Netzprogrammierung
Rückblick

Prof. Dr.-Ing. Robert Tolksdorf
Freie Universität Berlin
Institut für Informatik
Netzbasierte Informationssysteme
mailto: tolk@inf.fu-berlin.de
http://www.robert-tolksdorf.de

Vorlesungsrückblick

Vorlesungsblöcke

- Organisation und Einführung
- Fernaufrufe
  - Remote Procedure Call, Konzepte und Grundelemente
  - RMI in Java, entfernter Objektaufruf
  - CORBA, sprachunabhängige Nutzung verteilter Objekte

- Nachrichtenbasierte Kommunikation
  - Sockets, Basis der Kommunikation im Internet
  - Internet Dienste
  - Internet Dienste in Java

Vorlesungsblöcke

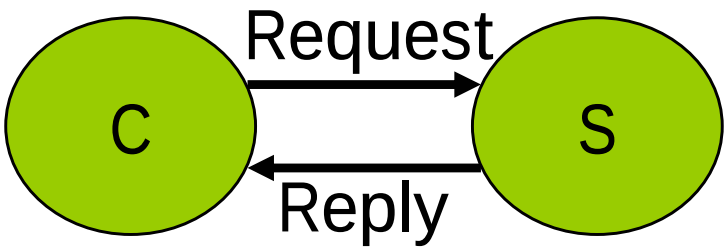
- Web Programmierung Server- und Klientenseitig
  - HTTP-Kommunikation in Java (Zustand, Sicherheit etc.)
  - Serverseitige Ausführung: CGI, Servlets, SSI, JSP
  - HTML und Verarbeitung
  - XML, XML-Verarbeitung
  - Clientenseitige Ausführung: Javascript, Applets
- Weitere Modelle
  - Koordinationssprachen, entkoppelter Kommunikation und Koordination
  - Agenten, autonome netzbasierte Entitäten
- Klausur

- Remote Procedure Call
  Konzepte und Grundelemente des entfernten Prozeduraufrufs
- RMI
  Konzepte und Technologien des entfernten Objektaufrufs in Java
- CORBA
  Konzepte und Technologien der sprachunabhängigen Nutzung verteilter Objekte

Remote Procedure Calls

Client-Server

- Rechner interagieren über Rechnergrenzen durch
  - Nachrichtenaustausch (z.B. Internet Mitteilungen)
  - Fernaufruf (RPC, RMI, CORBA)
  - Simulierten gemeinsamen Speicher (Tupelraum)
  - ...
- Dominierendes Interaktionsmodell: Client/Server
- *Client*: Prozess, der Dienst von einem anderen Prozess anfordert (Anforderung, Request)
- *Server*: Prozess, der auf Anforderung eines Clients einen Dienst erbringt und Ergebnis vermeldet (Antwort, Reply)



- Feste (starre) Rollenverteilung
- Fester Interaktionsablauf, z.B. keine Zwischenergebnisse vorgesehen

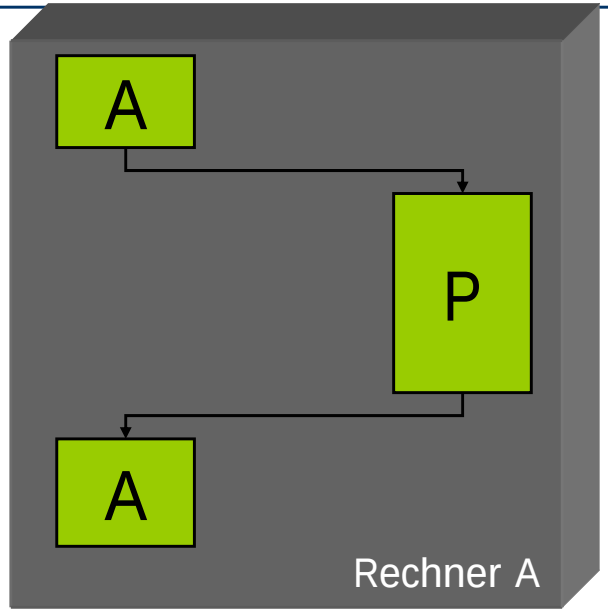
RPC Kontroll- und Datenfluss

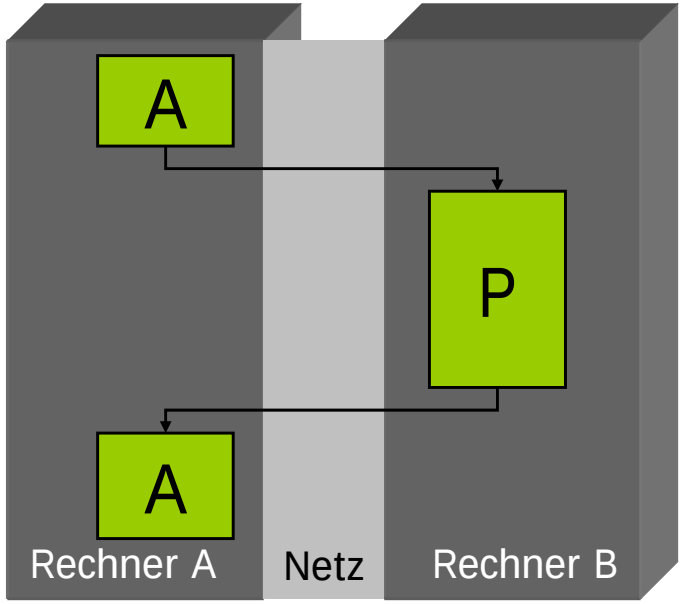
- Prozeduraufruf steuert
  - Kontrollfluss und
  - Parameter

  vom Aufrufer in eine Prozedur
- Prozedurbeendigung steuert
  - Kontrollfluss und
  - Ergebnisdaten

  zurück

- Entfernter Prozeduraufruf (Remote Procedure Call, RPC) transferiert Kontrollfluss und Daten über ein Netzwerk zwischen Rechnern



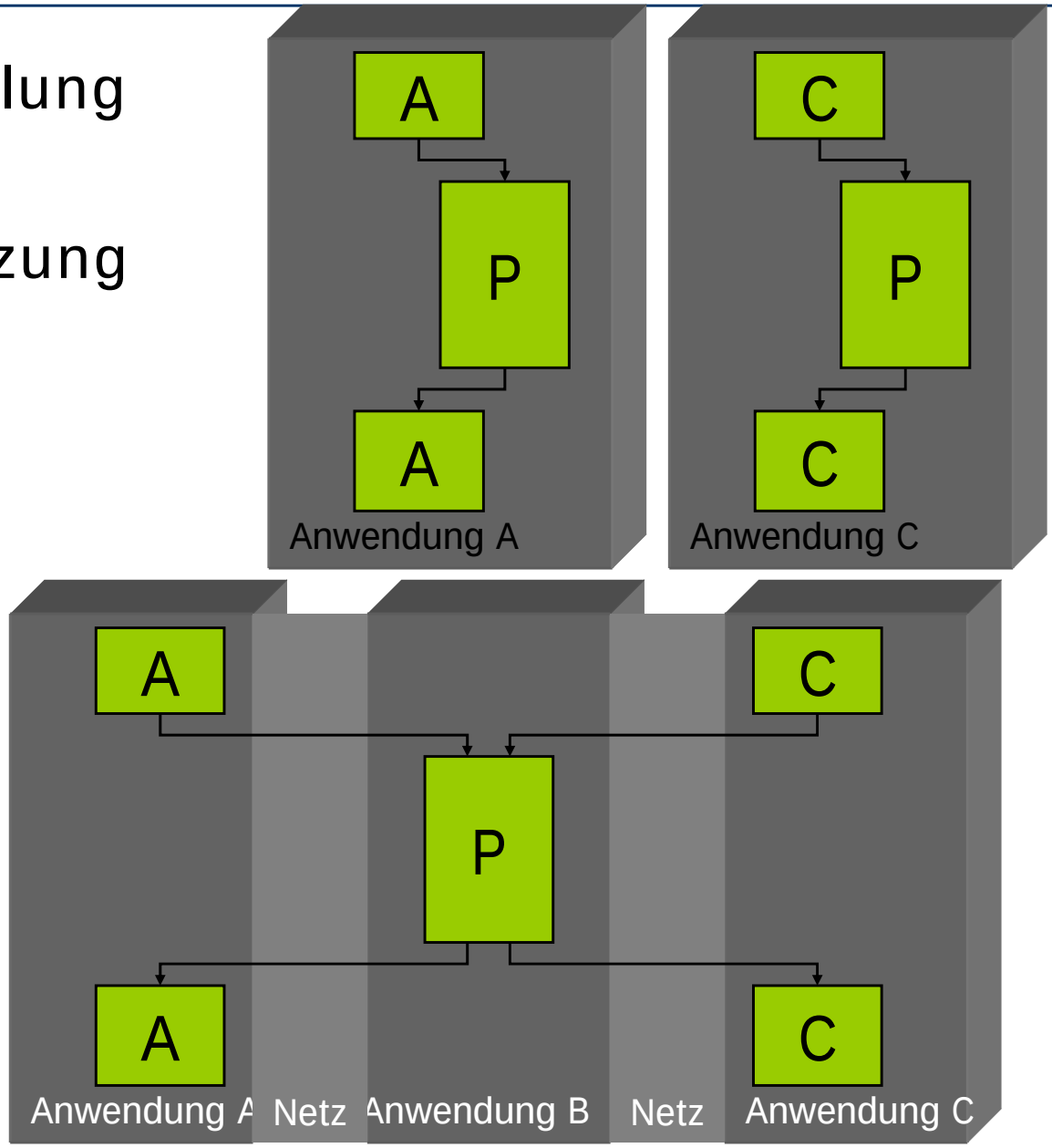


Unterschiede

Lokaler Aufruf	*Entfernter Aufruf*
Aufrufer und Prozedur im selben Prozess ausgeführt	Aufrufer und Prozedur in unterschiedlichen nebenläufigen Prozessen
Aufrufer und Prozedur im selben Adressraum	Aufrufer und Prozedur in unterschiedlichen Adressräumen
Aufrufer und Prozedur in selben Hard- und Software-umgebung	Aufrufer und Prozedur in unterschiedlicher Hard- und Softwareumgebung
Aufrufer und Prozedur haben gleiche Lebensdauer	Aufrufer und Prozedur haben unterschiedliche Lebensdauer
Aufruf ist immer fehlerfrei	Aufruf ist fehlerbehaftet (Netz, Aufgerufener)
Nur Anwendungsfehler berücksichtigt	Zusätzlich Aufruffehler behandeln

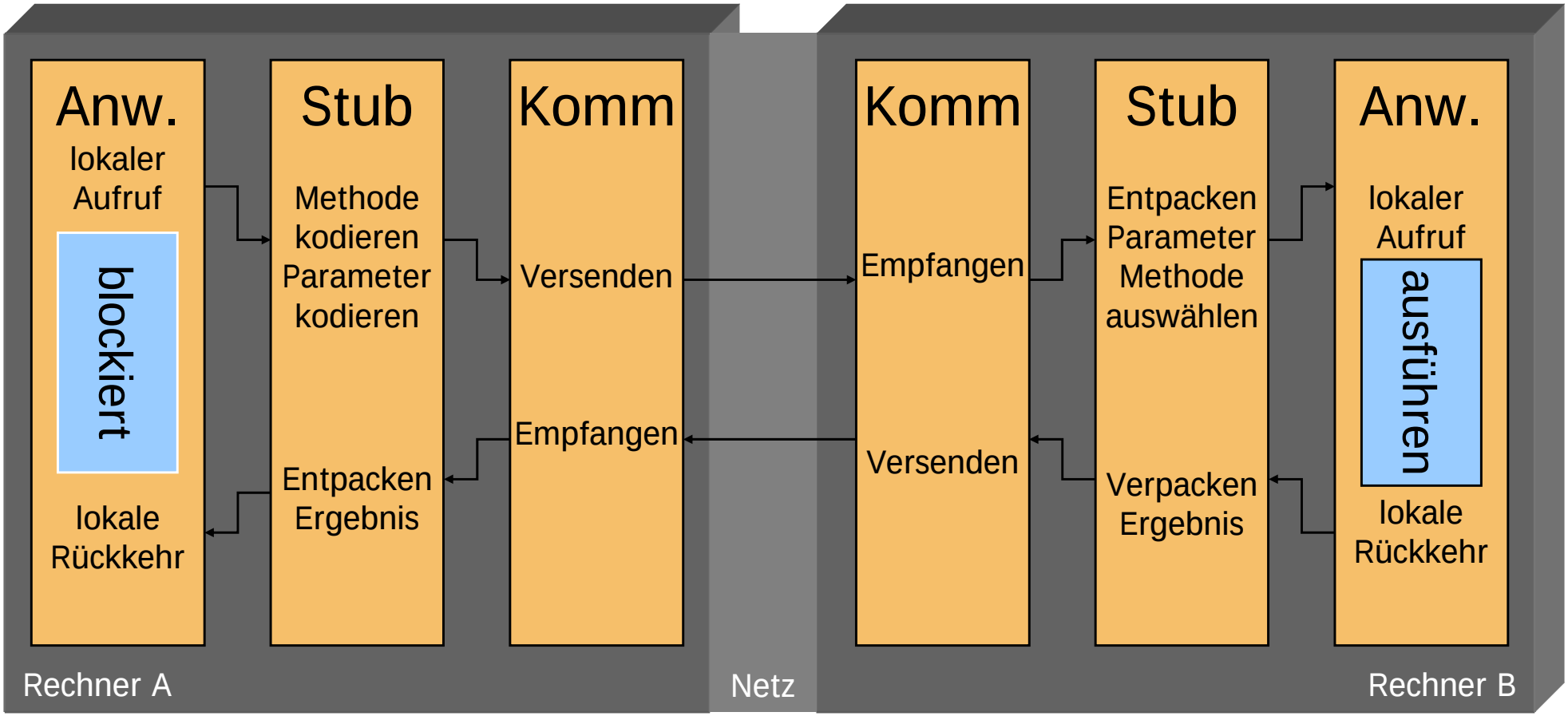
Vorteile der Netzbasierung TOOD

- Bessere Aufgabenverteilung
- Bessere Lastverteilung
- Bessere Ressourcennutzung
- Bessere Modularität
- Bessere Wieder-verwendbarkeit
- Größere Offenheit
- Besser Integrationsfähigkeit
- ...



Komponenten beim RPC

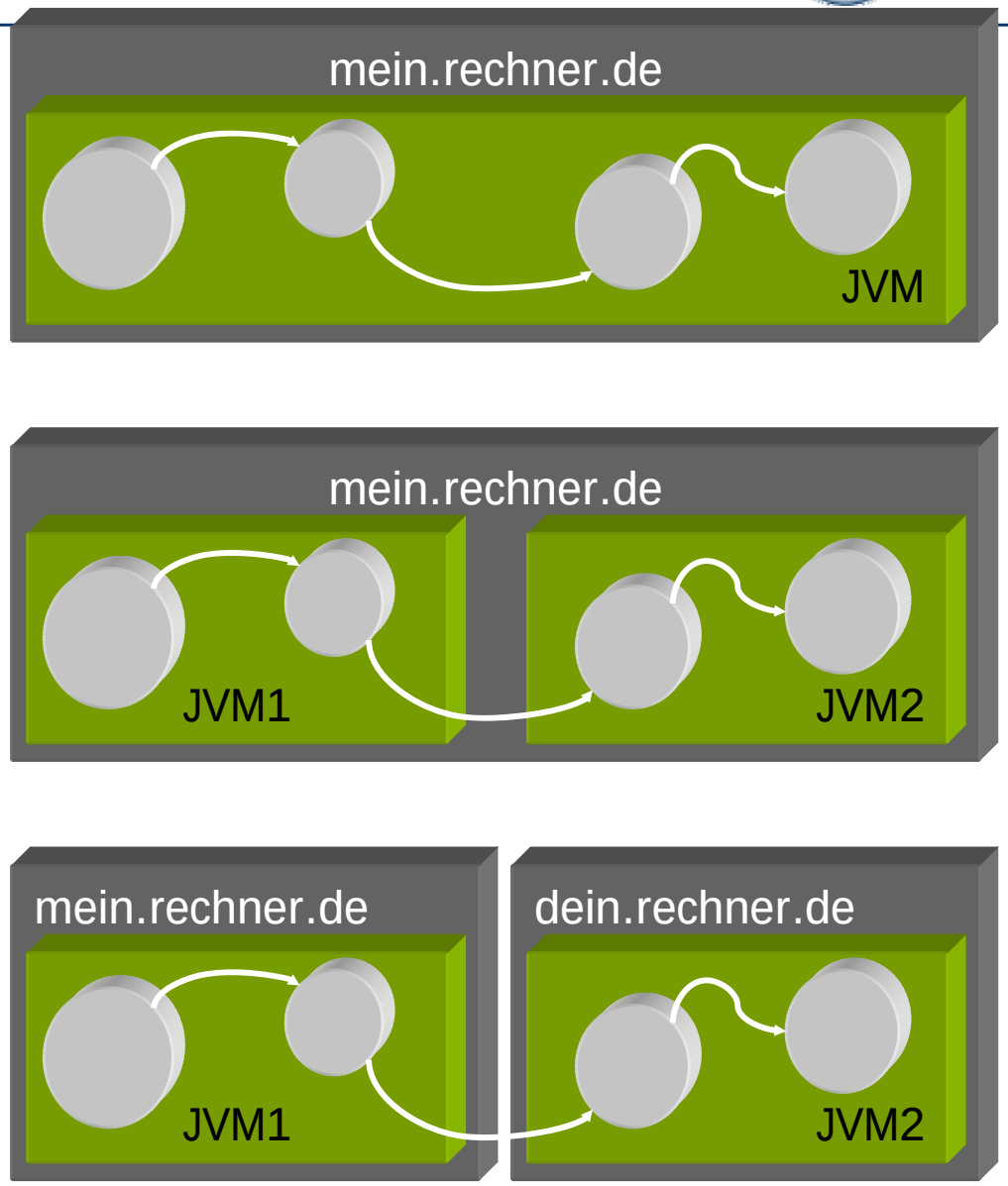
- Anwendungsprozeduren: Eigentliche Arbeit
- Stubs: Ver- und Entpacken von Daten zum Transport
- Kommunikation: Transport von Daten



RMI

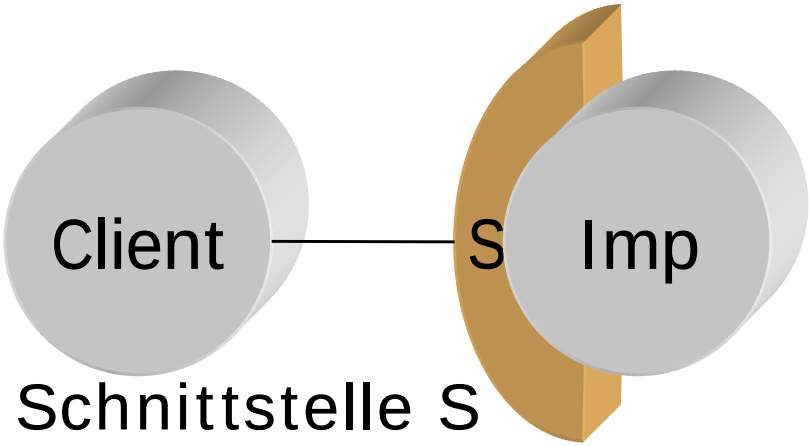
Lokale vs. verteilte Java-Programme

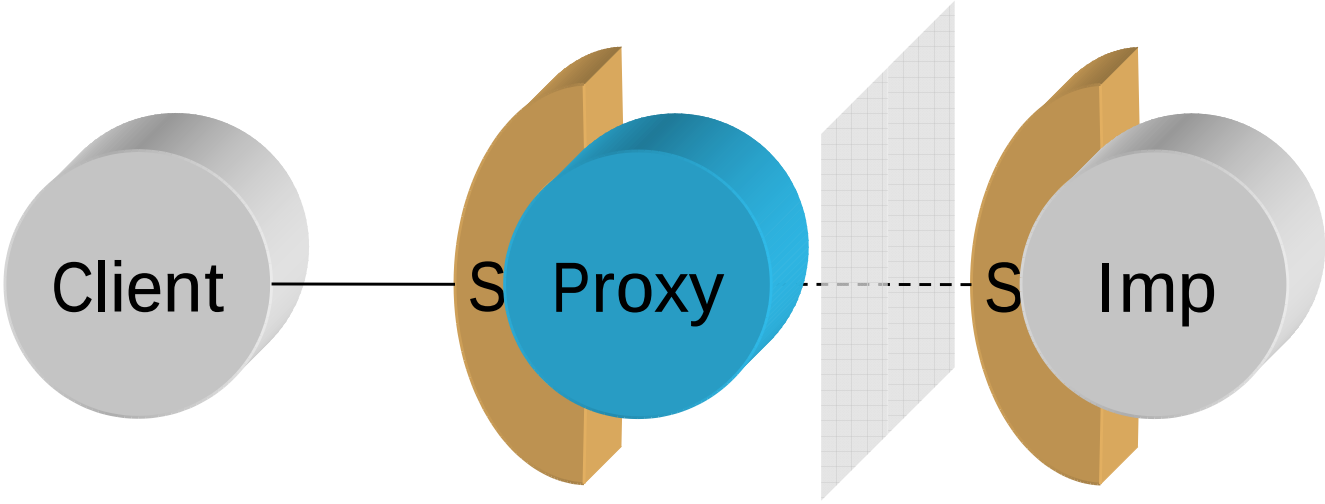
- Ein Java Programm arbeitet in einer virtuellen Java Maschine (JVM)
- Zwischen JVMs können mit *Remote Method Invocation* Methoden an Objekten aufgerufen werden
- JVMs können auf unterschiedlichen Internet-Rechnern laufen
- Sie müssen es aber nicht...



Schnittstellen

- Objektschnittstellen definieren Methoden des Objekts
- Unterschiedliche Implementierungen für gleiche Schnittstelle S
- Modulschnittstellen des RPC werden auf Objektschnittstellen abgebildet
- Aufrufweiterleitung durch Stellvertreter (Proxy)



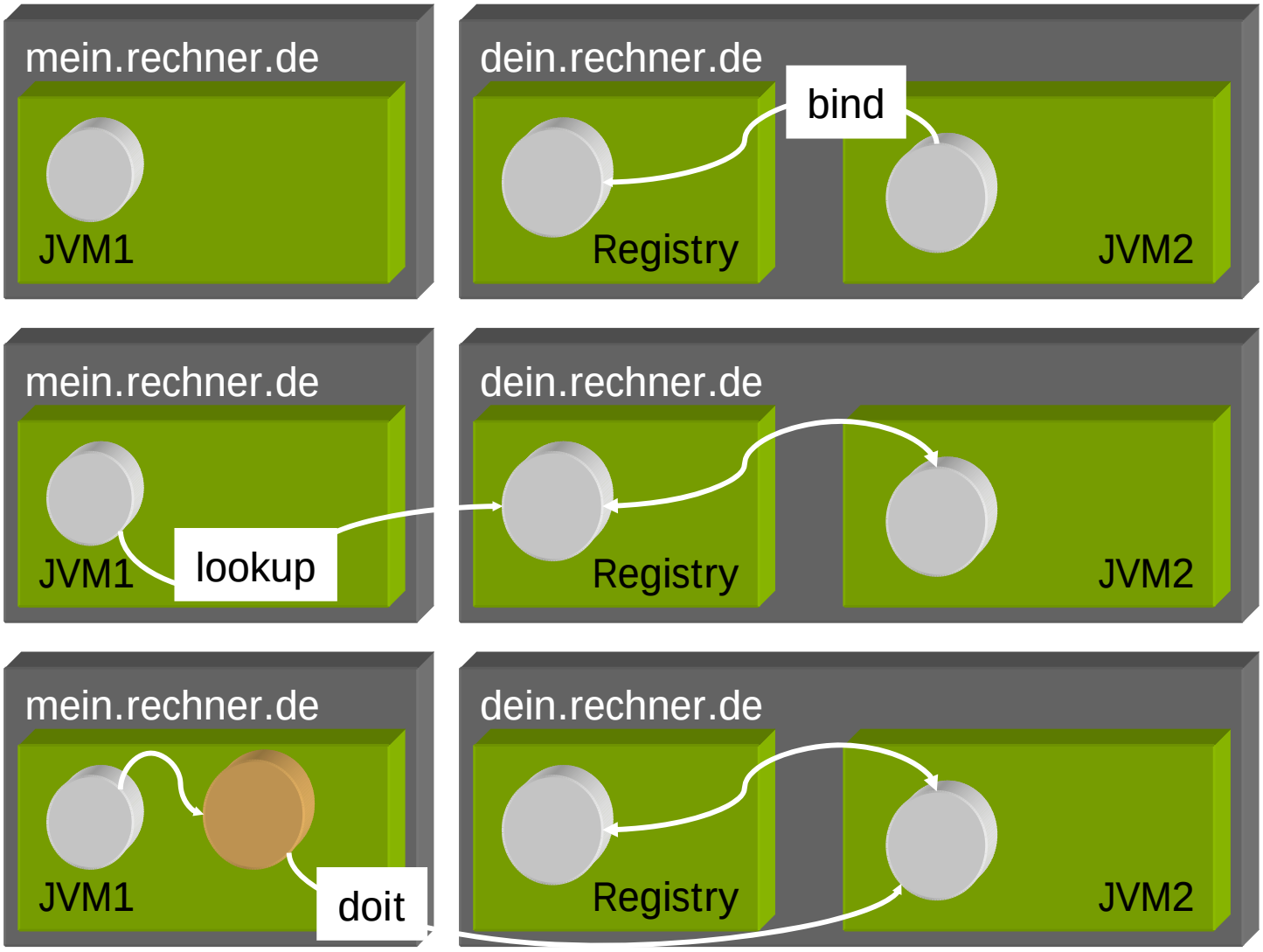


Lokale vs. verteilte Objekte

Lokales Objektmodell	*Verteiltes Objektmodell*
Aufruf an *Objekten*	Aufruf an *Interfaces*
Parameter und Ergebnisse als *Referenzen*	Parameter und Ergebnisse als *Kopien*
Alle Objekte fallen *zusammen* aus	*Einzelne* Objekte fallen aus
Keine Fehlersemantik	*Komplizierte* Fehlersemantik (Referenzintegrität, Netzfehler, Sicherheit etc.)
...	

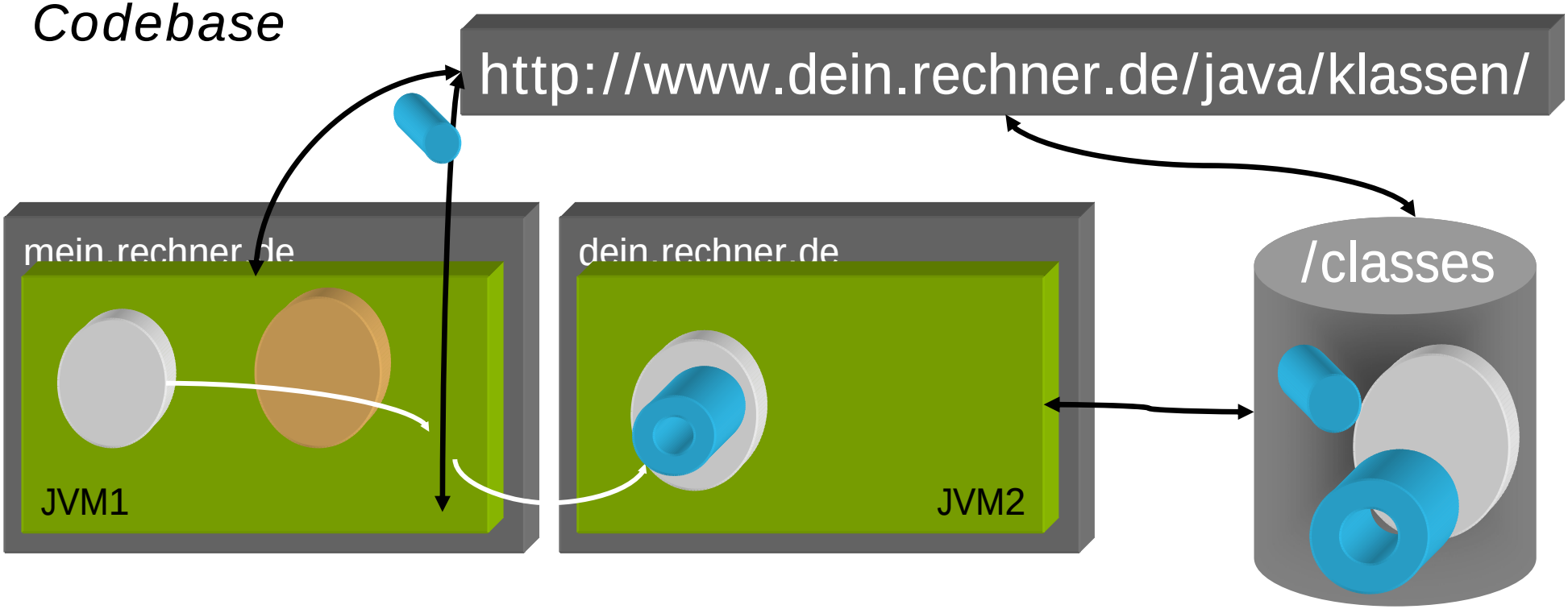
Referenzen auf entfernte Objekte

- *Registry* Objekt liefert Referenzen auf Objekte



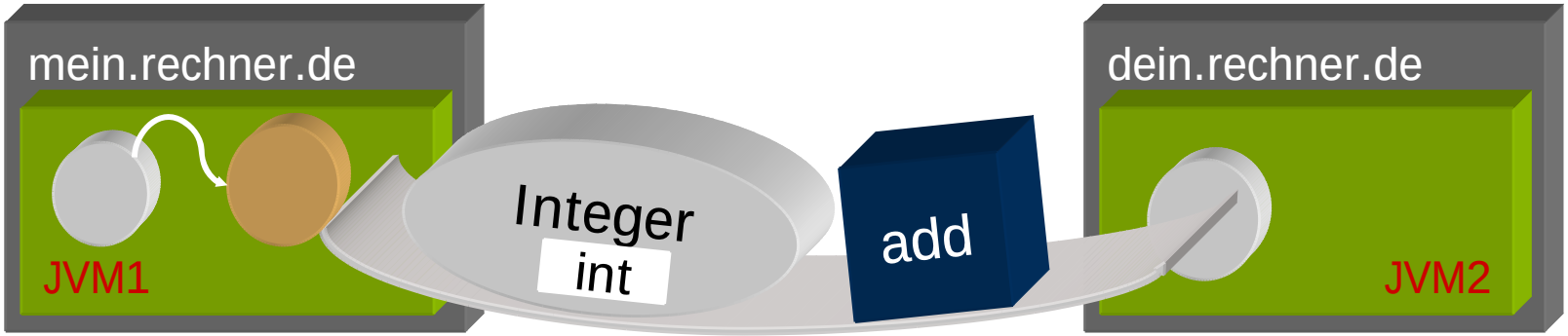
Nachladen über Web-Server

- Wenn JVM1 und JVM2 in getrennten Dateisystemen arbeiten nutzt CLASSPATH nichts
- ServerKlasse_stub.class muss über das Netz nachgeladen werden
- Dies geschieht durch Angabe einer Basis-URL, der *Codebase*



Serialisierung: Klassen nachladen

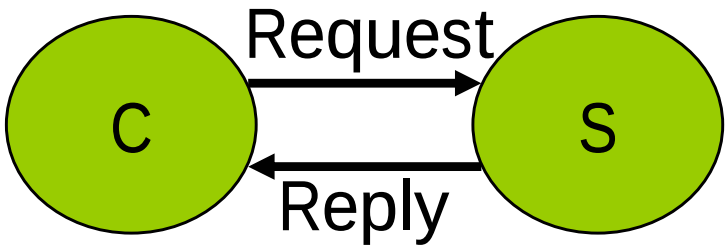
- Objekte = Daten und Verhalten
- Serialisierte Java-Objekte: Datenstrom + Klasse (konzeptionell)



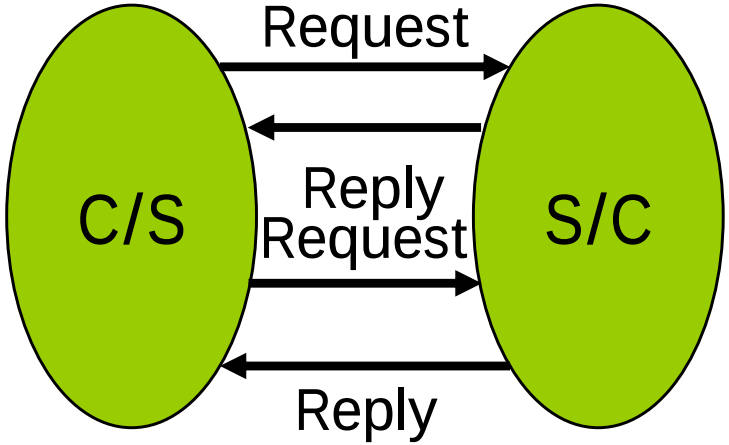
- Klassen zu übermittelten Objekten nachladen, falls nicht
  - vorher nachgeladen
  - vorher schon vorhanden (java.* Klassen)
- Bedrohung durch Angreifer:
  - Bildet Unterklasse von Street, ändert dabei toString()
  - Erzeugt Objekt davon
  - Übergibt Objekt als Argument beim Methodenaufruf
  - Beim Aufruf von toString() dort wird geänderter Code ausgeführt
  - Mit allen Rechten des RMI Objekts

Callbacks

- Zwischen Client und Server-Objekt ist eventuell ein komplexeres Protokoll notwendig
- Client/Server folgt Anfrage/Antwort Protokoll

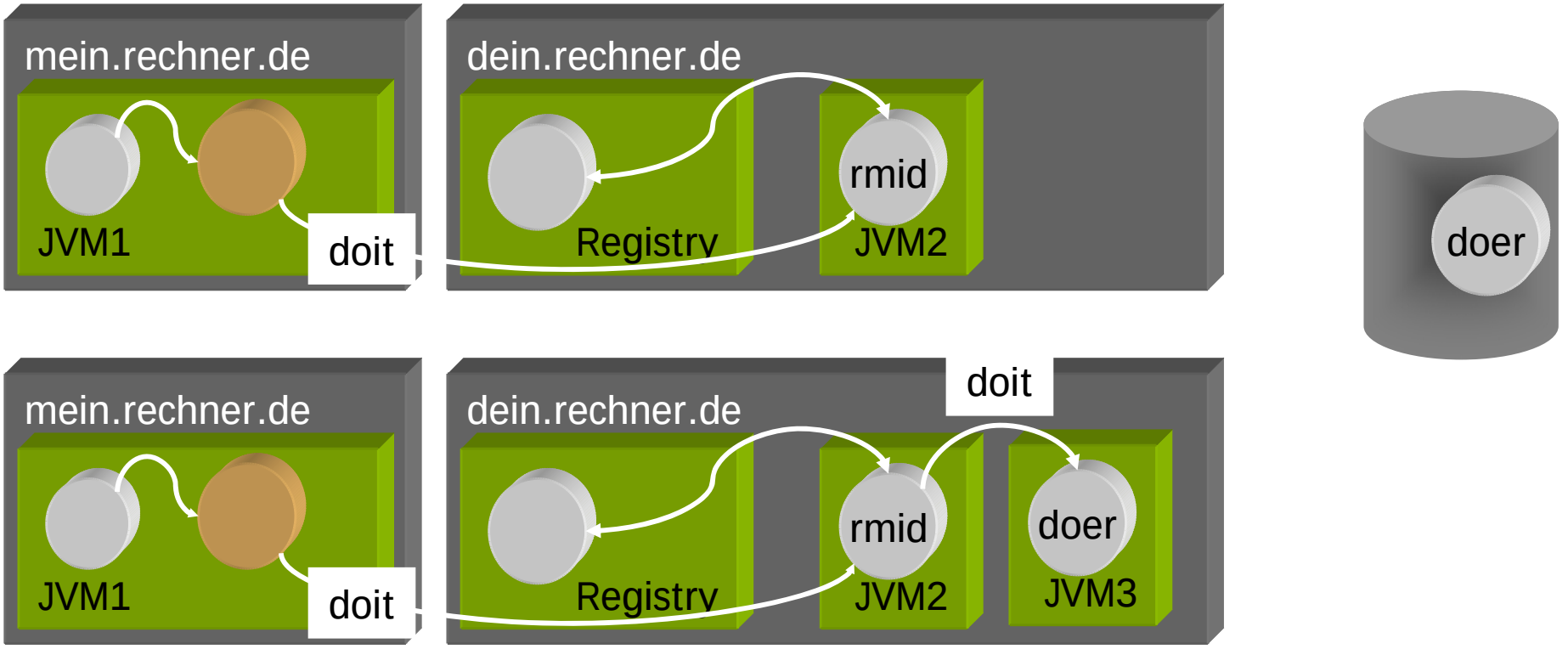


- Für Anfrage/Nachfrage/Antwort Protokoll muss Client selber zum Server werden



RMI Aktivierung

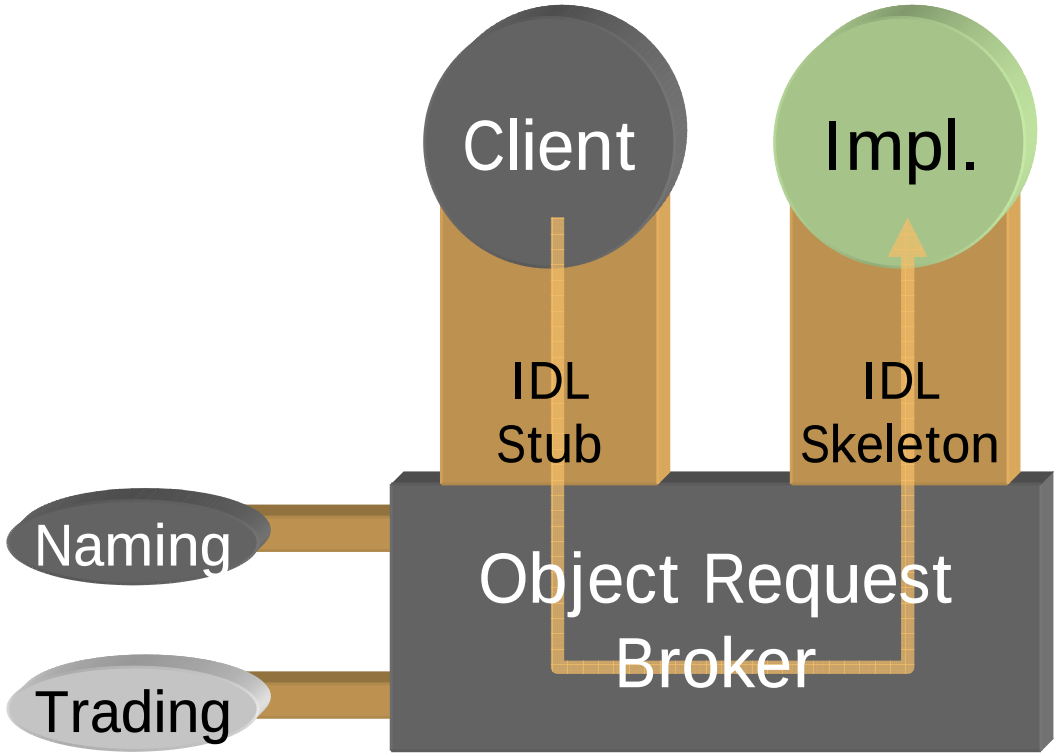
- Mit RMI Aktivierung werden Objekte in einer eigenen JVM beim Aufruf gestartet
- Zuständig für Aktivierung: rmid Programm
- rmid registriert sich für das Objekt und kann es aktivieren



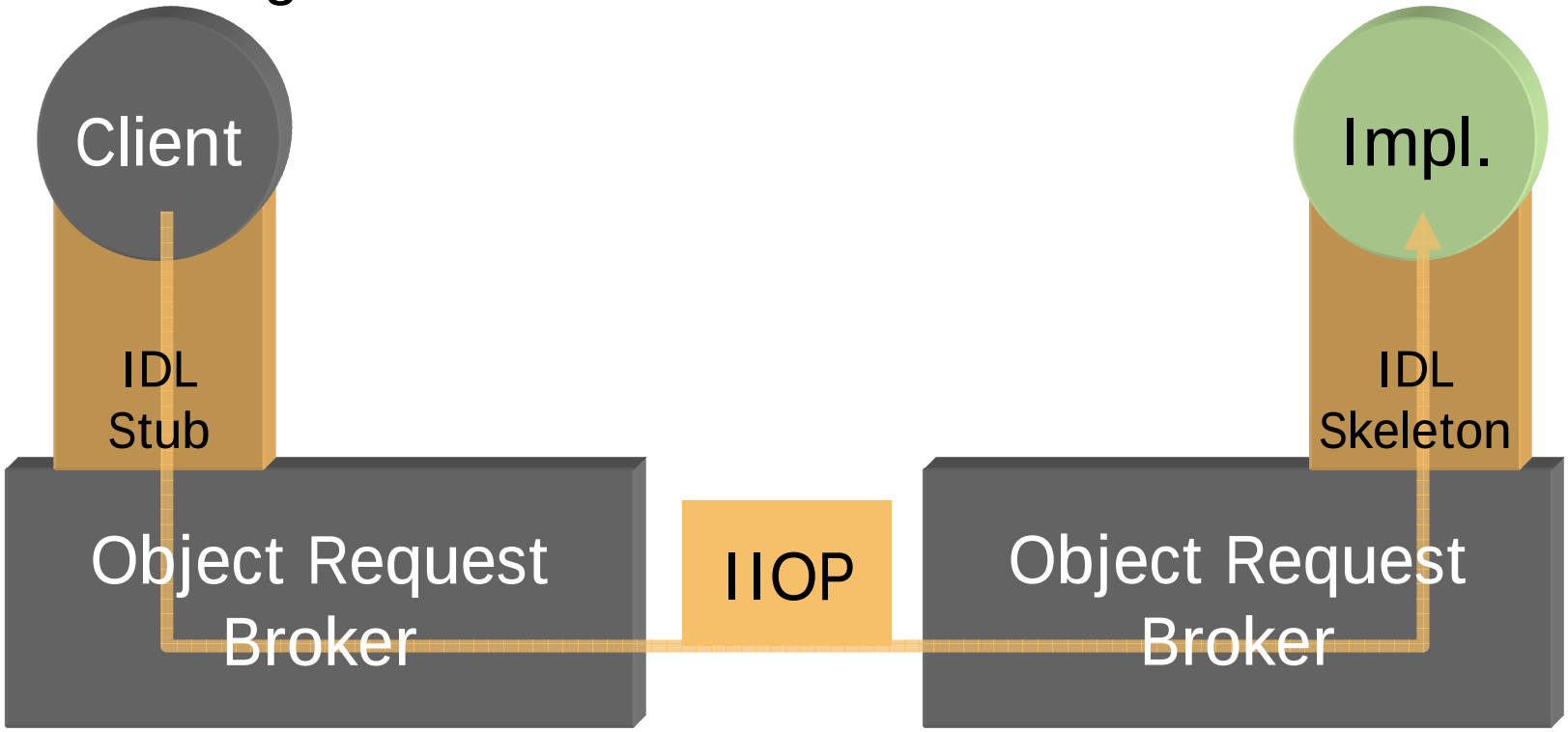
CORBA

OMG

- CORBA Objekte sind Bausteine von Anwendungen
- Sie haben eine typisierte Schnittstelle
- Von der Schnittstelle getrennte Implementierungen in unterschiedlichen Programmiersprachen
- Aufrufe werden durch Object Request Broker transportiert
- Weitere Dienste unterstützen



- Ortstransparenz wird durch Internet Inter-ORB Protocol IIOP erzeugt



- Interworking zwischen ORBs unterschiedlicher Hersteller möglich

RMI vs. CORBA

RMI	*CORBA*
Eine Sprache	Sprachunabhängig
keine Mappings nötig	Mappings notwendig
Objekte können als Argumente verwendet werden (Weil Klassen nachladbar sind)	Nur Referenzen auf Objekte möglich
Ein Laufzeitsystem	Unterschiedliche Laufzeit-systeme (ORBs) integriert
Proprietär (?) „Eigentümer“: Sun Prozess: JCP	Offener Standard (?) „Eigentümer“: OMG Prozess: OMG

- In Java sind mehrere „Middlewares“ integriert (RMI, CORBA, EJB, ...)

IDL

- IDL ist eine Sprache zur Definition von Schnittstellen
- Sprachunabhängig
- Semantik der Typen definiert
- Zur Nutzung Mapping zu konkreter Sprache notwendig
- OMG definiert solche Mappings für unterschiedlichste Sprachen
- Abbildungsprobleme müssen zur Laufzeit durch Ausnahmen signalisiert werden

Praktische Verwendung

1. ORB
   1. Initialisieren
   2. POA erfragen
2. POA
   1. Aktivieren
   2. Mit Policies initialisieren
3. NameService
   1. Als Standarddienst auffindbar
   2. Namenskontexte und -bindungen
4. Ausführung
   1. Über Namensdienst verbunden
5. Persistente Serverobjekte
   1. Durch Policy Aktivierung transparent

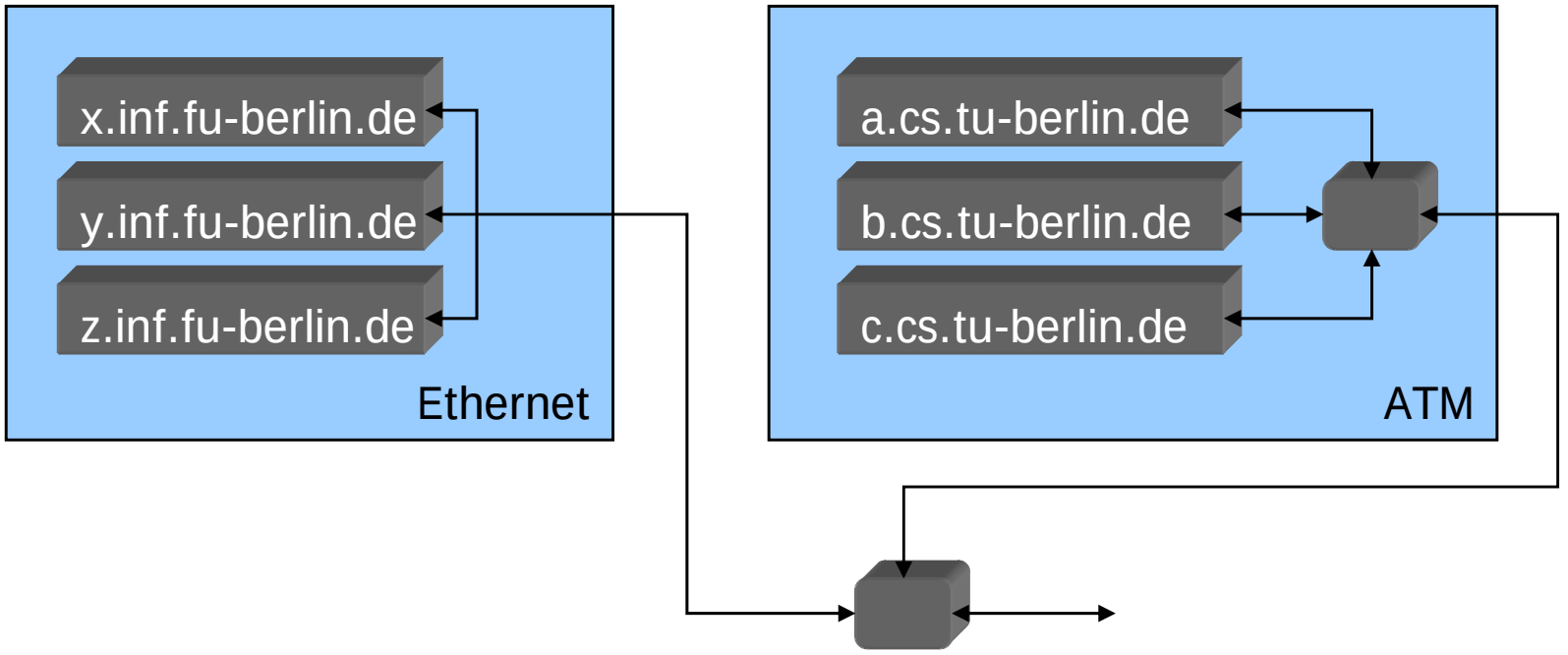
Freie Universität Berlin

- Sockets zur Kommunikation
  Basis der Kommunikation im Internet und ihre Nutzung
- Internet Dienste und ihre Nutzung
  Basis- und Anwendungsdienste im Internet und ihre Nutzung

Internet-Kommunikation Sockets

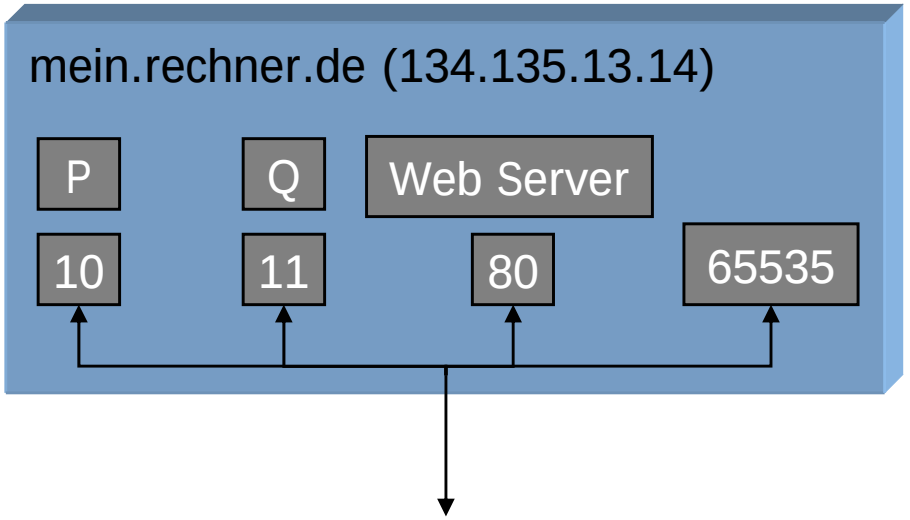
Internet als vernetzter Rechnerverbund

- Das Internet Protokoll IP ermöglicht Internetworking durch Etablierung eines Datenformats und Transportprotokollen, die auf unterschiedlichen Datenverbindungen aufgesetzt werden können



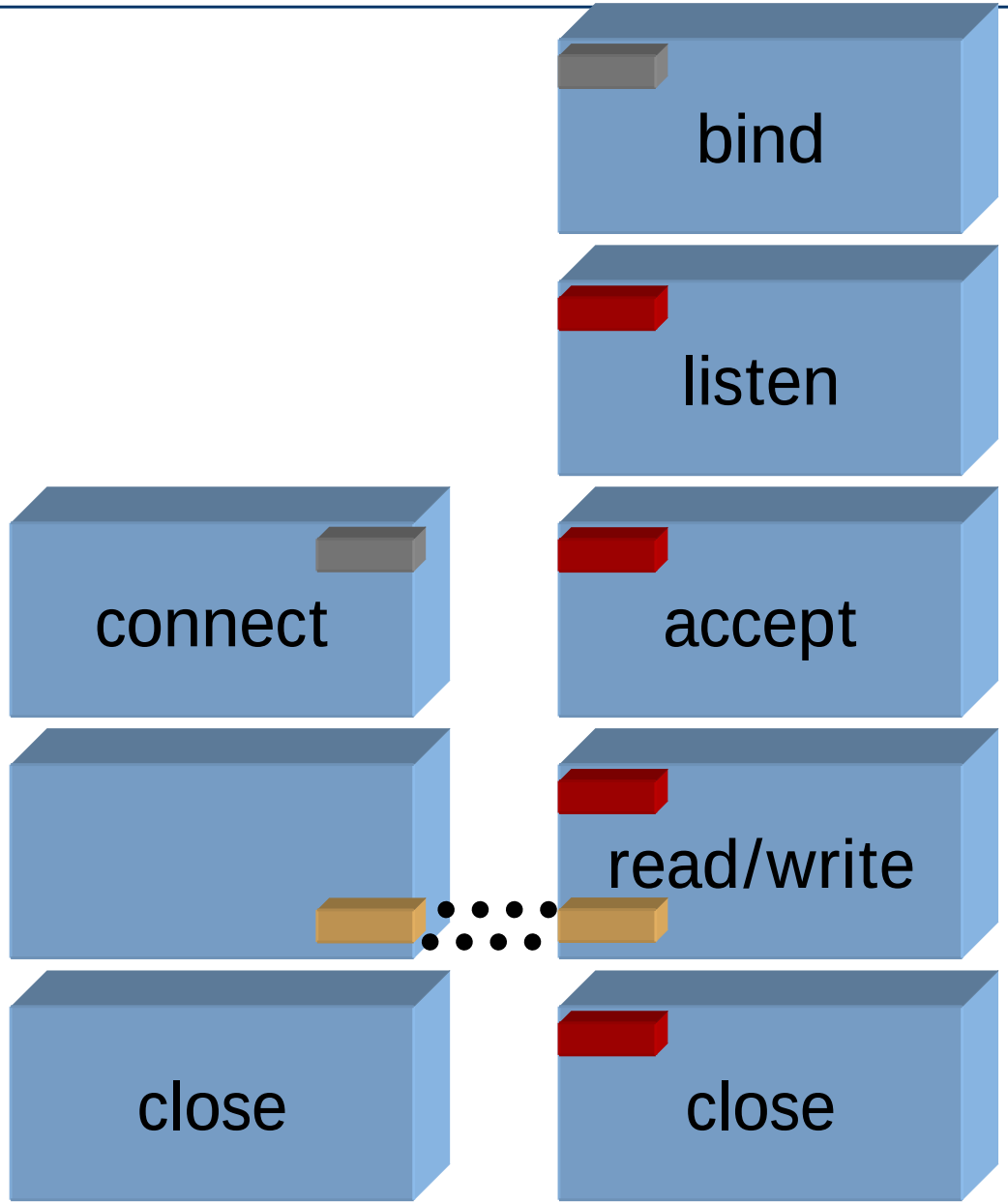
Transport Protokolle

- Zwei Protokolle zum Datentransport
  - *UDP*: Ein Paket (Datagram) von Rechner A nach Rechner B schaffen – Verbindungslos
  - *TCP*: Pakete werden *geordnet* und *zuverlässig* über eine Verbindung transportiert – Verbindungsorientiert
- Ports als Kommunikationsadresse
  - Ein *Port* ist ein logischer Netzanschluss, benannt von 0 bis 65535
- *Socket* ist Endpunkt einer Verbindung



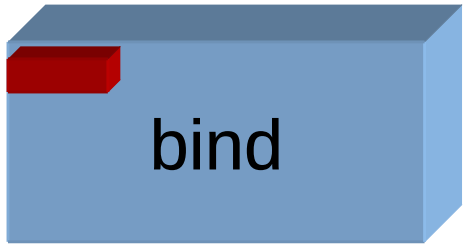
TCP Sockets

1. Server reserviert Port
2. Server nimmt Verbindungswünsche an
3. Client schickt Verbindungswunsch
4. Client und Server sind verbunden
   bidirektional!
5. Verbindung wird abgebaut

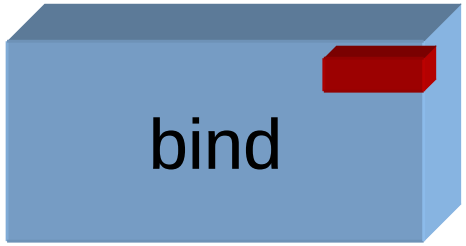


UDP Sockets

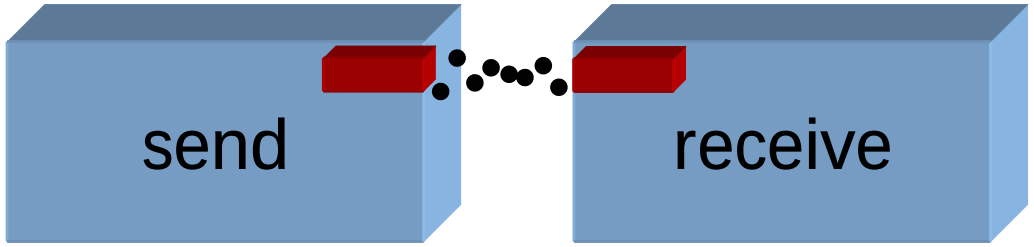
1. Server bindet Socket



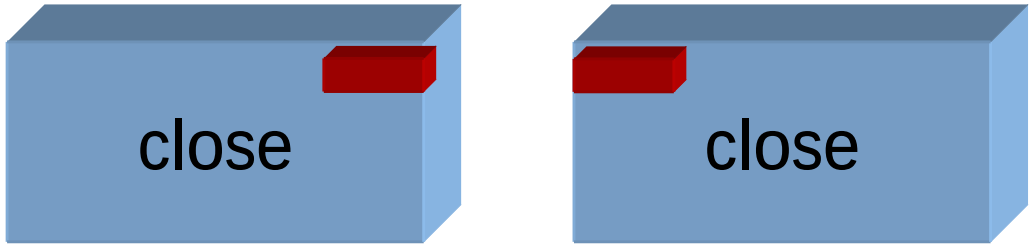
2. Client bindet Socket



3. Client und Server
   senden und empfangen
   bidirektional!

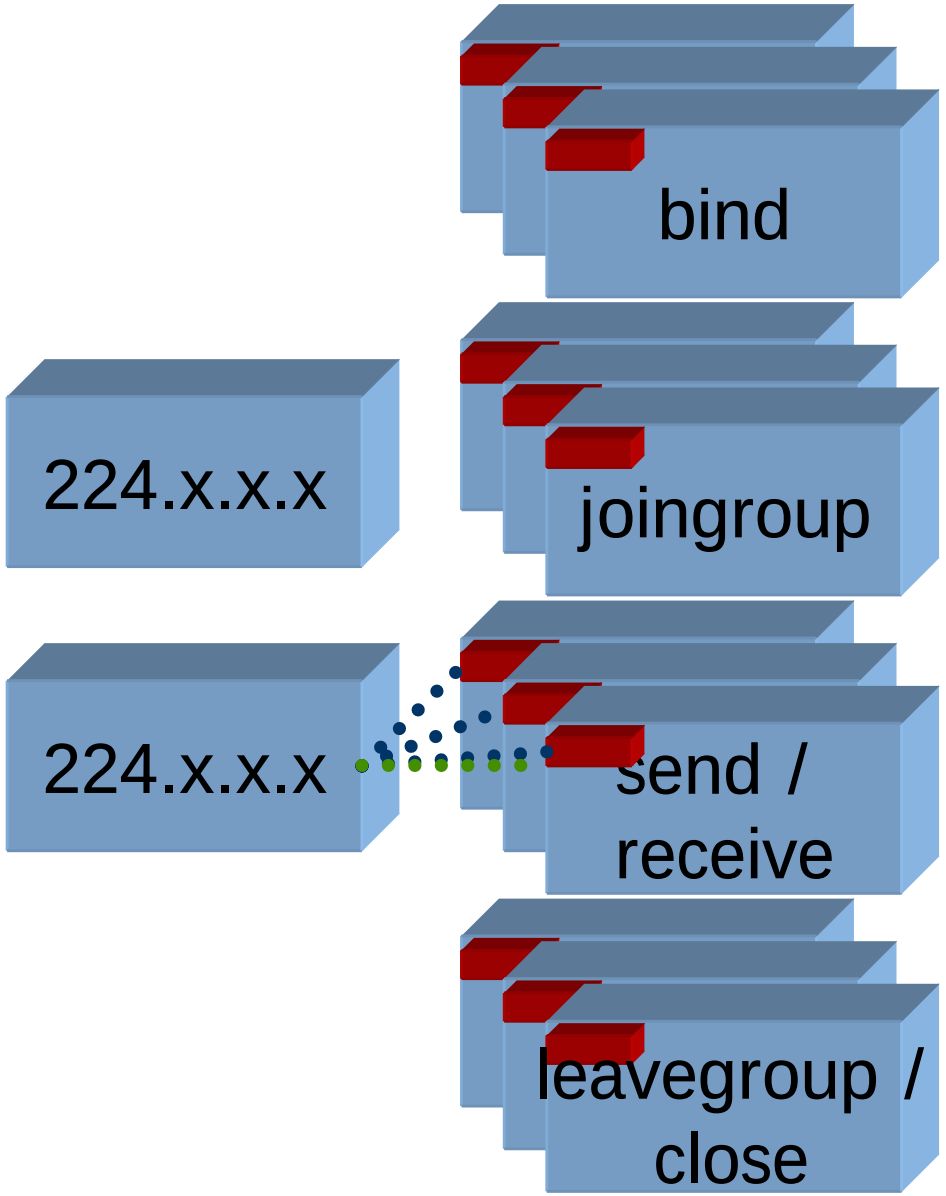


4. Sockets werden
   aufgegeben



UDP Gruppen Sockets

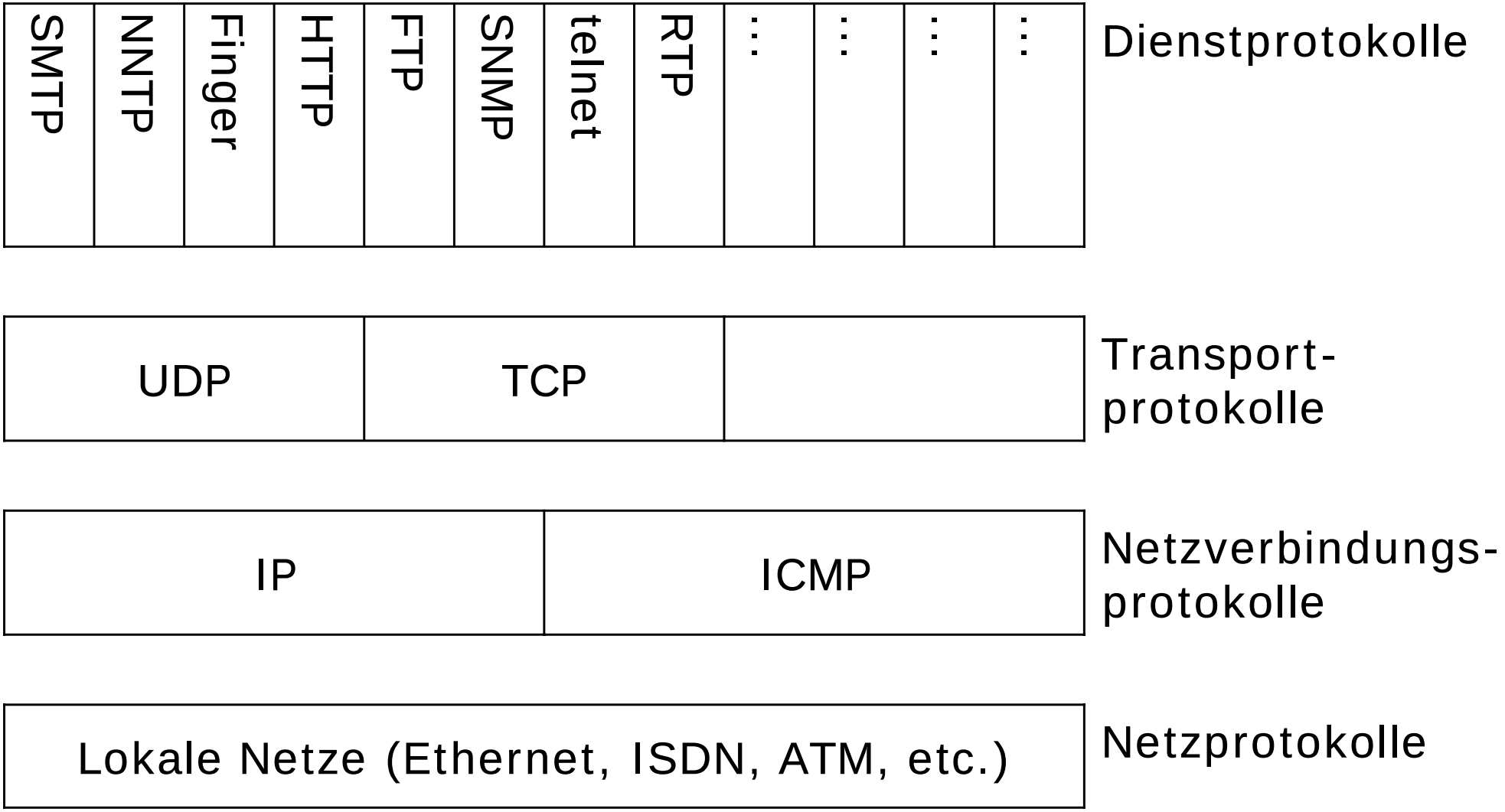
1. Teilnehmer binden Sockets
2. Teilnehmer treten Gruppe bei
3. Teilnehmer senden und empfangen
4. Teilnehmer verlassen Gruppe und geben Socket auf



Internet Dienste

Internet-Protokolle und -Dienste

- Einordnung von Internet-Protokollen:



Internet als dienstorientiertes offenes System

Freie Universität Berlin

- Internet Dienste sind (zumeist) definiert durch
  - Aufgabe
  - Portnummer auf dem der Dienst angeboten wird
  - Transportprotokoll (TCP oder/und UDP)
  - Protokoll

- Z.B.: Web Dienst
  - Übertragen von HTML Seiten
  - Port 80
  - TCP
  - HTTP

- Z.B.: Usenet Dienst
  - Übertragen von News
  - Port 119
  - TCP
  - NNTP

Simple Mail Transfer Protocol SMTP

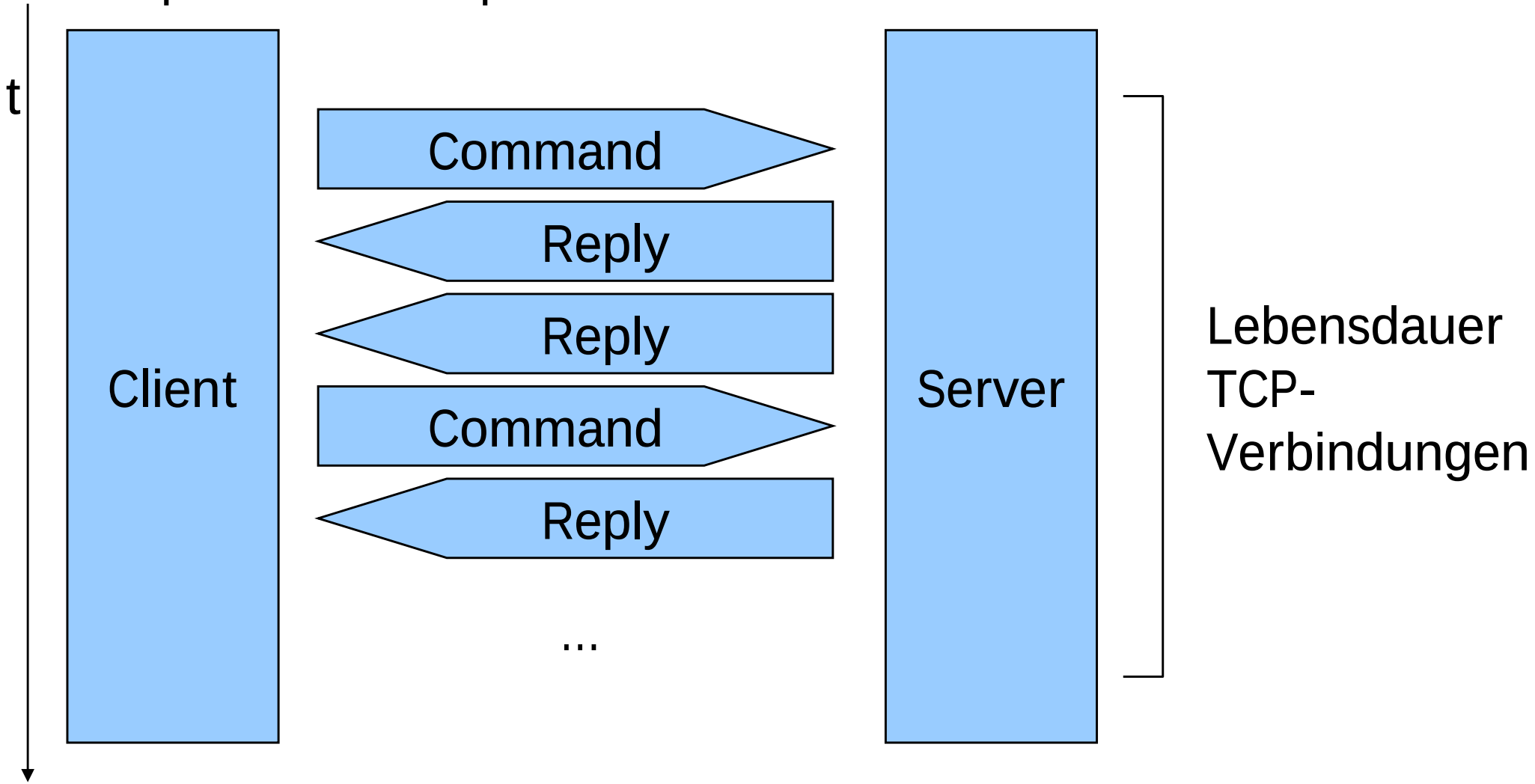
Freie Universität Berlin

- Aufgabe:
  Transfer von Mails zwischen Mail-Client beim Absender und Mailserver beim Empfänger
- Ports:
  25 zur kompletten Protokollabwicklung
- Transportprotokoll:
  TCP
- Protokoll:
  J. Klensin, Editor. *Simple Mail Transfer Protocol*, April 2001. RFC 2821, http://www.ietf.org/rfc/rfc2821.txt

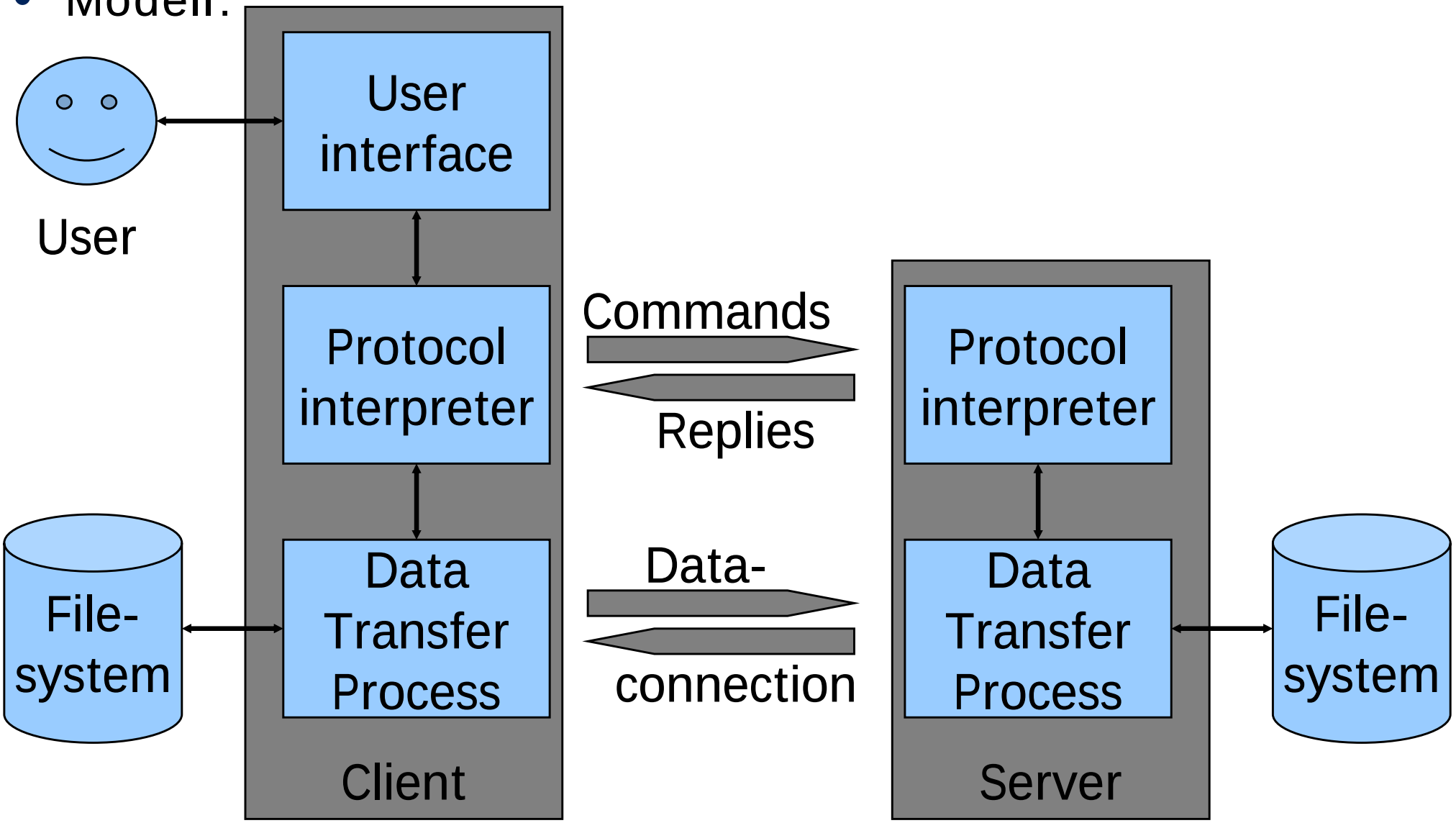
File Transfer Protocol

- Aufgabe:
  Transfer von Dateien zwischen FTP-Servern und Clients
- Ports:
  21 ist für FTP Kontrollverbindung reserviert
  Weitere Ports sind für FTP Datenverbindung reserviert
- Transportprotokoll:
  TCP
- Protokoll:
  J. Postel und J. Reynolds. *FILE TRANSFER PROTOCOL (FTP)*, Oktober 1985. RFC 959, http://www.ietf.org/rfc/rfc959.txt

- Zustandshaltiges Protokoll
- Request mit Response beantwortet



- Modell:


User
User interface
Protocol interpreter
Data Transfer Process
Client
Commands
Replies
Data-
connection
Protocol interpreter
Data Transfer Process
Server
File-system
File-system

III. Web Programmierung

Freie Universität Berlin

- HTTP Nutzung
- Kommunikation mit Web-Servern
- HTML
- XML und XML Verarbeitung
- Klienten- und serverseitige Programmierung

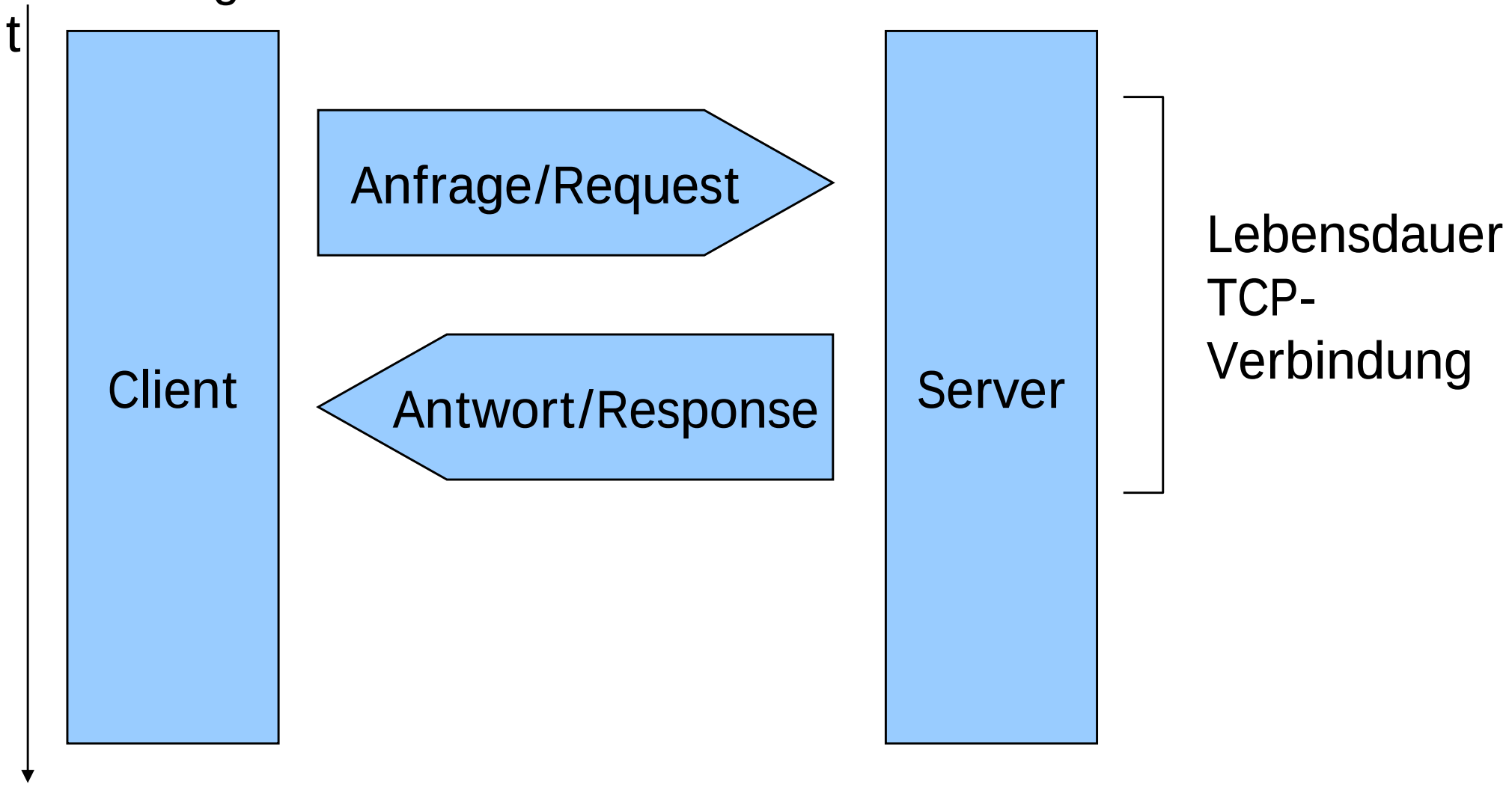
HTTP

Hypertext Transfer Protocol

- Aufgabe:
  Transfer von Informationen zwischen Web-Servern und Clients
- Port:
  80 ist für HTTP reserviert
- Transportprotokoll:
  TCP
- Protokoll:
  R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach und T. Berners-Lee. *Hypertext Transfer Protocol - HTTP/1.1*. RFC 2616, http://www.ietf.org/rfc/rfc2616.txt

HTTP

- Zustandsloses Protokoll
- Anfrage mit Antwort beantwortet



Aufbau Anfrage

- Anfrage besteht aus
  - Anfragemethode
  - Anfragebeschreibung durch Kopfzeilen
    - Allgemeine Beschreibungen
    - Anfragespezifische Beschreibungen
    - Beschreibung eventuell beiliegenden Inhalts
  - Leerzeile
  - Eventueller Inhalt
- Beispiel:

```
GET / HTTP/1.0
Connection: Keep-Alive
User-Agent: Mozilla/3.04Gold (Win95; I)
Host: megababe.isdn:80
Accept: image/gif, image/jpeg, image/pjpeg, */*
```

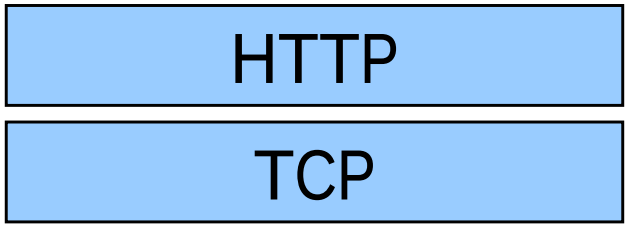
Aufbau Antwort

- Antwort besteht aus
  - Antwortcode
  - Antwortbeschreibung durch Kopfzeilen
    - Allgemeine Beschreibungen
    - Antwortspezifische Beschreibungen
    - Beschreibung eventuell beiliegenden Inhalts
  - Leerzeile
  - Eventueller Inhalt
- Beispiel:

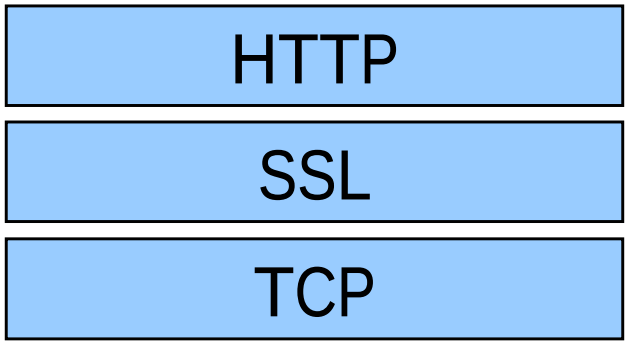
```
HTTP/1.0 200 OK
Last-Modified: Sun, 15 Mar 1998 11:26:50 GMT
MIME-Version: 1.0
Date: Fri, 20 Mar 1998 16:43:11 GMT
Server: Roxen-Challenger/1.2beta1
Content-type: text/html
Content-length: 2990

<HTML><HEAD><TITLE>TU Berlin ---
```

HTTP über SSL Sockets

- HTTP benutzt TCP Sockets zur Kommunikation



- Secure Sockets Layer SSL erweitert Sockets um Sicherheitsmerkmale
- HTTPS bezeichnet eine HTTP Kommunikation über solche sicheren Sockets

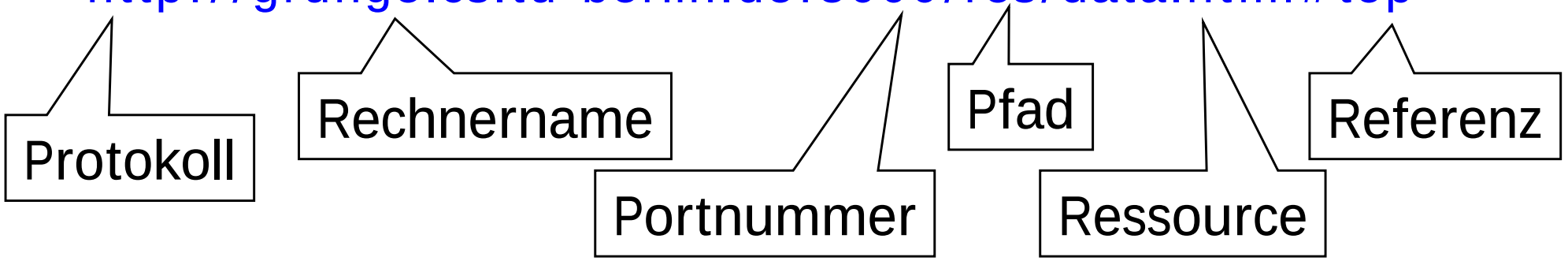


- Port 443 als Default-Port festgelegt

URL

- URL Schemas sind für Internet-Dienste definiert und vereinheitlichen damit deren Nutzung syntaktisch:
  - http://grunge.cs.tu-berlin.de:8000/
  - ftp://ftp.cs.tu-berlin.de/pub/net/www
  - mailto:tolk@inf.fu-berlin.de

- Form:
  http://grunge.cs.tu-berlin.de:8000/res/data.html#top



- Bedeutung ist von Schema abhängig, URL ist nur als Syntax definiert

java.net.URLConnection

- Abfragen der Ressource und Eigenschaften
  - Object getContent()
  - String getHeaderField(String name)
  - InputStream getInputStream()
  - OutputStream getOutputStream()
- Übliche Header
  - getContentEncoding()
  - getContentLength()
  - getContentType()
  - getDate()
  - getExpiration()
  - getLastModifed()
- Sind unter Umständen errechnet oder leer

Eigene URL Schemata

- URL Architektur in Java ist erweiterbar
- Dazu müssen definiert werden
  - eine Klasse Handler
  - eine Klasse *Schema*URLConnection
- Sie müssen in einem Paket stehen:
  package de.fuberlin.inf.nbi.*schema*
- Durch die Property java.protocol.handler.pkgs muss dem Laufzeitsystem mitgeteilt werden wo die eigenen Klassen stehen

```
java -Djava.protocol.handler.pkgs=de.fuberlin.inf.nbi
    GetURL daytime://np.ag-nbi.de
```

Client-Pull und Server-Push

- HTTP Interaktion mit Anfrage und Antwort wird durch Client-Pull und Server-Push erweitert
- Client-Pull
  - Client lädt Inhalte in regelmäßigen Abständen nach
  - Server löst das Verhalten durch zusätzlichen Header aus
- Server-Push
  - Server schickt mehrere Antworten nacheinander
  - Client ersetzt jeweils darstellung

Interaktion zur Authentifizierung

- Seiten im Web können Zugriffsschutz tragen
- Interaktion zum Abruf
  - Normales GET
  - Antwort 401 und WWW-Authenticate: Header, der Nachweis in unterschiedlichen Schemata anfordert
  - Weiteres GET mit Authorization: Header, der je nach Schema Parameter trägt
  - Antwort 200

Parameter für Web-Skripte

- Zwei Arten der Übermittlung von Parametern an Skripte:
  - GET: Daten werden in URL kodiert
  - POST: Daten werden kodiert über Standardeingabe geliefert
- HTML:

```
<html><body>
<form action="http://flp.cs.tu-berlin.de/~tolk/echo.cgi"
   method="get"> <input name="Eingabe" type="text">
   <input type="submit" value="Per GET">
</form>
<form action="http://flp.cs.tu-berlin.de/~tolk/echo.cgi"
   method="post">
  <input name="Eingabe" type="text">
   <input type="submit" value="Per POST">
</form>
</body>
</html>
```

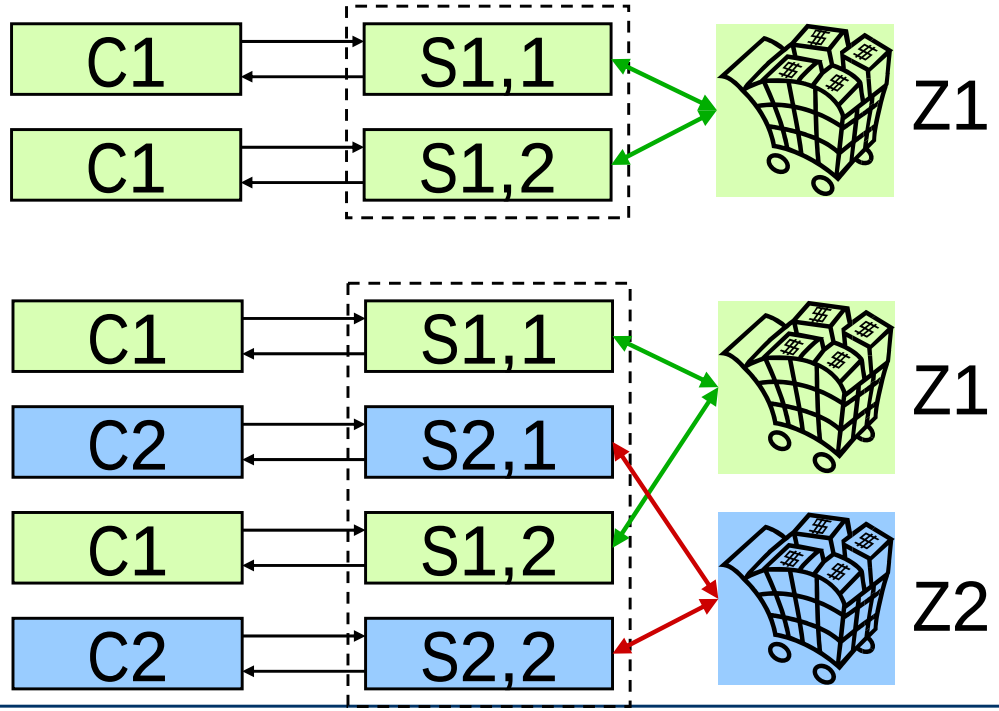
Kodierung von Eingabewerten

Freie Universität Berlin

- Parameter haben bestimmten Übergabeformat
  - $Name_1=Wert_1\&Name_2=Wert_2$
- Dieser *Query String* wird
  - bei GET an die URL mit ? getrennt angehängt
    - http://flp.cs.tu-berlin.de/~tolk/echo.cgi?Eingabe=Hallo!
  - bei POST als Inhalt übermittelt und beim Web-Server über stdin einem Skript übergeben
- Query String selber muss kodiert werden
  - Um Transport zu sichern
  - Um bedeutungstragende Zeichen (=, & etc.) übertragen zu können
  - Medientyp der Nachricht ist application/x-www-form-urlencoded

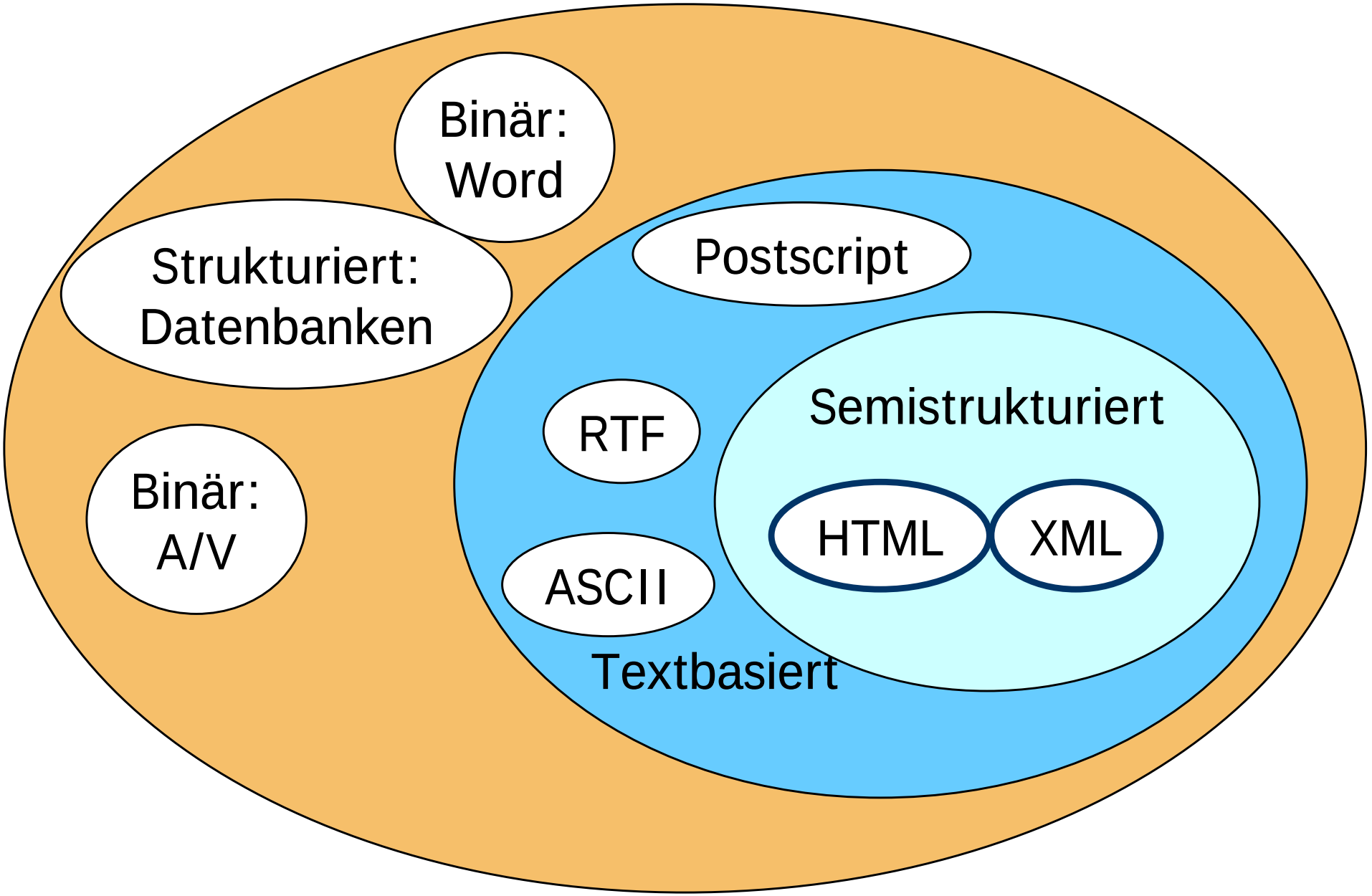
Sessions

- Einführung von Sitzungen (Sessions)
- Sitzung: Folge von Interaktionen, die einen gemeinsamen Zustand haben
- Identifikation in der Interaktion durch eindeutige Sitzungsnummer (Session-ID)
- Ermittlung des Zustand auf Basis der Session-ID



HTML

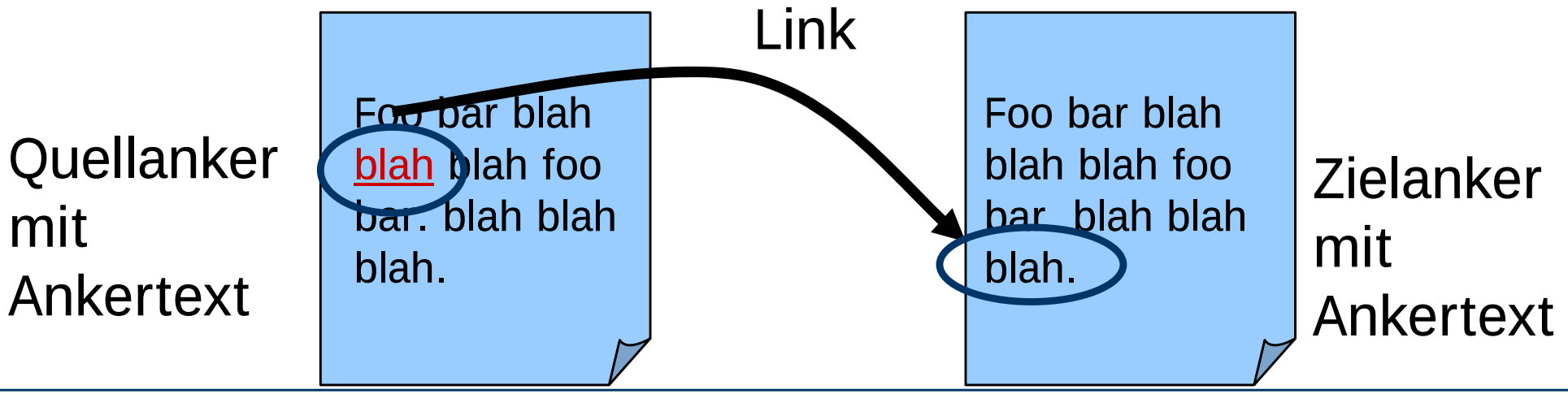
Daten im Netz



Hypertext Markup Language

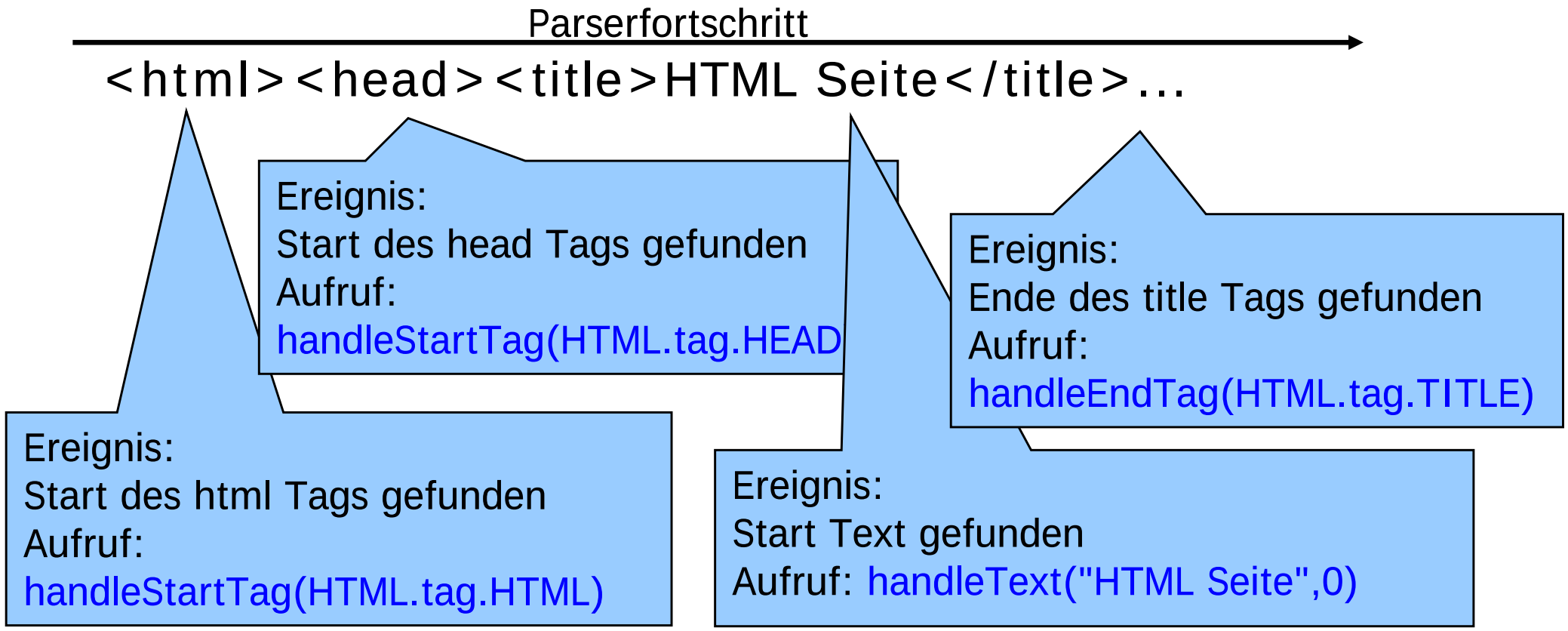
- Dominierende Sprache zur Auszeichnung von Dokumenten im Internet
- Definiert vom World Wide Web Consortium, W3C:
  - MIT (Massachusetts Institute of Technology, Computer Science and Artificial Intelligence Laboratory (CSAIL))
  - ERCIM (European Research Consortium in Informatics and Mathematics)
  - Keio University of Japan
- Jedes Informationssystem im Netz muss:
  - HTML Informationen integrieren können
  - HTML Ausgaben erzeugen
  - Mit HTML-Mitteln mit Nutzern interagieren

- Konzepte:
  - Informationen werden als Dokumente aufgefasst
  - Dokumenteninhalte werden als Klartext dargestellt
  - Dokumententeile werden durch *Markierungen/Elemente/Tags* ausgezeichnet
    - Inhaltlich (<h1>Einleitung</h1>, wichtig)
    - Gestalterisch (wichtig)
  - Dokumente werden durch Links zu einem Hypertext verbunden (dadurch entsteht ein Netz, das Web)



javax.swing.text.html.parser

- Paket javax.swing.text.html.parser enthält einen Parser für HTML
- Basiert auf Rückrufen bei Auftreten einer bestimmten Informationseinheit in HTML:



XML

XML Q&A

- *Was ist XML?*
  Die Extensible Markup Language ist die Definition einer Untermenge von SGML, mit der man einfach Auszeichnungssprachen definieren kann
- *Woher kommt XML?*
  XML ist ein Standard des World Wide Web Konsortiums W3C
- *Was macht man mit XML?*
  Anwendungsspezifische Auszeichnungssprachen definieren und standardisieren
- *Was ist der Vorteil von XML-basierten Auszeichnungssprachen?*
  Standardisierung ermöglicht Datenaustausch

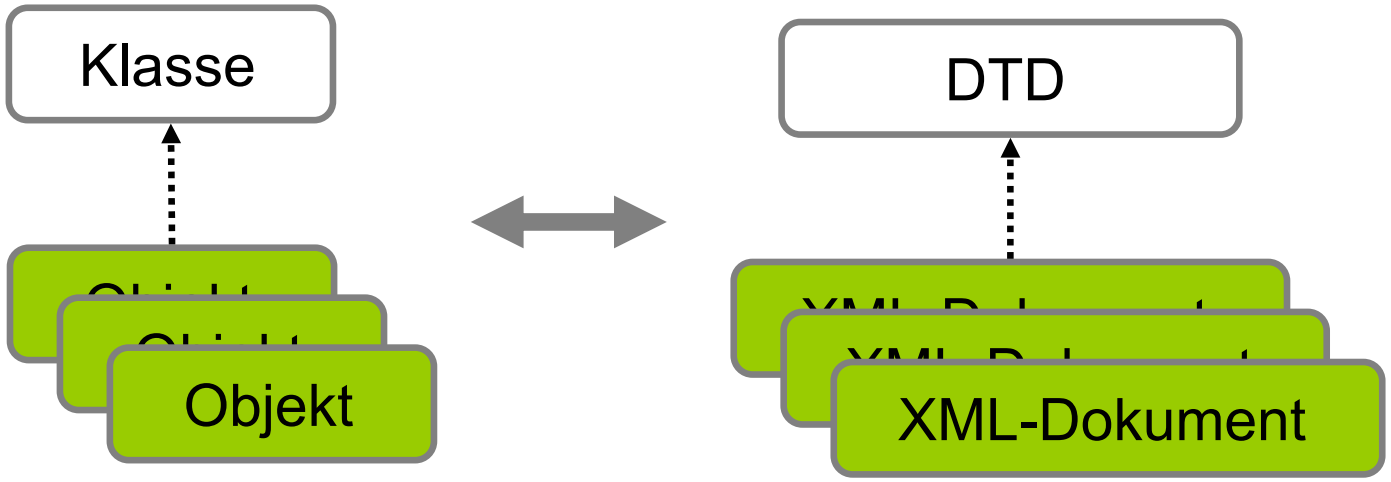
Regeln für wohlgeformte Dokumente

Freie Universität Berlin

- Jedes Anfangs-Tag muss ein zugehöriges End-Tag haben.
- Elemente dürfen sich nicht überlappen.
- XML-Dokumente haben genau ein Wurzel-Element.
- Elementnamen müssen den Namenskonventionen von XML entsprechen.
- XML beachtet grundsätzlich Groß- und Kleinschreibung.
- XML belässt Formatierungen (white space) im Text.

Dokument-Typen

- Beschreiben den prinzipiellen Aufbau von Dokumenten eines bestimmten Typs.
- Können mit DTDs (Document Typ Definitions) spezifiziert werden.
- DTDs wurden von SGML übernommen.
- DTDs benutzen Syntax, die regulären Ausdrücken ähnlich ist.



Deklaration der Elementstruktur

```
<!ELEMENT catalog (CD | cassette | record | MP3)*>
```

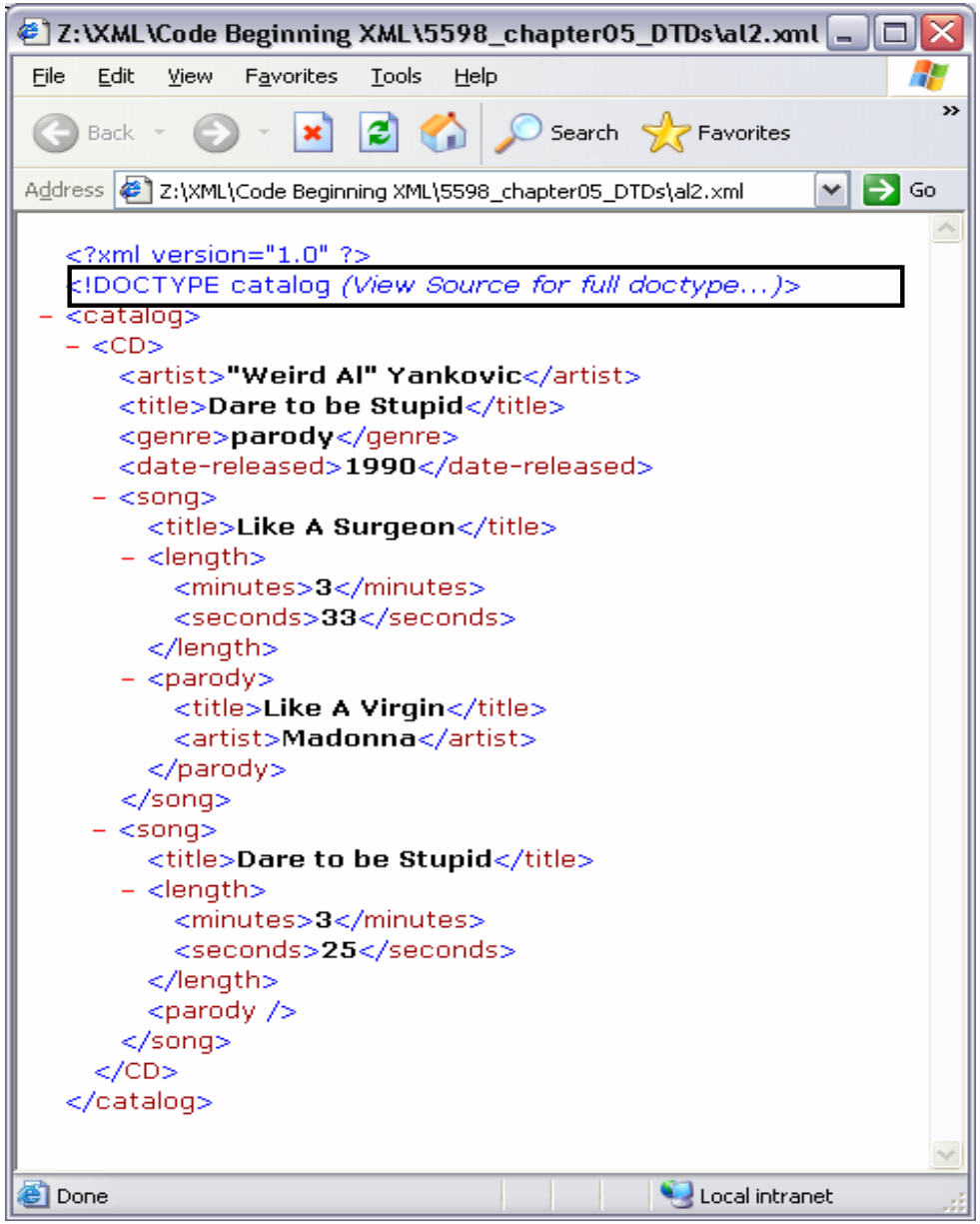
- Deklariert Element catalog
- * bedeutet n Wiederholung mit n ≥ 0.
- | bedeutet Auswahl.
- CD, cassette, record und MP3 sind jeweils Kind-Elemente.

Primitive Datentypen

- Beachte: Elementare Datentypen, wie sie von Programmiersprachen bekannt sind, stehen in DTDs *nicht* zur Verfügung.
- <!ELEMENT minutes (INTEGER)>

Zulässige Dokumente

- In einem XML-Dokument kann ein Dokument-Typ spezifiziert werden.
- Das Wurzelelement des Dokumentes muss genau der Struktur dieses Elementes entsprechen, wie sie im Dokument-Typ festgelegt ist.
- In diesem Fall bezeichnet man das Dokument als zulässig (engl. valid).



Deklaration von Attributen

```
<!ATTLIST catalog
    version CDATA #IMPLIED "1.0">
```

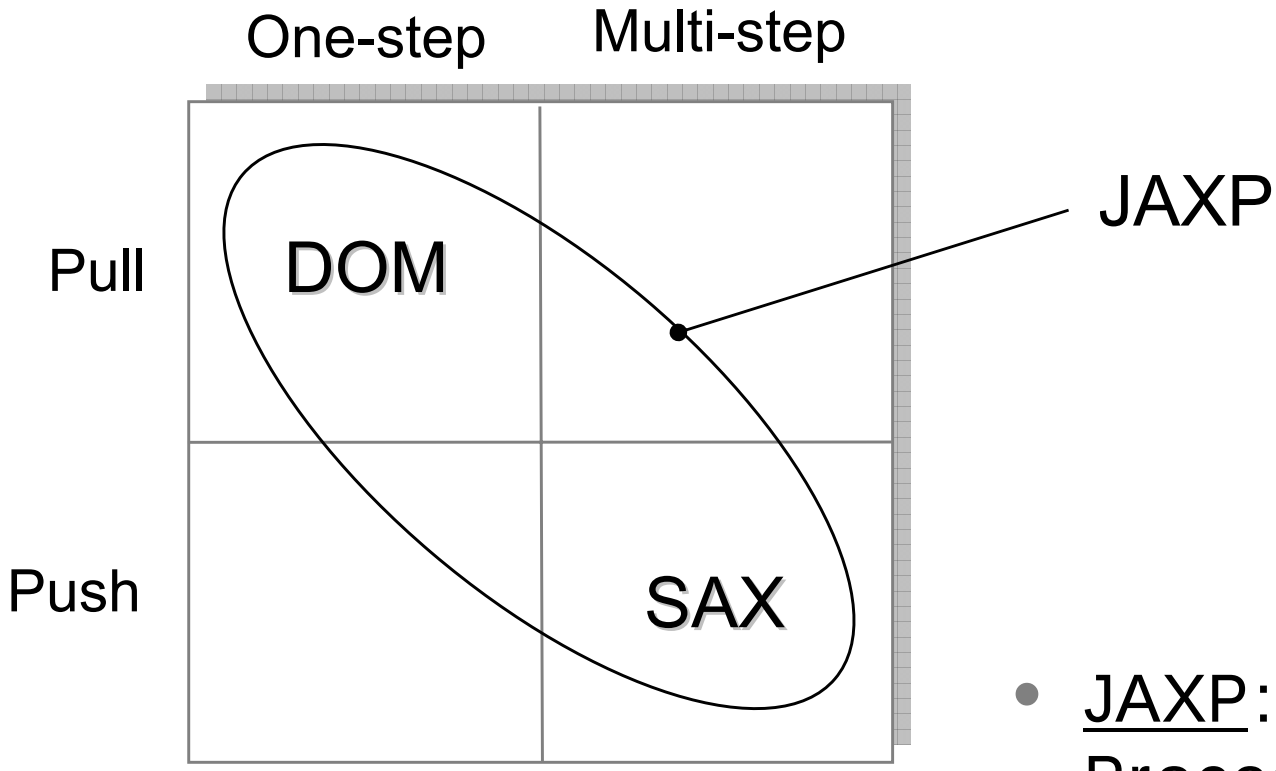
- Das Element catalog hat ein Attribut mit dem Namen version.
- Außer version hat catalog keine weiteren Attribute.
- Das Attribut ist vom Typ String (CDATA).
- #IMPLIED: Das Attribut ist optional.
- "1.0" ist der Standard-Wert.
- #REQUIRED: Das Attribut ist obligatorisch.
- #FIXED: Das Attribut hat immer den gleichen Wert.

DTDs

- Dokument-Typen beschreiben den prinzipiellen Aufbau von Dokumenten eines bestimmten Typs.
- Dokument-Typen können mit Klassen und XML-Dokumente mit Objekten verglichen werden.
- In einem XML-Dokument kann ein bestimmter Dokument-Typ spezifiziert werden.
- Das Wurzelelement des Dokumentes muss dann genau der Struktur dieses Elementes entsprechen, wie sie im Dokument-Typ festgelegt ist.
- In diesem Fall bezeichnet man das Dokument als zulässig (engl. valid).

XML-Parser

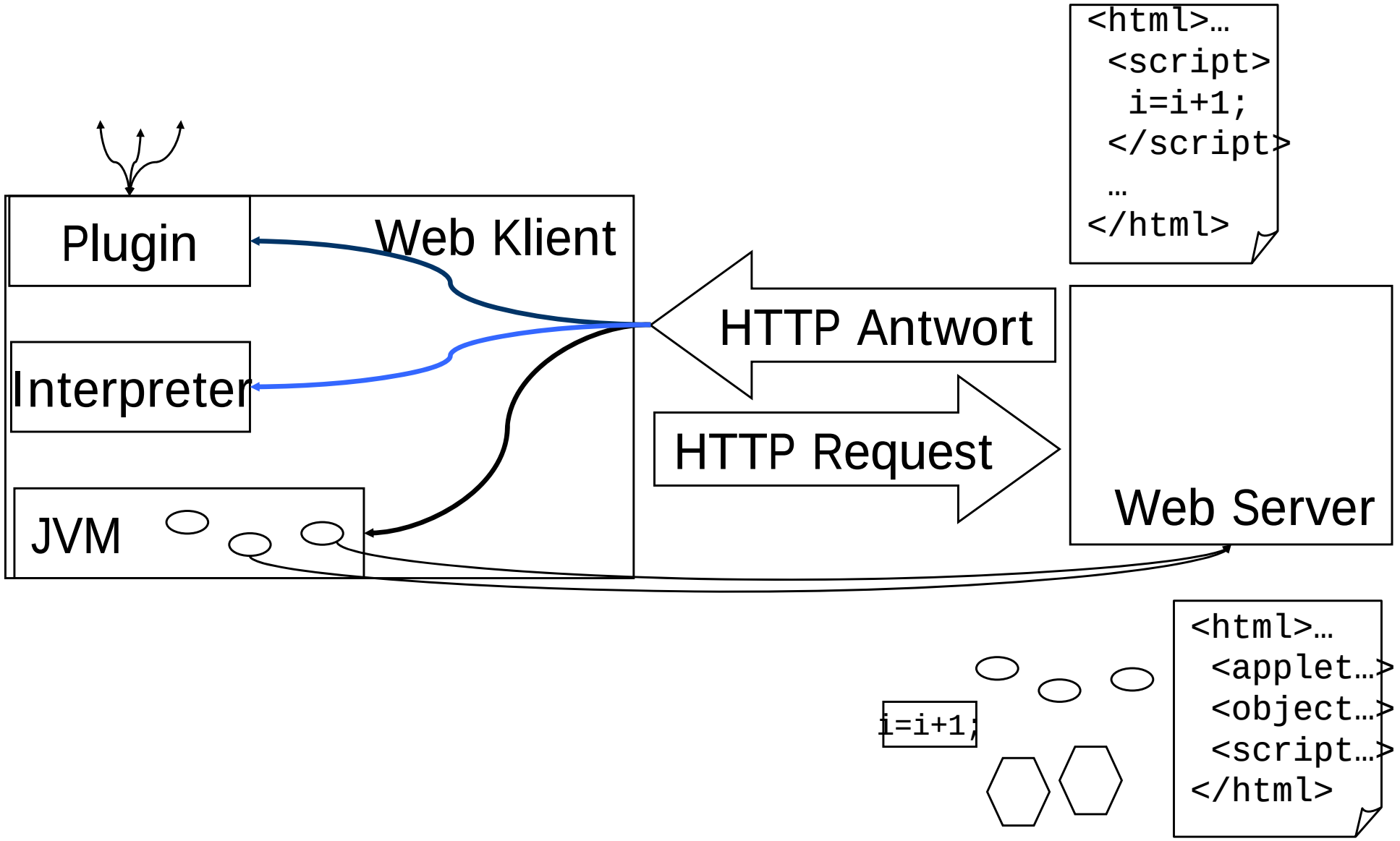
- DOM: Document Object Model
- SAX: Simple API for XML



- JAXP: Java API for XML Processing

Klienten- und serverseitige Verarbeitung

Klientenseitige Verarbeitung



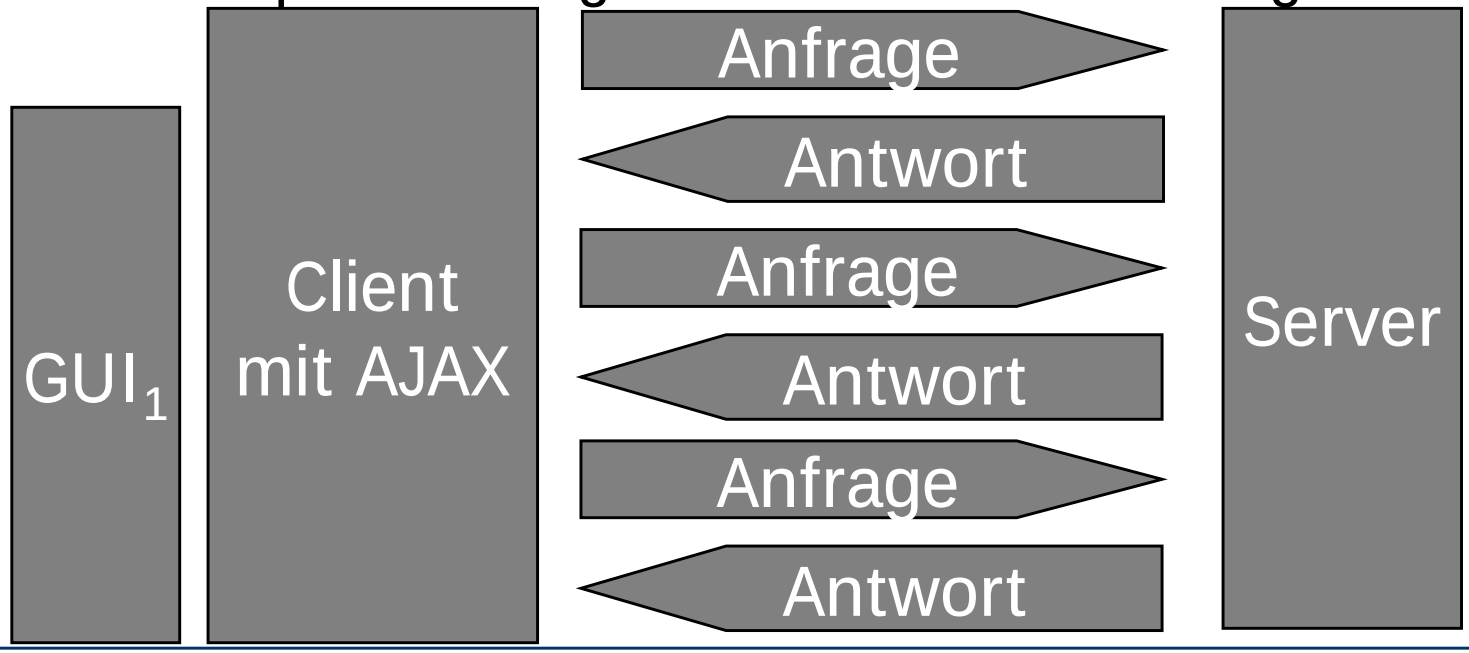
JavaScript Entwicklung

Freie Universität Berlin

- JavaScript: Einfache imperative Programmiersprache
  - Programmcode als Quelle
  - Eingebettet in HTML-Seite
  - Ausgeführt durch Interpreter im Browser
  - Zugriff auf Ereignisse und Dokumentenstruktur

- Meilensteine und Implementierungen:
  - JavaScript (1995): Netscape/Sun
  - ECMAScript: ECMA
  - JScript: Microsoft (JavaScript+Windows)
  - Seit Version 1.2 nicht mehr kompatibel in Browsern implementiert.
  - Seit 1.5: DOM Beachtung, uniforme Repräsentation des Dokumenteninhalts

- Asynchronous JavaScript and XML (AJAX) realisiert dies durch Kombination von
  - Präsentationssprachen XHTML und CSS
  - Interaktion und Modifikation im Browser mit DOM
  - Datenaustausch mit XML
  - Datentransfer durch asynchrone HTTP-Anfragen
  - Javascript als Integration dieser Technologien



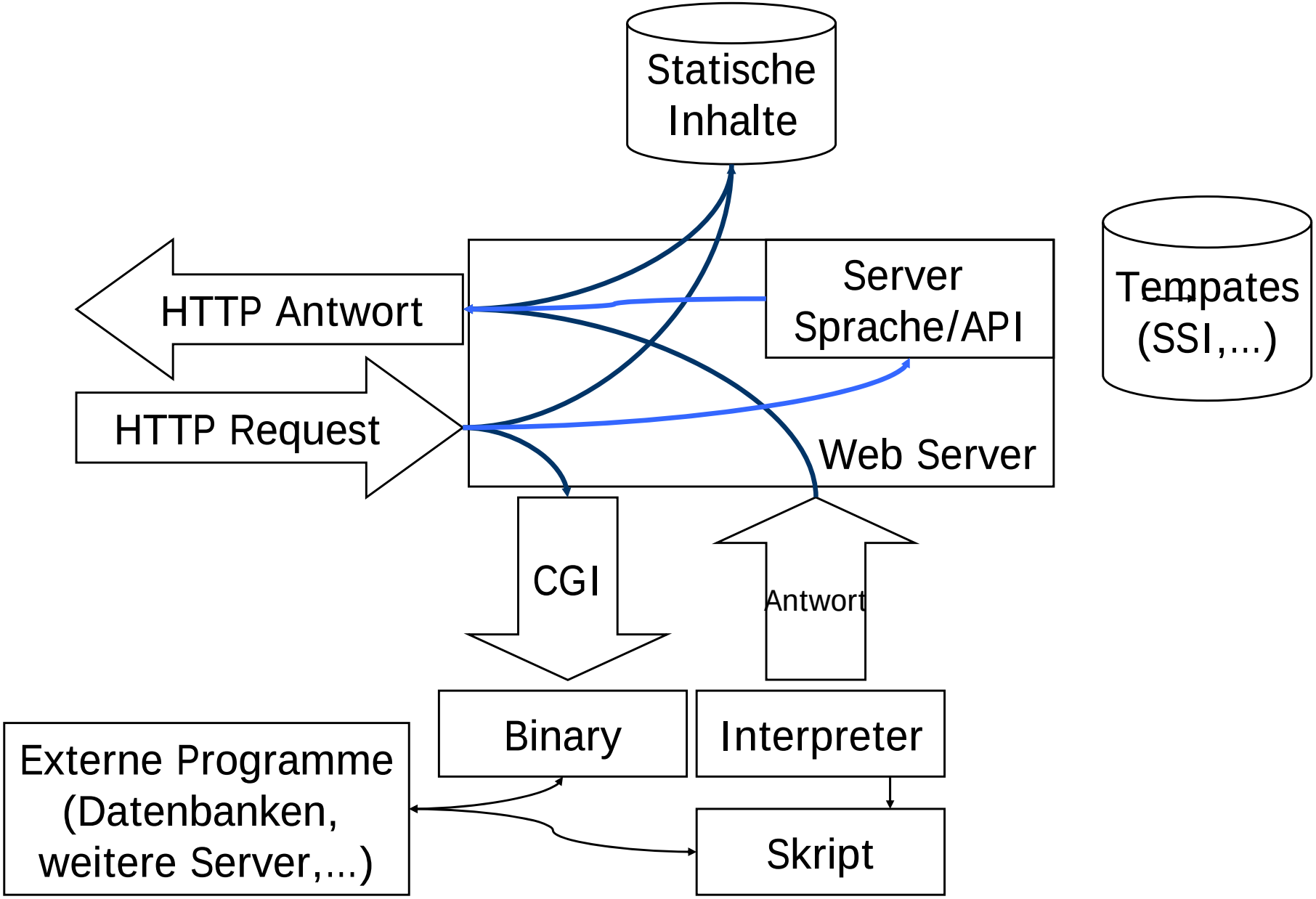
Applets

- Applets sind (kleinere) Java-Programme, die in einem Java-fähigen Web-Browser gestartet werden können.
- Das Einbinden von Applets in eine HTML-Seite erfolgt mit dem <applet>-Tag.
- Alle Applets sind Unterklassen von java.applet.Applet
- Die Klasse Applet hat folgende Oberklassen
  - java.lang.Object
  - java.awt.Component
  - java.awt.Container
  - java.awt.Panel

RMI Zugriff aus Applets heraus

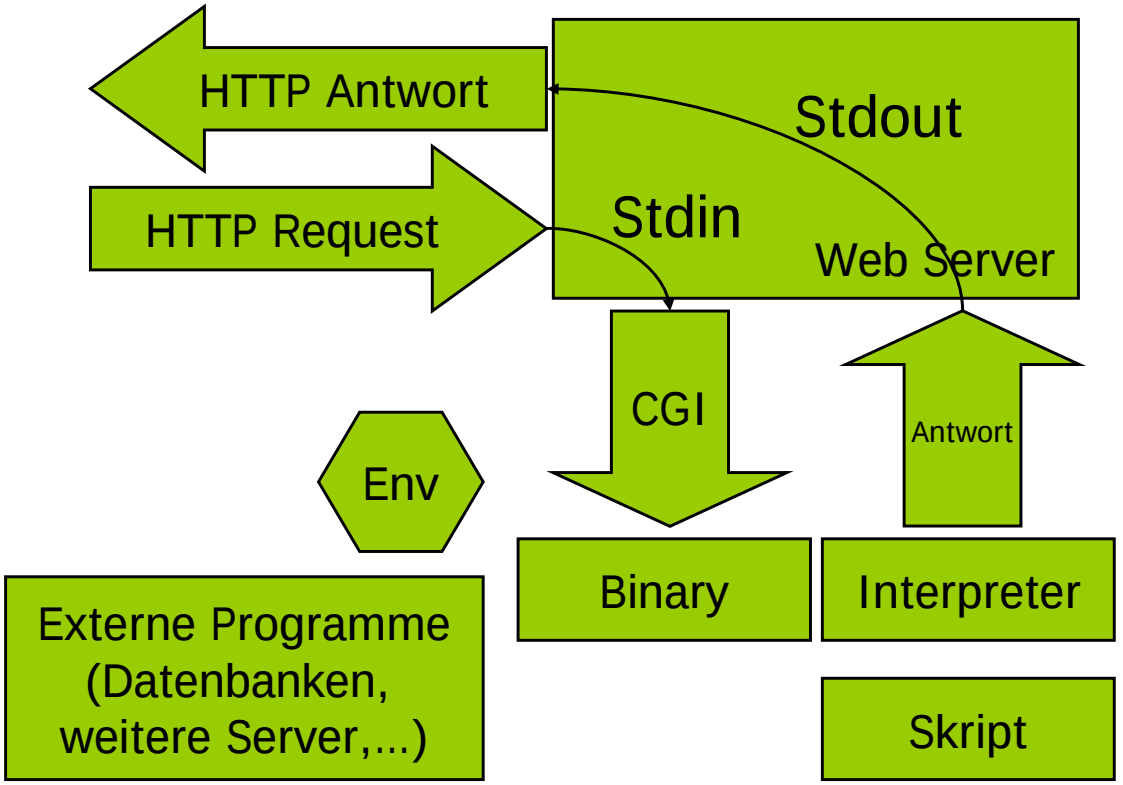
- Applets können auch auf RMI Objekte zugreifen
- Interaktion wie beim normalen RMI
- Zugriff in der Regel aber auf den Rechner beschränkt, von dem die HTML Seite geladen wurde

Im Überblick



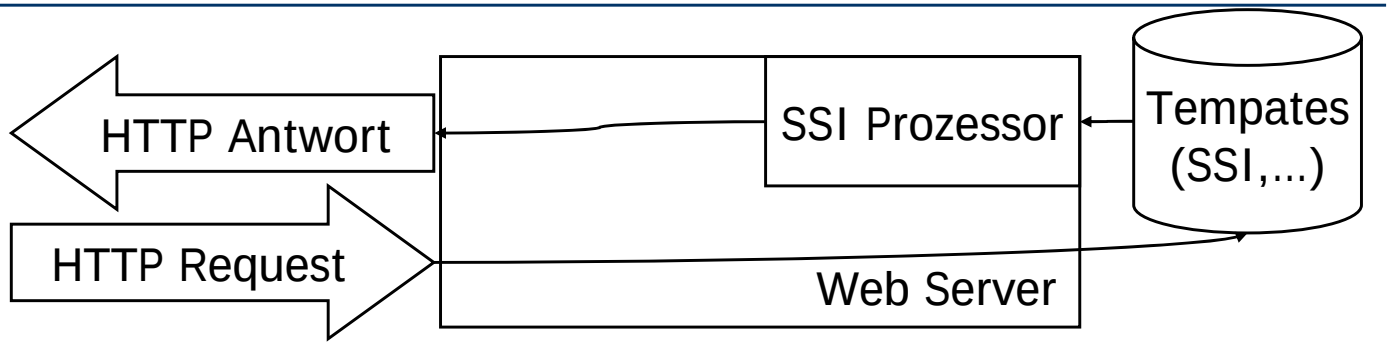
Ablauf

1. Web Server erkennt, dass URL ein Skript bezeichnet
2. Prozess wird gestartet
3. Prozess werden Eingaben mitgeteilt
   1. Initialisierte Umgebungsvariablen
   2. Standardeingabe



4. Standardausgabe des Prozesses wird zum Web-Server gelenkt
5. Skript wird ausgeführt
6. Ausgaben werden an Klienten weitergeleitet

Server Side Includes



- Server Side Includes (SSI) sind Anweisungen an den Web Server, die im HTML Code eingebunden sind
- Werden vom Server bei Auslieferung ausgeführt und zu HTML Code expandiert
- SSI Anweisung im statischen HTML-Code:

```
<div align="right">
 Letzte &Auml;nderung: <!--#flastmod file=""-->.
</div>
```

- Ergebnis nach Auslieferung

```
<div align="right">
 Letzte &Auml;nderung: Wednesday, 08-Jan-2003 09:30:15 CET.
</div>
```

PHP

- SSI sind nicht wirklich eingebettete Programme
- PHP: HTML um „Skriptsprache“ erweitern die serverseitig ausgeführt wird und dessen Ergebnis in den HTML-Text eingebettet ist
- Einbettung im HTML-Code:

```
<html>
<head><title>Das Einführungsbeispiel schlechthin
  ...</title></head>
<body>
<?php echo "Brave New PHP World"; ?>
</body>
</html>
```

- Auch <script language="php">...</script> möglich

Servlets

- Servlets:
  - CGI Programme könnten auch in Java geschrieben werden
  - Als „Interpreter“ müsste eine JVM gestartet werden
  - Wenn der Webserver selber auf einer JVM läuft, könnte er eine „CGI-Komponente“ nachladen und ausführen
  - Java Servlets sind solche Komponenten
- Unterschiede zu CGI und Applets
  - Parameterkommunikation läuft nicht über Umgebung und stdin/stdout sondern über Java-Schnittstelle
  - Applets als Komponente in JVM eines Browsers geladen
  - Servlets als Komponente in JVM eines Servers geladen
- Schnittstelle:
  - Erweitern von HttpServlet
  - Überschreiben von doGet und doPost

Java Server Pages

- Java Server Pages
  - In SSI gibt es rudimentäre Ausdrücke
  - Man könnte auch komplexere Programmfragmente mit HTML-Code mischen, wie bei PHP
  - Java Server Pages erlauben die Mischung von HTML-Code mit Java Fragmenten
  - Aus ihnen werden automatisch Servlets generiert und ausgeführt
    - HTML-Code nach Ausgabe schreiben
    - Java-Fragmente in Servlet Rahmen einbetten
- JSP bestehen aus
  - Scriptlets
  - JSP Ausdrücken
  - Deklarationen
  - JSP Anweisungen
  - HTML

III: Weitere Modelle

- Koordinationssprachen, entkoppelter Kommunikation und Koordination
- Agenten, autonome netzbasierte Entitäten

Koordinationssprachen

- Haupteigenschaften
  - *Kommunikationspartner sind anonym zueinander*
  - *Lebensdauer der Kommunikationspartner muss nicht überlappen / Kommunikation ist asynchron*
  - Entitäten kommunizieren nur indirekt über einen gemeinsamen Datenraum (Tuplespace)
  - Mehrparteienkommunikation möglich
  - Inhärent nebenläufig
  - Abstrahiert völlig von Orten der Teilnehmer
    -> verteiltes Modell für Netzprogrammierung
- Hauptproblem: Skalierbarkeit

Operationen

- out(tuple): Ablegen eines Tupels in den Tuplespace
- in(template): Herausnehmen eines passenden Tupels aus dem Tuplespace
  - Blockiert bis passendes Tupel vorliegt
    - Gleiche Anzahl Felder
    - Gleiche Typen der Felder
    - Gleicher Wert falls vorhanden
  - Zu <1,2,“start“> passen
    - <?int, ?int, ?string>
    - <1, ?int, ?string>
    - <1,2,“start“>

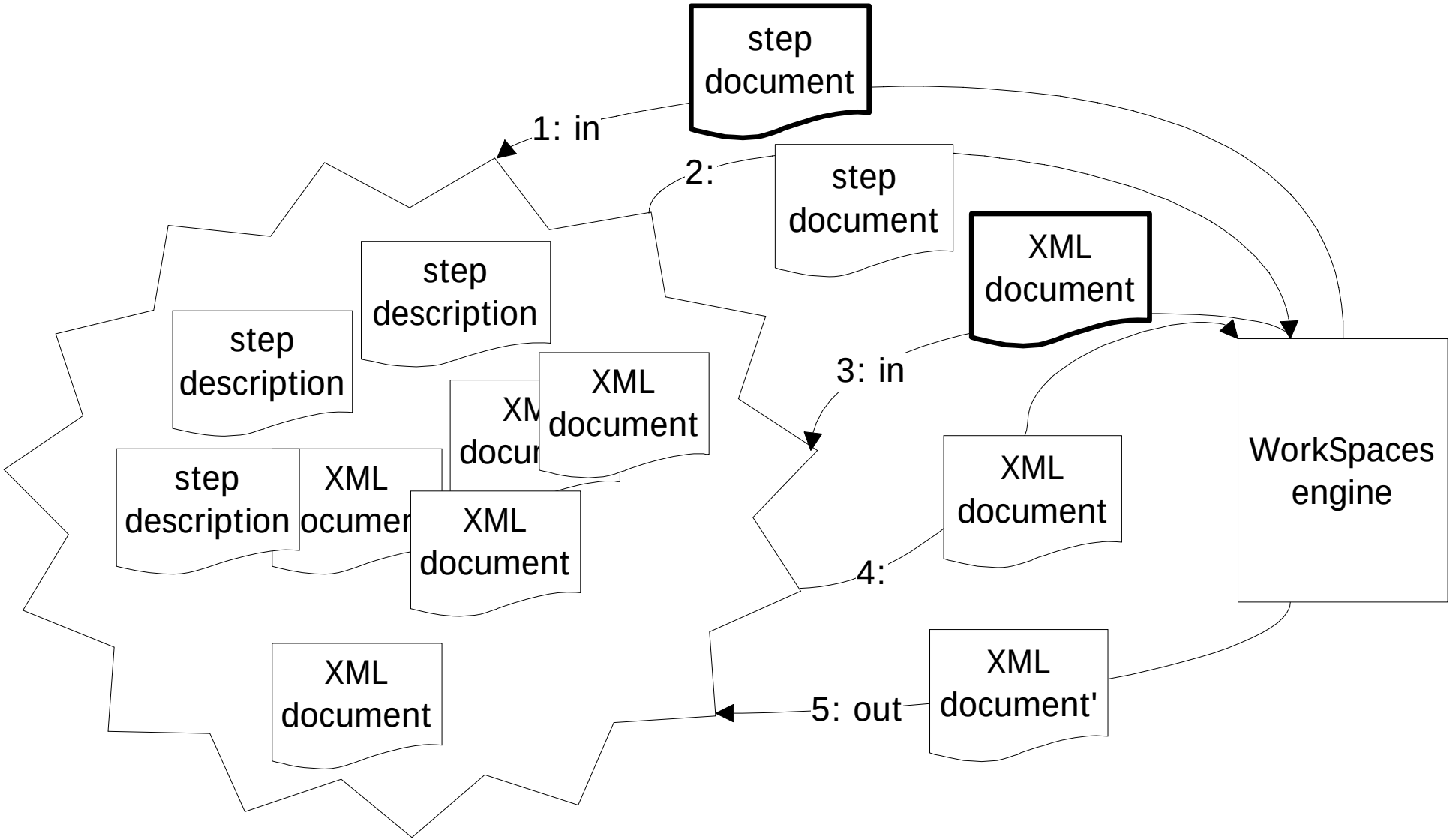
    aber nicht
    - <?int, ?int>
    - <?int, ?string, ?int>
    - <10, ?int, ?string>
    - <1,2,3>

Operationen

- rd(template):
  Auslesen eines passenden Tupels aus dem Tuplespace
  - Nur Kopie, Tupel bleibt erhalten
  - Konkurrenz mit anderen passenden out

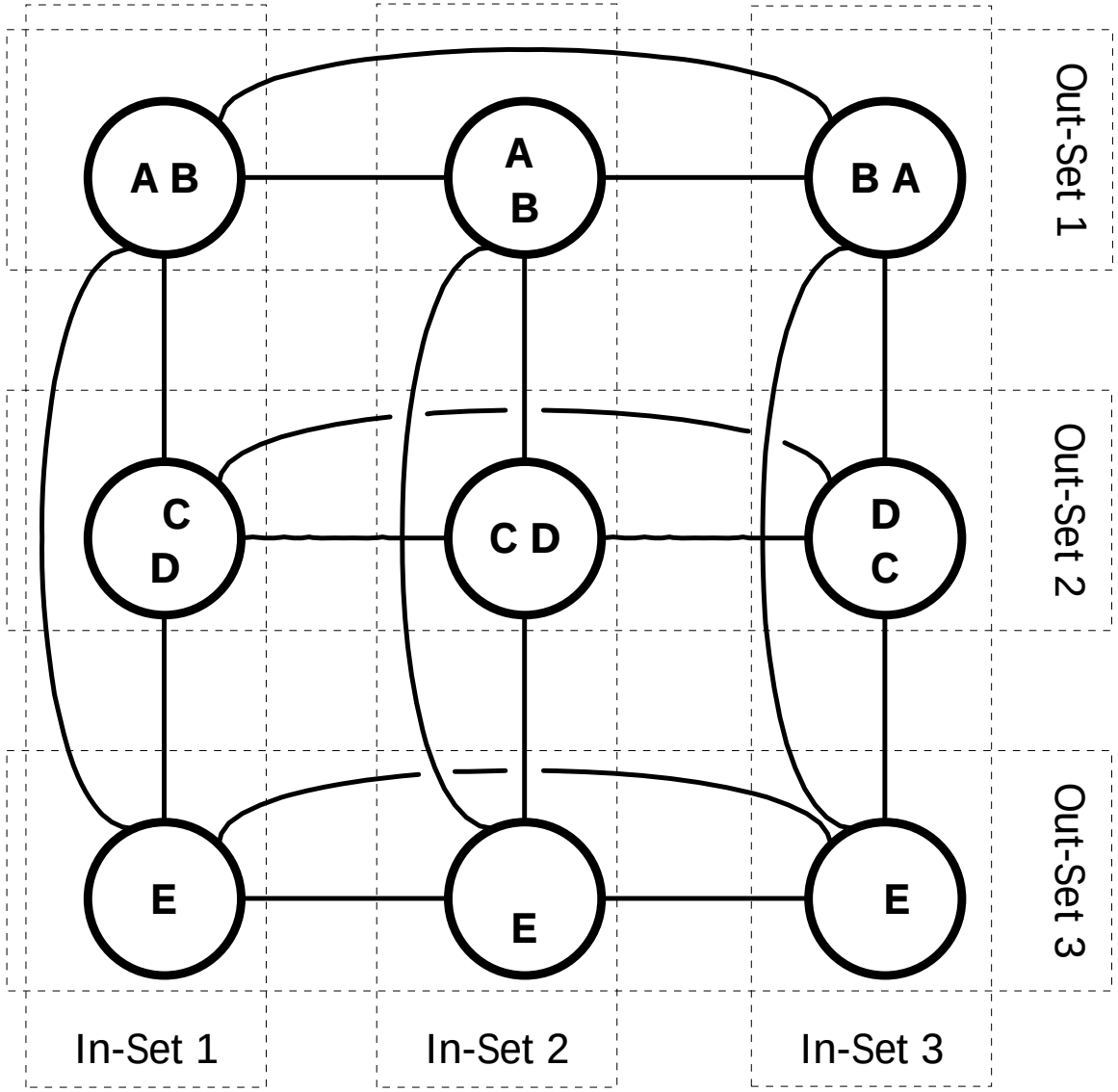
- eval(f()): Evaluation von f() parallel zum laufeden Prozess, Ergebnis als Tuple in Tuplespace
  - P:eval(f());q... ähnlich (out(f()))||q...

XML access in Workspaces



Partial replication scheme

- Nodes in out-set contain replica
- Union of contents in nodes in in-set is whole content of space



Agenten

- Agent ist
  - ein Rechnersystem, das *eigenständig* für einen Benutzer agiert
- Mehragentensystem ist
  - ein Rechnersystem, das aus mehrere Agenten besteht, die miteinander *interagieren*
- Agenten müssen
  - kooperieren
  - sich koordinieren
  - verhandeln

Agenten

- Schwache Definition
  - Autonomie: Agent operiert ohne direkte Steuerung von außen und kontrolliert seine Aktionen selber
  - Responsiveness/Empfindlichkeit: Agent beobachtet Umgebung und reagiert auf Änderungen darin
  - Proaktiveness/Initiativ: Agent reagiert nicht nur, sondern wird selbständig zur Erreichung von Zielen aktiv
  - Sozial: Agent interagiert mit anderen zur Erreichung eigener und deren Ziele
- In der Regel mit Fokus auf Software-Agenten

Agenten

- Starke Definition
  - Mobilität: Agenten bewegen sich in ienem elektronischen Netzwerk (siehe auch: Roboter)
  - Veracity/Aufrichtigkeit: Agenten geben nicht wissentlich falsche Informationen weiter
  - Rationalität: Agenten beschädigen nicht durch Aktionen ihre eigenen Ziele
  - Kooperativität: Agenten arbeiten mit (menschlichen) Auftraggebern zusammen und übernehmen deren Ziele
- Zieht Aspekte menschlichen Verhaltens ein
- ... Intelligent Agents, Smart Agents ...

Freie Universität Berlin

- Viel Glück in der Klausur

- Die in der neuen BSc-Ordnung vorgesehen Freiversuchsregelung gilt in dieser Vorlesung bis zur Veröffentlichung der neuen Ordnung *nicht*!