name=besuchertyp value=KIT-extern> KIT-Besucher<br>
<center>
<input type=Abschicken>
</center>
</form>
</body></html>
```

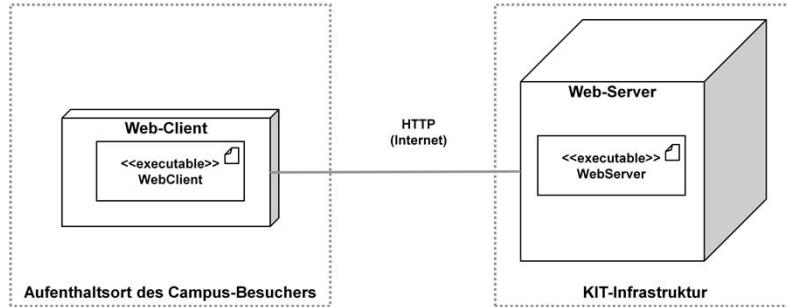
(5) Auf der rechten Seite ist die vom Browser aufgrund des Inhalts der HTML-Datei erzeugte (engl. render) Web-Seite abgebildet.

[BS+08] Bryan Basham, Kathy Sierra, Bert Bates: Head First Servlets & JSP, Second Edition, O'Reilly, 2008.

- (1) Welche Methoden werden in HTTP/1.1 unterstützt?
- (2) Aus welchen Teilen setzt sich eine Anfrage-Nachricht bzw. eine Antwort-Nachricht zusammen
- (3) Bieten sich GET oder POST oder beide Methoden zur Implementierung der nachfolgenden Funktionalität an:
  - (1) Ein Benutzer gibt einen Login-Namen und ein Passwort zurück
  - (2) Ein Benutzer fragt eine neue Seite via Hyperlink
  - (3) Ein Chatroom-Benutzer sendet eine geschriebene Antwort
  - (4) Ein Benutzer wählt die "Next"-Schaltfläche, um die nächste Seite zu sehen
- (4) Aus welchen Schritten besteht die Bearbeitung einer HTTP GET-Anfrage?

## LZ HTTP – PA HTTP-PROGRAMMIERUNG

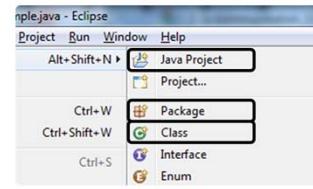
- (1) Es ist eine Client/Server-Anwendung zu schreiben, die HTTP als Kommunikationsprotokoll nutzt
  - (1) Der Client stellt eine HTTP-konforme Anfrage, die der Server bearbeitet
  - (2) Der Server schickt das Ergebnis durch eine HTTP-konforme Antwort an den Client zurück



Der Unterschied zur Praktischen Aufgabe "SOCKET-PROGRAMMIERUNG" besteht darin, dass der Client und der Server durch ein Protokoll miteinander kommunizieren. Das bedeutet, dass gewisse Regeln beim Kommunizieren einzuhalten sind.

## PA HTTP-PROGRAMMIERUNG – Anlegen eines neuen Projekts

- (1) "File" → "New"
  - (1) Java-Projekt "2-2\_HTTP-PROGRAMMIERUNG" anlegen
- (2) Rechtsklick auf das Projekt
  - (1) Pakete "clientSide" und "serverSide" anlegen
- (3) Klassen erstellen
  - (1) "WebClient" im Paket "clientSide"
  - (2) "HTTPProcessor" und "WebServer" im Paket "serverSide"



(1) Die Konvention für Projektnamen lautet:

<Nummer der WASA-Kurseinheit\_<Name der Praktischen Aufgabe>

(1) Durch die Einführung der Pakete lässt sich die Verteilung des zu erstellenden Softwaresystems in eine Client-Seite und eine Server-Seite geeignet abbilden.

(3.2) Die "HTTPProcessor"-Klasse ist dafür verantwortlich, HTTP-konforme Anfragen zu bearbeiten und HTTP-konforme Antworten zu liefern. Die Klasse implementiert den Antwortmechanismus des Web-Servers und gehört deshalb in das Paket "serverSide".

## PA HTTP-PROGRAMMIERUNG – Client-Programm `HTTPClient.java`

```
1. package clientSide;
2. import java.io.*;
3. import java.net.*;
4.
5. public class WebClient {
6.
7.     public static void main(String[] args) throws Exception {
8.         String request;
9.         String reply;
10.        BufferedReader inFromUser = new BufferedReader(new InputStreamReader(System.in));
11.        Socket clientSocket = new Socket("localhost", 8081);
12.        DataOutputStream outToServer = new DataOutputStream(clientSocket.getOutputStream());
13.        BufferedReader inFromServer =
14.            new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));
15.        request = inFromUser.readLine();
16.        outToServer.writeBytes(request + "\r\n");
17.        System.out.println("Antwort vom Server: ");
18.        while ((reply = inFromServer.readLine()) != null)
19.            System.out.println(reply);
20.        clientSocket.close();
21.    } // main
22. } // WebClient
```

Das Programm "WebClient" hat viele Ähnlichkeiten mit dem Programm "TCPClient". Zeilen, die identisch sind, werden hier nicht mehr kommentiert.

(6., 7.) Um den Code für das HTTP-Szenario verständlicher zu machen, werden andere Variablennamen benutzt.

Das String-Objekt "request" ist die Eingabe vom Benutzer und wird als Anfrage an den Server übernommen.

Das String-Objekt "reply" ist die Antwort, die der Client vom Server erhält.

(13.) Dem Benutzer wird das Erstellen von korrekten Anfragen über die Konsoleeingabe überlassen.

(14.) Der Anfrage wird ein CRLF ("\r": Carriage Return (CR) und "\n": Line Feed (LF)) angehängt, um die Nachricht HTTP-konform zu machen.

(16. – 18.) Eine "while"-Schleife wurde eingeführt, weil der Server laut HTTP-Protokoll eine mehrzeilige Antwort liefert. Der Client liest vom Socket solange, bis keine Informationen vom Server kommen und beendet die Kommunikation durch Schließen des Sockets.

## PA HTTP-PROGRAMMIERUNG – Server-Programm "WebServer.java"

```
1. package serverSide;
2. import java.io.*;
3. import java.net.*;
4. public class WebServer {
5.     public static void main(String[] args) throws Exception {
6.         HTTPProcessor http = new HTTPProcessor();
7.         String requestMessageLine;
8.         ServerSocket listeningSocket = new ServerSocket(8000);
9.         while (true) {
10.             Socket connectionSocket = listeningSocket.accept();
11.             BufferedReader inFromClient =
12.                 new BufferedReader(new InputStreamReader(connectionSocket.getInputStream()));
13.             DataOutputStream outToClient =
14.                 new DataOutputStream(connectionSocket.getOutputStream());
15.             requestMessageLine = inFromClient.readLine();
16.             outToClient.writeBytes(http.processInput(requestMessageLine));
17.             connectionSocket.close();
18.         } // while - listen for clients
19.     } // main
20. } // WebServer
```

Das Programm "WebServer" hat viele Ähnlichkeiten mit dem Programm "TCPServer". Zeilen, die identisch sind, werden hier nicht mehr kommentiert.

(16.) Das Bearbeiten der Anfrage wird an das in Programmzeile (6.) definierte Objekt "http" des Typs "HTTPProcessor" delegiert. Die "processInput()"-Methode der "HTTPProcessor"-Klasse erhält als Parameter die Anfrage des Client und hat als Rückgabewert eine HTTP-konforme Antwort in String-Form. Der Server schickt die von "HTTPProcessor" berechnete Antwort direkt an den Client.

Die "HTTPProcessor"-Klasse wird im Folgenden beschrieben.

# PA HTTP-PROGRAMMIERUNG – HTTP-Anfrage und der HTTP-Antwort (Wdhl.)

## (1) HTTP-Anfrage

method	Sp	URL	Sp	Version	Cr	If	Request line
Header field name	:	value		Cr	If		
	:						
Header field name	:	value		Cr	If		
Cr	If						
Entity Body							

## (2) HTTP-Antwort

Version	Sp	Status Code	Sp	Phrase	Cr	If	Status Line
Header field name	:	value		Cr	If		
	:						
Header field name	:	value		Cr	If		
Cr	If						
Entity Body							

Der Aufbau der HTTP-Anfrage und der HTTP-Antwort wurde auf vorhergehenden Seiten bereits ausführlich behandelt und wird an dieser Stelle der Praktischen Aufgabe nochmals wiederholt, da diese Information wichtig ist, um die Erstellung der HTTP-Nachrichten nachvollziehen zu können.

## PA HTTP-PROGRAMMIERUNG – Die "HTTPProcessor"-Klasse

```
1. package serverSide;
2. import java.io.*;
3. import java.util.StringTokenizer;
4. public class HTTPProcessor {
5.     private final static String CRLF = "\r\n";
6.     private final static Integer ERROR = 0;
7.     private final static Integer GET = 1;
8.     public String processInput(String requestMessageLine) throws Exception{
9.         String response;
10.        StringTokenizer tokenizedLine = new StringTokenizer(requestMessageLine);
11.        switch (determineRequest(tokenizedLine.nextToken())) {
12.            // if the method is GET then we need to fetch the resource
13.            case GET:
14.                response = buildGETResponse(tokenizedLine.nextToken());
15.                break;
16.            // if it is not GET it is a bad request
17.            default:
18.                response = "HTTP/1.0 400 Bad Request" + CRLF + CRLF;
19.        } //switch
20.        return response;
21.    } //processInput;
```

- (1.) Deklaration des Paketes, zu dem die Klasse gehört.
- (2.) Das "java.io"-Paket enthält Klassen für Eingabe- und Ausgabe-Ströme. Hierzu gehören auch die im Programm verwendeten Klassen "File" und "FileInputStream".
- (3.) Das "java.util"-Paket enthält nützliche Klassen für Internationalisierung, Datum- und Zeitbearbeitung und verschiedene Utility-Klassen, wozu auch die Klasse "StringTokenizer" gehört.
- (4.) Anfang des die Klasse "HTTPProcessor" definierenden Blocks. Der Block wird durch eine geschweifte Klammer (engl. curly bracket) geöffnet und enthält Variablen und Methoden. Die Klasse "HTTPProcessor" besitzt eine Klassenvariable "CRLF" und die auf dieser Seite beschriebene Methode "processInput()", sowie die auf den folgenden Seiten bechriebenen Methoden "determineRequest()", "getResource()" und "buildGETResponse()".
- (5.) Da "\r" (Carriage Return, CR) und "\n" (Line Feed, LF) häufig zusammen in einer HTTP-Antwort vorkommen, wird CRLF als Konstante mit dem Keyword (final) vom Typen String in der Klasse definiert, um die Arbeit zu vereinfachen und den Code übersichtlicher zu gestalten. Die Variable wird privat deklariert, da sie nur einen internen Nutzen hat und es immer gut ist das Prinzip der Einkapselung anzuwenden.
- (6.) Die "processInput()"-Methode erwartet als Parameter die Anfrage ("requestMessageLine") als String-Objekt und liefert eine Antwort ("response") als String-Objekt zurück.
- (7.) Das Objekt "response" ist der vom Server zurückgelieferte String als Antwort auf die Anfrage des Client.
- (8.) Das Objekt "tokenizedLine" ist eine Helperklasse vom Typen "StringTokenizer". Es bricht einen String in mehrere Stücke (engl. token) und erleichtert so das Parsen der Anfrage. Die "nextToken()" Methode der Klasse "StringTokenizer" liefert als Rückgabewert das nächste Stück des Strings. Die Stücke eines Strings sind die Wörter, die durch Leerzeichen getrennt sind. Es können auch andere Trennzeichen gewählt werden, aber da hier HTTP geparsst wird, ist dies nicht notwendig.
- (9.) Der "switch"-Befehl hat die Aufgabe, den Anfragetyp zu bestimmen. Die Bestimmung erfolgt durch den Rückgabewert der "determineRequest()"-Methode. Als Parameter wird das erste Stück des "requestMessageLine"-Strings übergeben, der bei einer HTTP-Nachricht die HTTP-Methode darstellt. Der "switch"-Befehl wurde gewählt um mehr Klarheit im Code zu haben.
- (11.) Wenn es eine GET-Methode ist, wird von der "determineRequest()"-Methode die konstante GET zurückgeliefert und die Anfrage wird als GET-Anfrage bearbeitet.
- (12.) Das nächste Stück der Anfrage ist der Name der Ressource (wird in der Methode "getResource()" behandelt, die in der Methode "buildGETResponse()" aufgerufen wird, siehe nächste Seite).
- (13.) Die "buildGETResponse()"-Methode bekommt als Parameter das nächste Stück der Anfrage, nämlich die URL und liefert die Antwort auf die GET-Anfrage als String zurück.
- (17.) Falls der zurückgelieferte Wert verschieden von der Konstanten GET ist, wird der "default"-Ast des Switch-Befehls weiterverfolgt und es wird eine HTTP-Fehlernachricht zum Server zurückgeliefert, die an den Client geschickt wird.
- (18.) Wenn der HTTP-Prozessor die Anfrage nicht versteht, wird eine Standardfehlernachricht erstellt, die auch HTTP-konform ist.

(20.) Zurückliefern der berechneten Antwort.

## PA HTTP-PROGRAMMIERUNG – "determineRequest()", "buildGETResponse()"

```
1. private int determineRequest(String method) {  
2.     if (method.equals("GET"))  
3.         return GET;  
4.     else  
5.         return ERROR;  
6. } //determineRequest  
  
7. private String buildGETResponse(String resourceName) throws Exception {  
8.     StringBuffer response = new StringBuffer();  
9.     String resourceData = getResource(resourceName);  
  
10.    response.append("HTTP/1.0 200 OK" + CRLF);  
  
11.    if (resourceName.endsWith(".txt"))  
12.        response.append("Content-Type: text/plain" + CRLF);  
13.    if (resourceName.endsWith(".html"))  
14.        response.append("Content-Type: text/html" + CRLF);  
  
15.    response.append("Content-Length: " + resourceData.length() + CRLF);  
16.    response.append(CRLF);  
17.    response.append(resourceData);  
18.    return response.toString();  
19. } // buildGETResponse
```

- (1.) Die "determineRequest()"-Methode erwartet als Parameter "method" als String-Objekt, das die HTTP-Methode darstellt und liefert eine ganze Zahl zurück, die im Switch-Befehl der "processInput()"-Methode benutzt wird, um den Anfragetyp zu bestimmen.
- (2.) Mit Hilfe der "equals()"-Methode des String-Objekts lässt sich entscheiden, ob es sich um eine HTTP-GET-Methode handelt und liefern die Konstante GET im positiven Fall zurück. Dies ist eine sehr einfache Entscheidungsroutine die ausgebaut werden kann um mehrere Typen von Methoden zu unterscheiden.
- (5.) Wenn man die HTTP-Methode nicht identifizieren kann, liefert die "determineRequest()"-Methode die Konstante ERROR zurück.
- (7.) Die "buildGETResponse()"-Methode erwartet als Parameter den Ressourcennamen "resourceName" (entspricht dem URL) als String-Objekt und liefert das die Antwort darstellende String-Objekt "response" zurück.
- (8.) Der Rückgabewert "response" ist vom Typ "StringBuffer". Bei einem "StringBuffer"-Objekt können die Veränderungen am Objekt selbst vorgenommen werden, im Gegensatz zum String-Objekt, bei dem ein Hinzufügen jedes Mal ein neues Objekt erzeugt. Der "StringBuffer" wird in dieser Implementierung bevorzugt, weil viele Zeichenketten, die die verschiedenen Zeilen der Antwort darstellen, dem "response"-Objekt angehängt werden. "StringBuffer" sind sicher beim Benutzen von mehreren Threads [JAVASE6-DOC] .
- (9.) Aufruf der "getResource()"-Methode und Speichern des Inhalts der angeforderten Ressource in das String-Objekt "resourceData".
- (10.) Die Statuszeile der HTTP-Antwort wird an den "StringBuffer" angehängt (engl. append), nachdem die Daten der Ressource ermittelt wurden.
- (11.-14) Je nach Endung der Datei wird der MIME-Typ, also der Content-Type ermittelt.
- (15.) Der Content-Length Header wird der Antwort hinzugefügt. Durch den Content-Length Header wird in HTTP 1.0 die Präsenz eines Nachrichtenkörpers, der Daten der Ressource, signalisiert. In HTTP 1.0 ist dieser Header verpflichtend, was auch für HTTP 1.1 gilt, um die Rückwärtskompatibilität zu HTTP 1.0 zu gewährleisten [HTTP1.1].
- (16.) Das dem HTTP-Kopf abgrenzende CRLF wird der Antwort hinzugefügt.
- (17) Die Daten der Ressource werden der Antwort hinzugefügt.
- (18) Als Rückgabe wird ein String-Objekt, mit der "toString()"-Methode, mit dem Inhalt des "StringBuffers" erzeugt.

```
1.     private String getResource(String resourceName) throws Exception {  
2.         if (resourceName.startsWith("/"))  
3.             resourceName = resourceName.substring(1);  
4.  
5.         File file = new File(resourceName);  
6.         int numBytes = (int) file.length();  
7.  
8.         FileInputStream inFile = new FileInputStream(file);  
9.  
10.        byte[] fileInBytes = new byte[numBytes];  
11.        inFile.read(fileInBytes);  
12.        return new String(fileInBytes);  
13.    } // getResource  
14. } // HTTPProcessor
```

(1.) Die "getResource()"-Methode erwartet als Parameter den Namen der zu holenden Ressource "resourceName" als String-Objekt und liefert den Inhalt der Ressource als String-Objekt zurück. Dies ist somit eine einfache Methode zum Holen einer Ressource aus dem Wurzelverzeichnis des Programms.  
(2., 3.) Um den Namen der Ressource aus z.B. "/POIxyz.dat" zu erhalten, muss man das "/" Symbol entfernen. Die "startsWith()"-Methode liefert das erste Zeichen des Strings. Wenn dieser das "/"-Symbol ist, wird mit Hilfe der "substring()"-Methode ein String erstellt, der nach dem ersten Symbol startet.

(4.) Dem Konstruktor der "File"-Klasse wird der Name der zu lesenden Datei übergeben. Ein "File"-Objekt ist eine abstrakte Darstellung von Datei- und Verzeichnispfaden.  
(5.) Die Länge der Datei wird ermittelt, so dass im nächsten Schritt bekannt ist, wie viele Bytes eingelesen werden müssen.

(6.) Das tatsächliche Einlesen von Dateien kann man mit Hilfe eines "FileInputStream"-Objekts realisieren. Ein "FileInputStream" öffnet eine Verbindung zu einer vorhandenen Datei.

(7.) Um einen Speicherort für die Daten aus der Datei zu haben, wird ein Byte-Array erstellt, dessen Länge der des Dateiinhalts entspricht.  
(8.) Einlesen der Datei in den Byte-Array.  
(9.) Die ausgelesenen Daten werden in einen String verpackt und zurückgeliefert, da ein String einfacher in den aufrufenden Methoden zu handhaben ist.

[JAVASE-DOC] Java Standard Edition 6 online Dokumentation, <http://download.oracle.com/javase/6/docs/index.html>

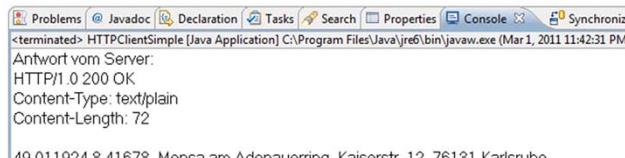
## PA HTTP-PROGRAMMIERUNG – Ausführen der HTTP-Anwendung

- (1) Erstellen einer Datei mit dem Namen "POIMensa.txt"
  - (1) Speichern unter "<Workspace>\2-2\_HTTP-PROGRAMMIERUNG\"
  - (2) Einen unstrukturierten Inhalt hinzufügen:

```
49.011924,8.41678, Mensa am Adenauerring, Kaiserstr. 12, 76131 Karlsruhe
```

- (2) Testen der Anwendung mit zwei HTTP-Anfragen

- (1) GET /POIMensa.txt HTTP/1.0
- (2) HEAD /POIMensa.txt HTTP/1.0



```
Problems @ Javadoc Declaration Tasks Search Properties Console Synchronize
<terminated> HTTPClientSimple [Java Application] C:\Program Files\Java\jre6\bin\javaw.exe (Mar 1, 2011 11:42:31 PM)
Antwort vom Server:
HTTP/1.0 200 OK
Content-Type: text/plain
Content-Length: 72

49.011924,8.41678, Mensa am Adenauerring, Kaiserstr. 12, 76131 Karlsruhe
```

(1) Die Daten, die über den Server verfügbar gemacht werden sollen, werden aus Gründen der Einfachheit aus einer Datei bezogen. Die "HTTPProcessor"-Klasse ist so gebaut, dass sie die Daten aus einer Datei beziehen kann.

(1.1) Die ".txt"-Datei muss sich in dem Ordner befinden, aus dem das Programm gestartet wird, z.B. im Wurzelordner des Projektes ("<Workspace>\<Projekt-Name>\"), wenn man den Server aus Eclipse startet. Wird der Server von der Eingabeaufforderung aus gestartet, muss die ".txt"-Datei im Elternverzeichnis des Paketes (also im selben Pfad, aus dem der Server gestartet wird) liegen. Beim Aufruf des "File"-Konstruktors in der "HTTPProcessor"-Klasse wird dort nach der Datei gesucht.

(2.1) Die GET-Anfrage sollte erfolgreich ausgeführt werden und es sollte eine HTTP-Antwort mit Status-Code "200" und dem Inhalt der Datei im Körper der Nachricht zurückgeliefert werden.

(2.2) Die HEAD-Anfrage wird nicht in der "HTTPProcessor"-Klasse behandelt, also sollte hier eine HTTP Antwort mit Status-Code "401" zurückgeliefert werden.