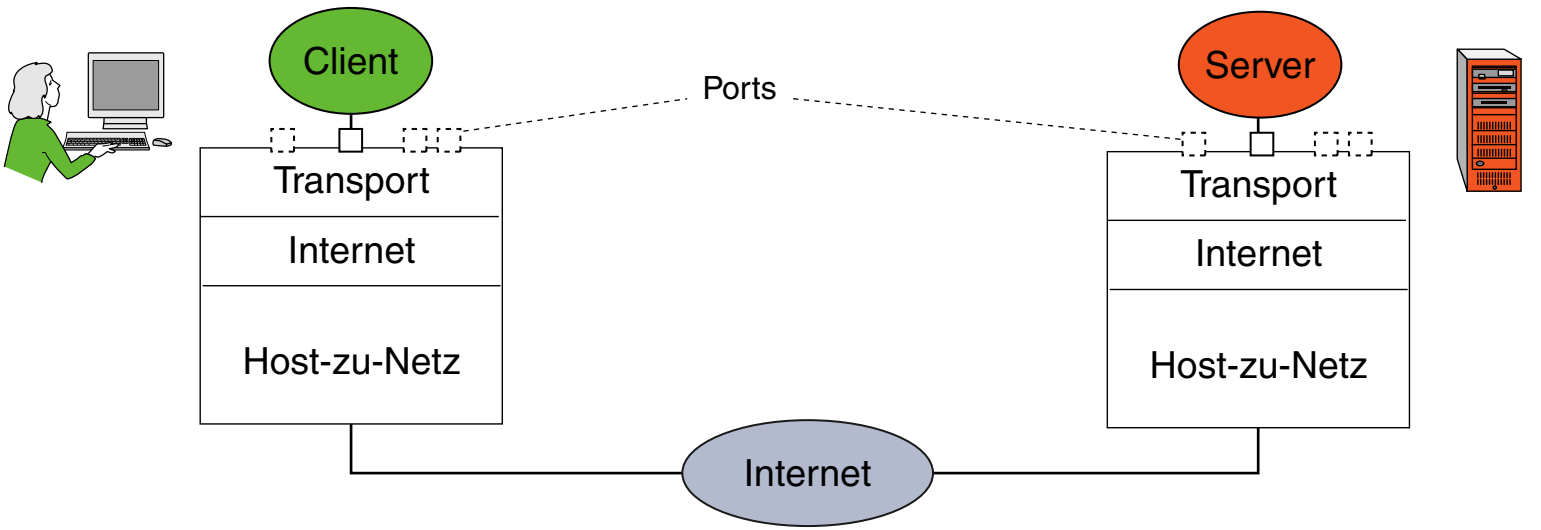
# Kapitel WT:II (Fortsetzung)

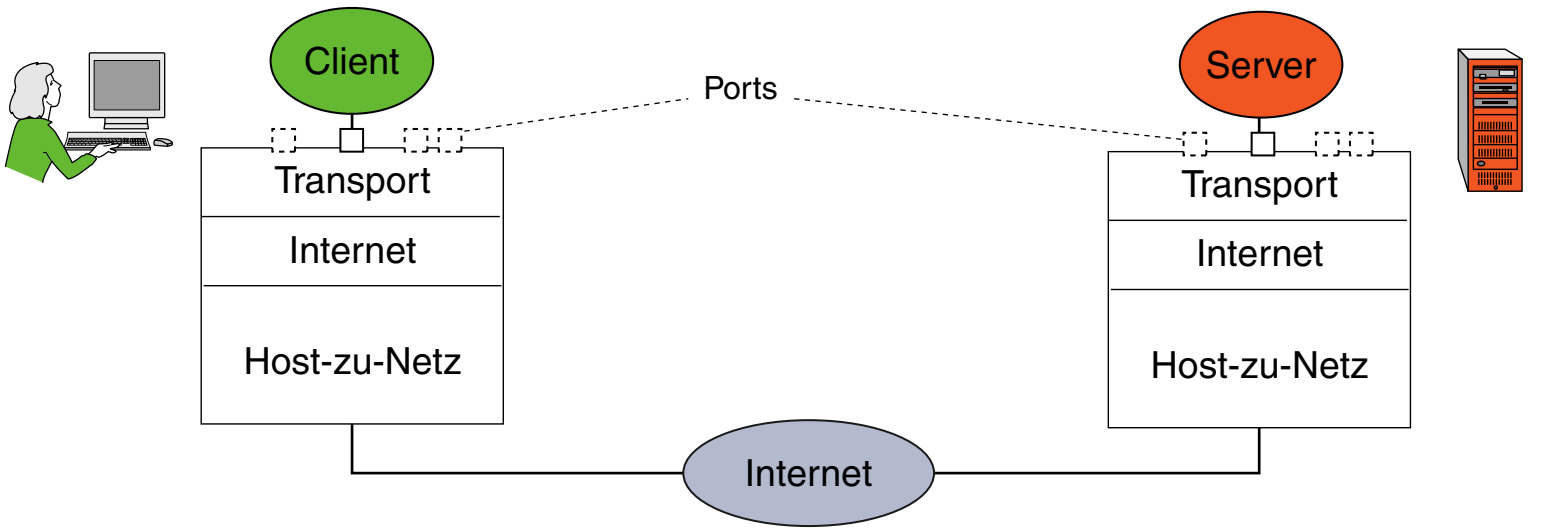
## II. Rechnerkommunikation und Protokolle

- Rechnernetze
- Prinzipien des Datenaustauschs
- Netzsoftware und Kommunikationsprotokolle
- Internetworking
- Client-Server-Interaktionsmodell
- Uniform Resource Locator
- Hypertext-Transfer-Protokoll HTTP
- Fortgeschrittene HTTP-Konzepte

# Client-Server-Interaktionsmodell



# Client-Server-Interaktionsmodell




Rollenverteilung in einem Web-basierten Informationssystem:

- Dienstgeber (*Server*), die einen bestimmten Dienst (*Service*) erbringen.
  Ein Dienst besteht aus einer oder mehreren Funktionen (Operationen, Methoden), die aufgerufen werden können.
  Ein Server ist ein Prozess oder eine Prozessgruppe auf einem Rechner.
- Dienstnehmer (*Clients*), die Serverdienste von anderen Prozessen in Anspruch nehmen.

Bemerkungen:

- Das Client-Server-Interaktionsmodell wird auch als Client-Server-Paradigma bezeichnet.
- Ein Prozess ist häufig sowohl Server (d.h., er bietet einen Dienst an) als auch Client (d.h., er benutzt andere Dienste). Bezeichnung in diesem Zusammenhang auch „Servant“.
- Das Gegenstück zum Client-Server-Paradigma ist das Peer-to-Peer-Paradigma: die Kommunikation unter Gleichgestellten.

# Client-Server-Interaktionsmodell

## Portkonzept

- Dienste (auf Anwendungsebene) werden über eine Endpunkt-zu-Endpunkt-Verbindung auf Basis der Transportschicht abgewickelt.
- Ein Port ist ein Dienstzugriffspunkt (*Service Access Point*) der Transportschicht des TCP/IP-Protokolls. Ports sind als 16 Bit-Zahl codiert.
- Zusammen definieren die IP-Adresse (Internetschicht, Schicht 3) und die Portnummer (Transportschicht, Schicht 4) einen Kommunikationskanal.

Endpunkt-zu-Endpunkt: $\underbrace{\text{IP-Adresse + Portnr.}}_{\text{Source}} \longleftrightarrow \underbrace{\text{IP-Adresse + Portnr.}}_{\text{Destination}}$

# Client-Server-Interaktionsmodell

## Portkonzept

- Dienste (auf Anwendungsebene) werden über eine Endpunkt-zu-Endpunkt-Verbindung auf Basis der Transportschicht abgewickelt.
- Ein Port ist ein Dienstzugriffspunkt (*Service Access Point*) der Transportschicht des TCP/IP-Protokolls. Ports sind als 16 Bit-Zahl codiert.
- Zusammen definieren die IP-Adresse (Internetschicht, Schicht 3) und die Portnummer (Transportschicht, Schicht 4) einen Kommunikationskanal.

Endpunkt-zu-Endpunkt: $\underbrace{\text{IP-Adresse + Portnr.}}_{\text{Source}} \longleftrightarrow \underbrace{\text{IP-Adresse + Portnr.}}_{\text{Destination}}$

Unterscheidung nach Verbindungsart bzw. Zuverlässigkeit:

- TCP-Port.
  Einrichtung von verbindungsorientiertem, zuverlässigem Transportdienst.
- UDP-Port.
  Einrichtung von verbindungslosem, unzuverlässigem Transportdienst.

# Client-Server-Interaktionsmodell

## Klassifikation von Ports

1. Standardisierte (“well-known”) Ports.
   Weltweit eindeutig für Standarddienste reserviert: 0 - 255 für TCP/IP-Anwendungen, 256 – 1.023 für besondere Unix-Anwendungen.
2. Registrierte (“registered”) Ports.
   1.024 – 49.151, werden von der IANA verwaltet.
3. Private, dynamische (“ephemeral”) Ports.
   Restliche Nummern von 49.152 – 65.535.

# Client-Server-Interaktionsmodell

## Klassifikation von Ports

1. Standardisierte (“well-known”) Ports.
   Weltweit eindeutig für Standarddienste reserviert: 0 - 255 für TCP/IP-Anwendungen, 256 – 1.023 für besondere Unix-Anwendungen.
2. Registrierte (“registered”) Ports.
   1.024 – 49.151, werden von der IANA verwaltet.
3. Private, dynamische (“ephemeral”) Ports.
   Restliche Nummern von 49.152 – 65.535.

Portnummern nach RFC 1700 [related] :

| Port | Beschreibung |
|---|---|
| 20 | File Transfer (Daten) |
| 21 | File Transfer (Steuerung) |
| 23 | Telnet, Network Virtual Terminal |
| 25 | Simple Mail Transfer (Mail-Weiterleitung) |
| 53 | Domain Name Service |
| 80 | World Wide Web, Hyper Text Transfer |

Bemerkungen:

- Die Ports von 0 bis 1023 (“well-known Ports”) sind in der RFC 1700 spezifiziert (Seite 16ff.).
- Auf Unix-Systemen sind in der Datei `/etc/services` die Ports mit ihren Service-Zuordnungen beschrieben.

# Client-Server-Interaktionsmodell

## Socket-API

- Die (Programmier-)Schnittstelle einer Anwendung zur Transportschicht wird als API (*Application Program Interface*) bezeichnet.
- Protokollstandards wie z.B. TCP/IP definieren keine API, sondern abstrakte Dienste bzw. Operationen.
- Der de-Fakto-Standard der API zur Transportschicht ist die Socket-API – kurz: Sockets. Sie stellt einen Dienst von Schicht 4 für Schicht 5 bereit.
- Ursprung: Teil von BSD-Unix, University of California at Berkeley.

# Client-Server-Interaktionsmodell

## Socket-API

- Die (Programmier-)Schnittstelle einer Anwendung zur Transportschicht wird als API (*Application Program Interface*) bezeichnet.
- Protokollstandards wie z.B. TCP/IP definieren keine API, sondern abstrakte Dienste bzw. Operationen.
- Der de-Fakto-Standard der API zur Transportschicht ist die Socket-API – kurz: Sockets. Sie stellt einen Dienst von Schicht 4 für Schicht 5 bereit.
- Ursprung: Teil von BSD-Unix, University of California at Berkeley.

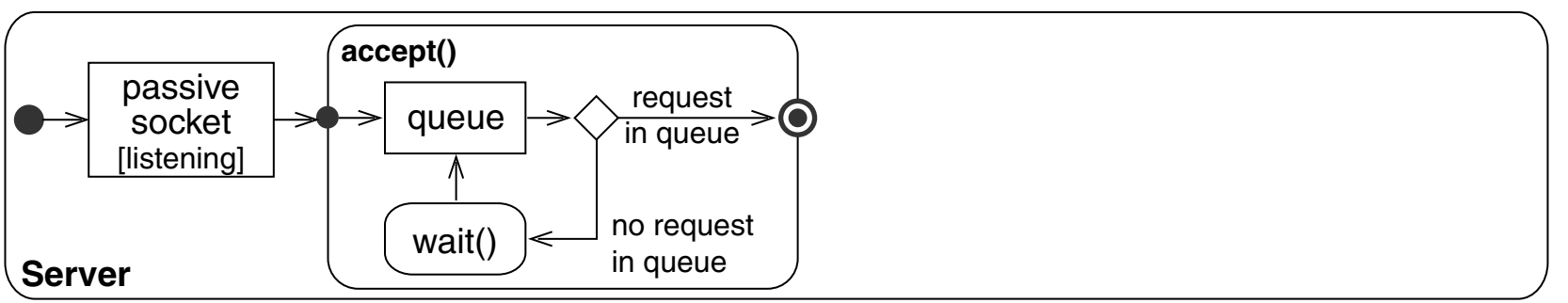
Socket-Datenstruktur:

- realisiert bei Client und Server einen Kommunikationsendpunkt
- ist vom Typ Stream, `SOCK_STREAM`, oder Datagramm, `SOCK_DGRAM`
- macht die Abwicklung des TCP/IP-Protokolls (IP-Adressen, Ports, Pufferung) transparent
- stellt Funktionen zum Schreiben, Lesen, Lauschen, etc. zur Verfügung

# Client-Server-Interaktionsmodell

## Verbindungsherstellung mit Sockets

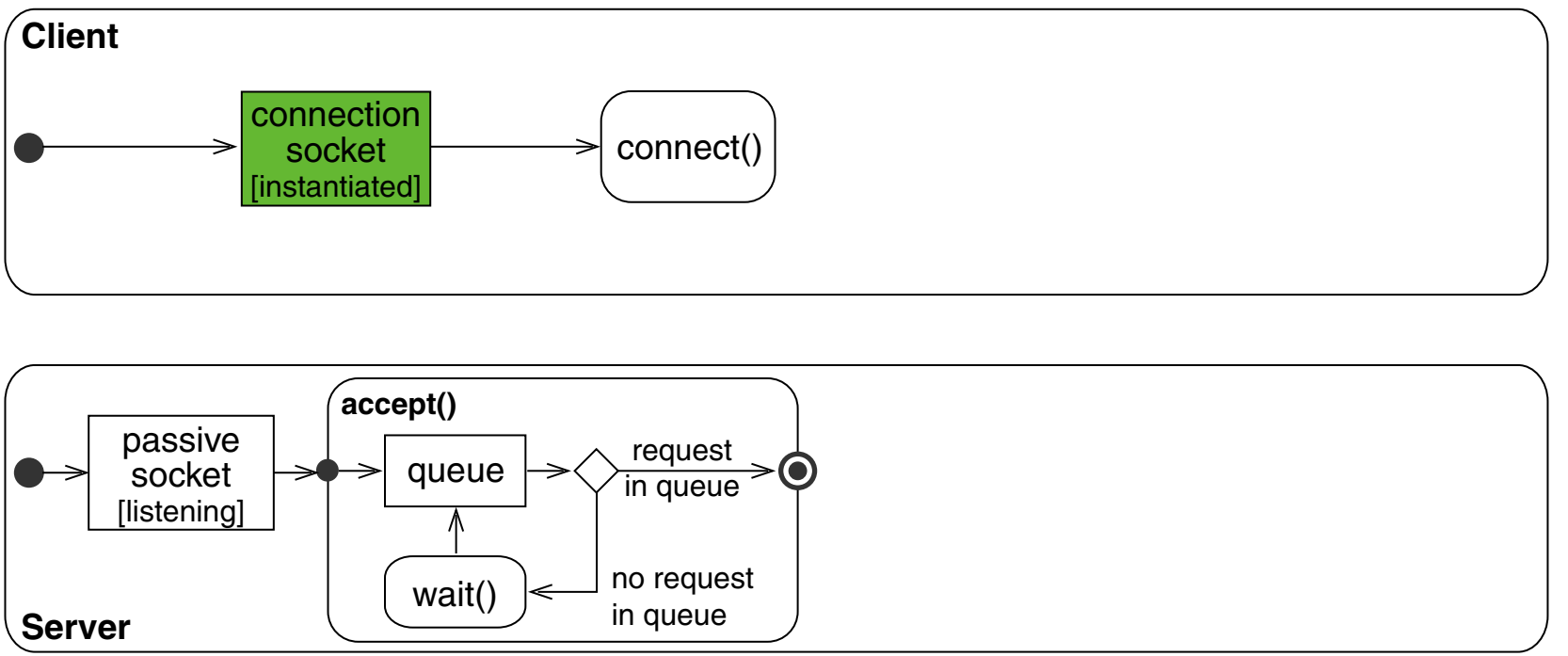
- Server fordert vom Betriebssystem eine Server-Socket-Datenstruktur an (eigene IP-Adresse, Service-Port).



# Client-Server-Interaktionsmodell

## Verbindungsherstellung mit Sockets

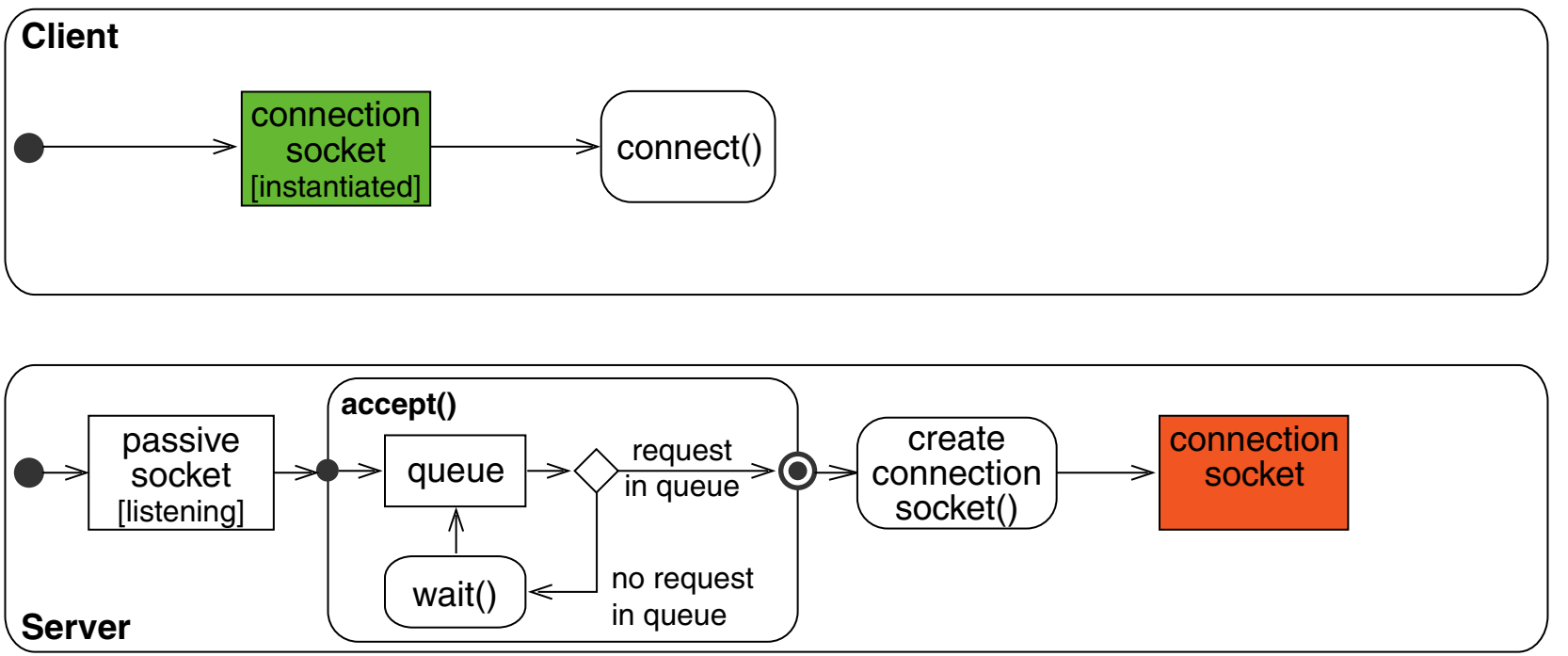
- Server fordert vom Betriebssystem eine Server-Socket-Datenstruktur an (eigene IP-Adresse, Service-Port).
- Client fordert vom Betriebssystem eine Client-Socket-Datenstruktur an (Server-IP-Adresse, Server-Service-Port); dabei wird auch ein lokaler Port beim Client reserviert.



# Client-Server-Interaktionsmodell

## Verbindungsherstellung mit Sockets

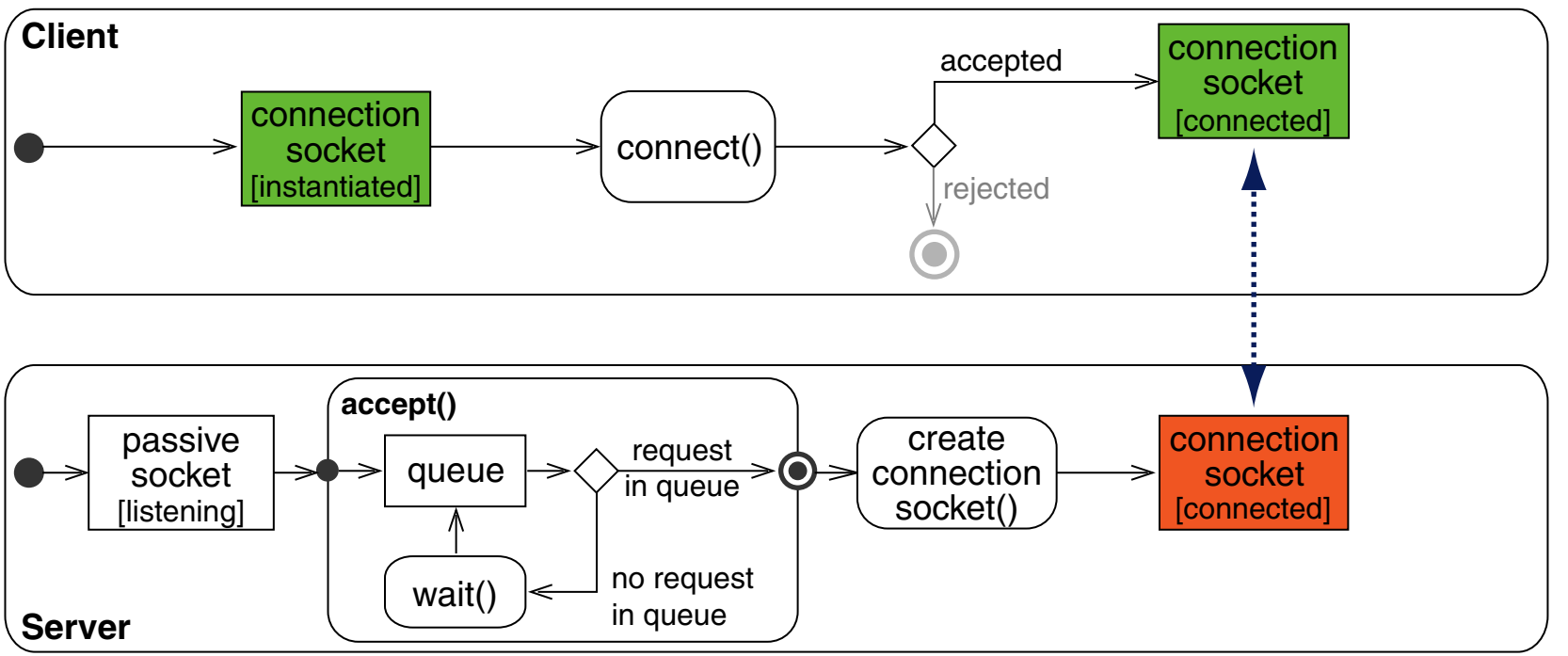
- Server fordert vom Betriebssystem eine Server-Socket-Datenstruktur an (eigene IP-Adresse, Service-Port).
- Client fordert vom Betriebssystem eine Client-Socket-Datenstruktur an (Server-IP-Adresse, Server-Service-Port); dabei wird auch ein lokaler Port beim Client reserviert.



# Client-Server-Interaktionsmodell

## Verbindungsherstellung mit Sockets

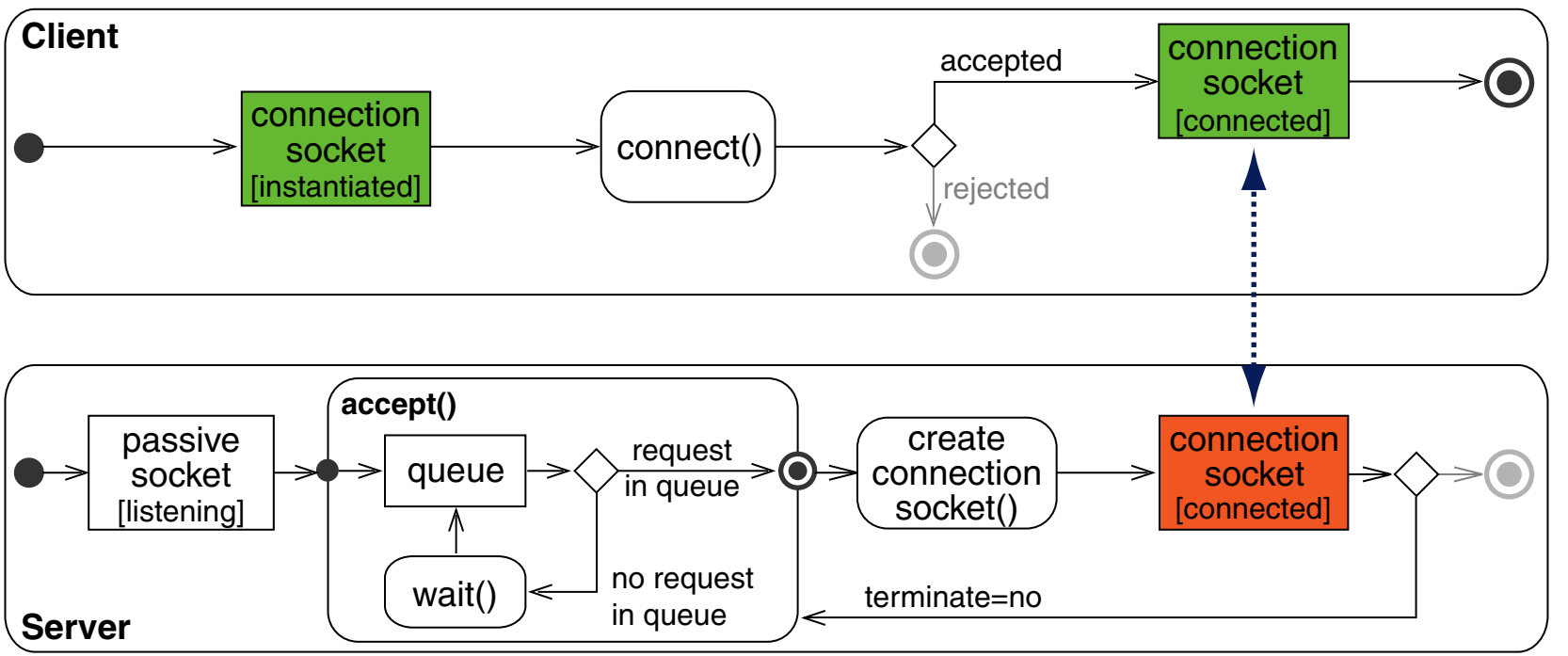
- Server fordert vom Betriebssystem eine Server-Socket-Datenstruktur an (eigene IP-Adresse, Service-Port).
- Client fordert vom Betriebssystem eine Client-Socket-Datenstruktur an (Server-IP-Adresse, Server-Service-Port); dabei wird auch ein lokaler Port beim Client reserviert.



# Client-Server-Interaktionsmodell

## Verbindungsherstellung mit Sockets

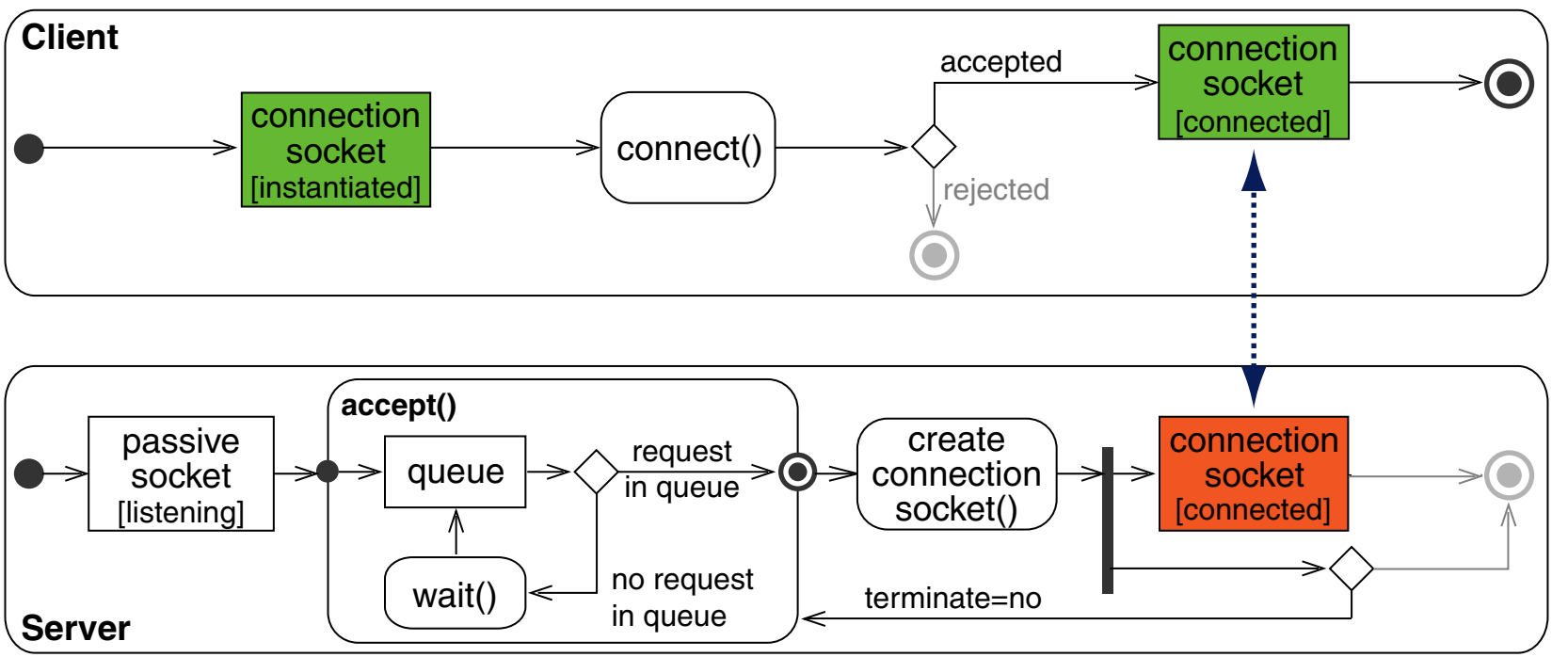
- Server fordert vom Betriebssystem eine Server-Socket-Datenstruktur an (eigene IP-Adresse, Service-Port).
- Client fordert vom Betriebssystem eine Client-Socket-Datenstruktur an (Server-IP-Adresse, Server-Service-Port); dabei wird auch ein lokaler Port beim Client reserviert.



# Client-Server-Interaktionsmodell

## Verbindungsherstellung mit Sockets

- Server fordert vom Betriebssystem eine Server-Socket-Datenstruktur an (eigene IP-Adresse, Service-Port).
- Client fordert vom Betriebssystem eine Client-Socket-Datenstruktur an (Server-IP-Adresse, Server-Service-Port); dabei wird auch ein lokaler Port beim Client reserviert.

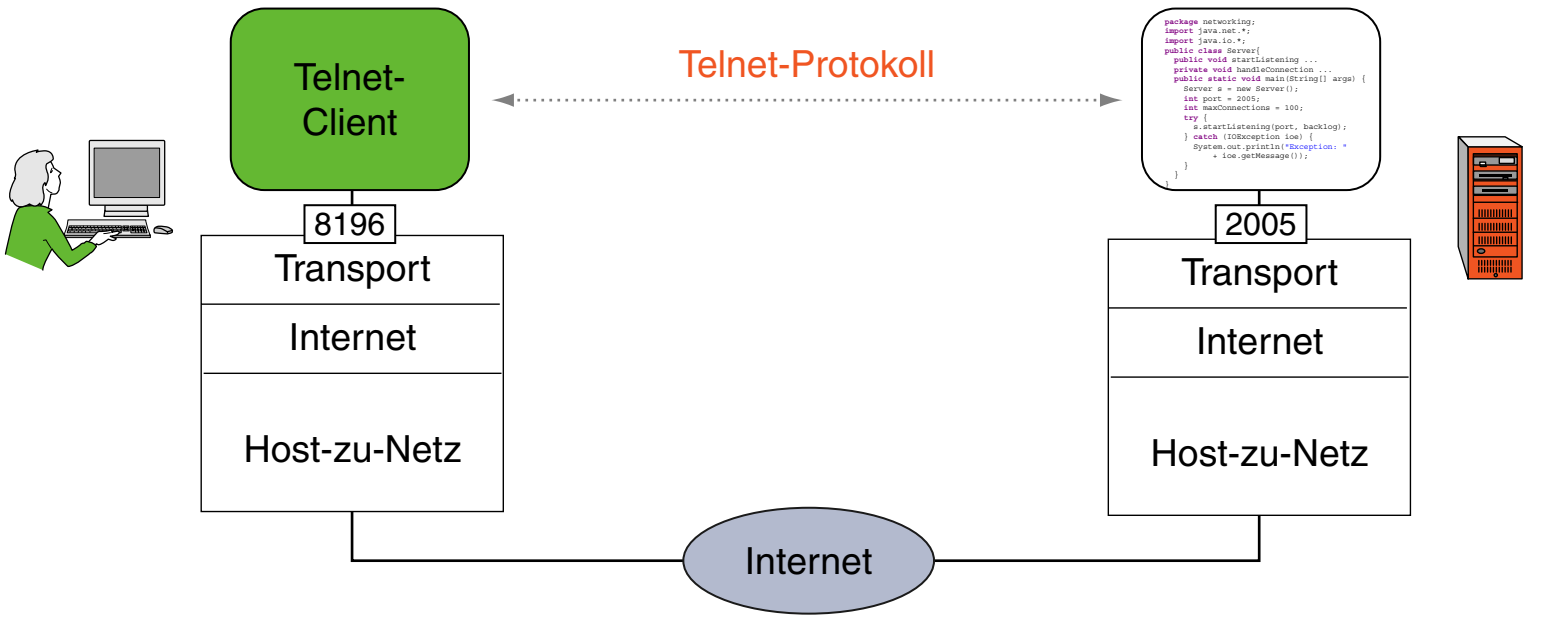


Bemerkungen:

- Schritte bei der Verbindungsherstellung:
  1. a) Server: Instanziiert passiven Socket (einmalig).
     b) Server: Wartet auf Anfrage (accept(), synchron).
  2. a) Client: Instanziiert Verbindungs-Socket.
     b) Client: Setzt Verbindungswunsch ab (connect(), synchron).
  3. Server: Der accept()-Aufruf liefert einen neuen Verbindungs-Socket bzgl. des Clients.
  4. Server, Client: Die beiden Verbindungs-Sockets bilden einen gemeinsamen Kommunikationskanal. Im Falle eines Stream-Sockets ist das ein bidirektionaler Stream.
- Der passive Socket ist in Java durch die Klasse `ServerSocket` implementiert. [Javadoc]
- Der Verbindungs-Socket ist in Java durch die Klasse `Socket` implementiert. [Javadoc]

# Client-Server-Interaktionsmodell

## Socket-Verbindung mit Java [Client-side]




Auf der Client-Seite bildet ein Telnet-Client das Gegenstück einer Verbindung zu dem Beispiel-Server. Der Telnet-Client lässt sich auf diese Art einsetzen, weil das Telnet-Protokoll lediglich Tastatureingaben vom Client zum Server bzw. Textausgaben vom Server zum Client überträgt. [RFC 854]

# Client-Server-Interaktionsmodell

## Socket-Verbindung mit Java

```
public void startListening(int port, int backlog) throws IOException {
     // Start a passive socket, step (1a).
     ServerSocket passiveSocket = new ServerSocket(port, backlog);
     // Now the passive socket is listening.
     System.out.println("Server started on port " + port + ".");
     boolean listen = true;
     while(listen) {
          // Blocking wait for a connection.
          // If a client connects, a connection socket is returned
          // by the passive socket.
          Socket connectionSocket = passiveSocket.accept(); // step (1b+3)
          handleConnection(connectionSocket); // step (4)
     }
}
```

# Client-Server-Interaktionsmodell

## Socket-Verbindung mit Java (Fortsetzung)

```
private void handleConnection(Socket connectionSocket) throws IOException {
     // Send a string to the client, step (4).
     OutputStream os = connectionSocket.getOutputStream();
     PrintWriter pw = new PrintWriter(os);
     pw.println("Tell me your name, please.");
     pw.flush(); // "Flushing" empties the send buffer.

     // Receive a string from the client, step (4).
     InputStream is = connectionSocket.getInputStream();
     BufferedReader br = new BufferedReader(new InputStreamReader(is));
     String name = br.readLine();
     if (name != null) {
          pw.println("Welcome " + name + "!");
          pw.flush();
     }

     // Close the connections.
     br.close();
     pw.close();
}
```

# Client-Server-Interaktionsmodell

## Socket-Verbindung mit Java (Fortsetzung)

```
package networkprotocol;

import java.net.*;
import java.io.*;

public class Server {

     public void startListening(int port, int backlog) ...
     private void handleConnection(Socket connectionSocket) ...

     public static void main(String[] args) {
          Server s = new Server();
          int port = 2005;
          int backlog = 100;
          try {
               s.startListening(port, backlog);
          } catch (IOException ioe) {
               System.out.println("There was an exception, the message is: "
                         + ioe.getMessage());
          }
     }
}
```

# Client-Server-Interaktionsmodell

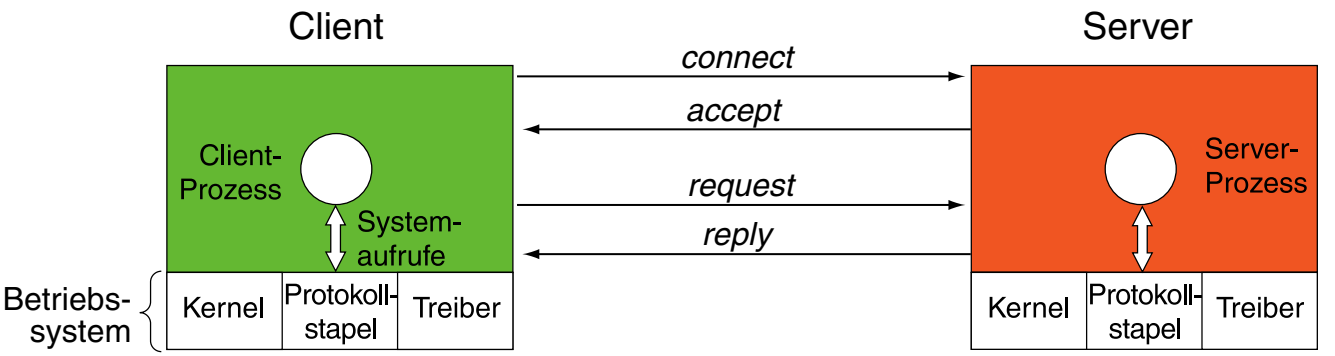
## Socket-Verbindung mit Java (Fortsetzung)

```
[stein@webis stein]$
[stein@webis stein]$ telnet localhost 2005
Trying 127.0.0.1...
Connected to localhost (127.0.0.1).
Escape character is '^]'.
Tell me your name, please.
Benno
Welcome Benno!

Connection closed by foreign host.

[stein@webis stein]$
```

# Client-Server-Interaktionsmodell

## Dienstabwicklung durch Anforderungs-/Antwortprotokoll

Zur Abwicklung eines Dienstes ist neben dem Kommunikationskanal noch ein Protokoll notwendig. Vorherrschend im Web: Anforderungs-/Antwortprotokoll.




[Tanenbaum]

- Client initiiert Kommunikation, sendet Auftrag an Server.
- Server nimmt Auftrag entgegen, bearbeitet ihn und schickt das Ergebnis an den Client zurück.
- Stichwort „synchrone Kommunikation“: Client blockiert, bis Antwort eintrifft.
- Server wartet auf nächsten Auftrag.

# Client-Server-Interaktionsmodell

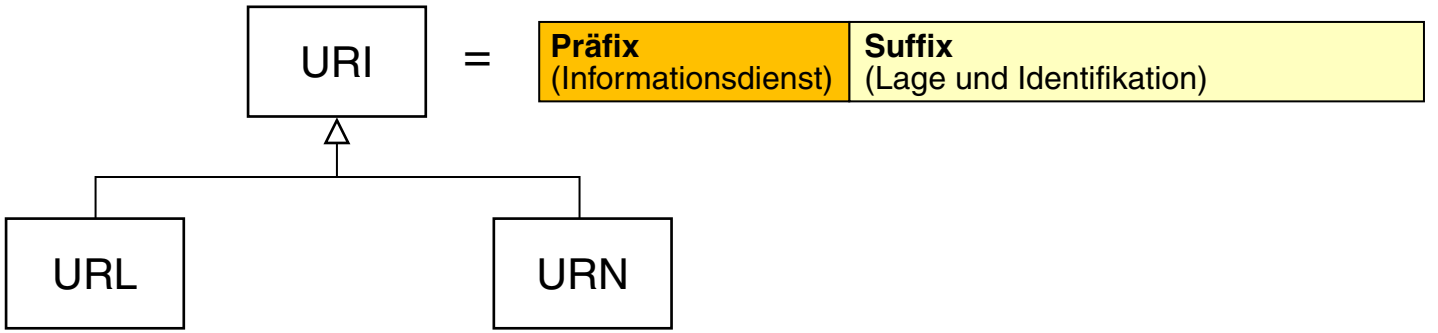
## Dienstabwicklung durch Anforderungs-/Antwortprotokoll

Ein Protokoll spezifiziert die zugelassen Anfragen (*Message Types*), die Art der Parameter-Codierung, Verhalten im Fehlerfall, Time-Outs, etc.

Bekannte Anforderungs-/Antwortprotokolle [related] :

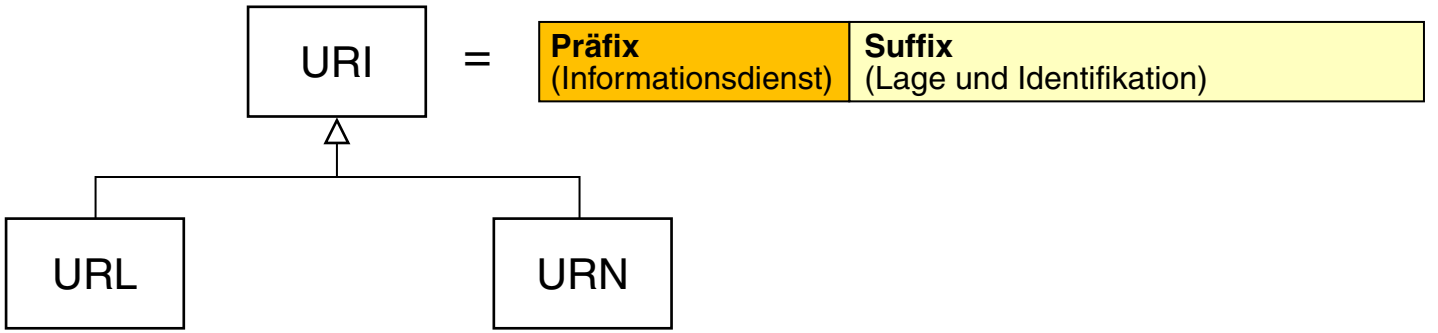
| **Dienst** | **Port** | **Beschreibung** |
|---|---|---|
| ftp-data | 20 | File Transfer (Daten) |
| ftp | 21 | File Transfer (Steuerung) |
| ssh | 22 | Secure Shell |
| smtp | 25 | Simple Mail Transfer (Mail-Vermittlung) |
| dns | 53 | Domain Name Service |
| www-http | 80 | World Wide Web, Hyper Text Transfer |
| pop3 | 110 | Post Office Protocol v3 (Mail ausliefern und abholen) |
| nntp | 119 | Network News Transfer |
| imap | 110 | Interactive Mail Access (Mail ausliefern und abholen) |

# Uniform Resource Locator




- URI [RFC 3986]
- URL [RFC 1738]
- URN [RFC 2141]

# Uniform Resource Locator




- URI [RFC 3986]
  Identifiziert eindeutig eine Informationsressource im WWW, unabhängig davon, ob es sich um ein Hypermedia-Dokument handelt.

- URL [RFC 1738]
  Identifiziert über eine eindeutige Adresse den Ort (*Location*) einer Informationsressource im WWW.

- URN [RFC 2141]
  Identifiziert über einen eindeutigen Namen eine Informationsressource im WWW. Auf Basis der URN soll (in Zukunft) der Zugriff auf die Ressource sowie die Abfrage ihrer Eigenschaften möglich sein.

# Uniform Resource Locator

## URL versus URI [RFC: 3986, 3305:2]

*The term "Uniform Resource Locator" (URL) refers to the subset of URIs that, in addition to identifying a resource, provide a means of locating the resource by describing its primary access mechanism (e.g., its network "location").*

*. . .*

*A common misunderstanding of URIs is that they are only used to refer to accessible resources. The URI itself only provides identification; access to the resource is neither guaranteed nor implied by the presence of a URI. Instead, any operation associated with a URI reference is defined by the protocol element, data format attribute, or natural language text in which it appears.*

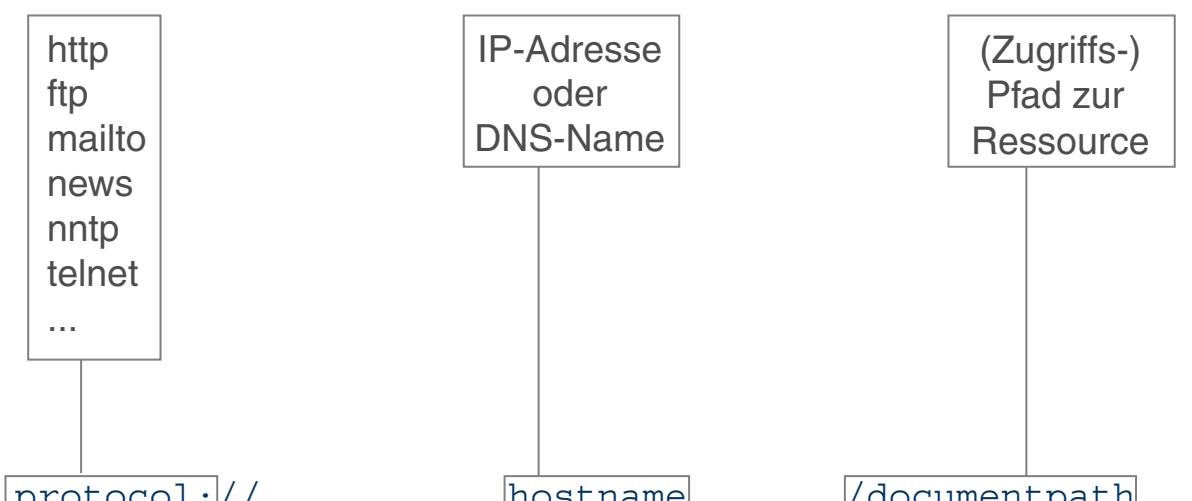
# Uniform Resource Locator

## URL versus URI [RFC: 3986, 3305:2]

*The term "Uniform Resource Locator" (URL) refers to the subset of URIs that, in addition to identifying a resource, provide a means of locating the resource by describing its primary access mechanism (e.g., its network "location").*

*. . .*

*A common misunderstanding of URIs is that they are only used to refer to accessible resources. The URI itself only provides identification; access to the resource is neither guaranteed nor implied by the presence of a URI. Instead, any operation associated with a URI reference is defined by the protocol element, data format attribute, or natural language text in which it appears.*



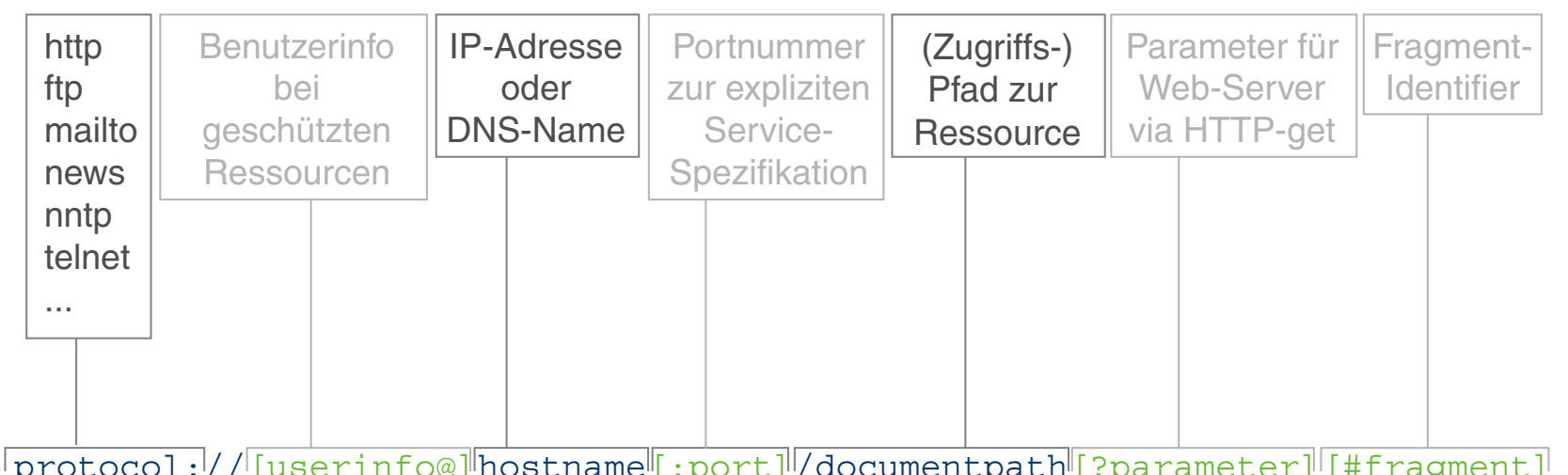
# Uniform Resource Locator

## URL versus URI [RFC: 3986, 3305:2]

*The term "Uniform Resource Locator" (URL) refers to the subset of URIs that, in addition to identifying a resource, provide a means of locating the resource by describing its primary access mechanism (e.g., its network "location").*

*. . .*

*A common misunderstanding of URIs is that they are only used to refer to accessible resources. The URI itself only provides identification; access to the resource is neither guaranteed nor implied by the presence of a URI. Instead, any operation associated with a URI reference is defined by the protocol element, data format attribute, or natural language text in which it appears.*



# Uniform Resource Locator

## BNF-Syntax der HTTP-URL

```
URI ::=             scheme ':' hier-part [ '?' query ] [ '#' fragment ]

scheme ::=          'http' | ...
hier-part ::=       '//' authority path-rooted
authority ::=       [ userinfo '@' ] host [ ':' port ]
host ::=            hostname | IPv4address | IPv6address

hostname ::=        { domainlabel '.' }* toplabel
domainlabel ::=     alphadigit { { alphadigit | '-' }* alphadigit }?
toplabel ::=        alpha { { alphadigit | '-' }* alphadigit }?

path-rooted ::=     { '/' segment }*
path ::=            { segment-r }? { '/' segment }*
segment ::=         { uchar | ':' | '@' }*
segment-r ::=       { uchar | '@' }+

query ::=           { uchar | ':' | '@' | '/' | '?' }*
fragment ::=        { uchar | ':' | '@' | '/' | '?' }*

userinfo ::=        { uchar | ':' }*
IPv4address ::=     digits . digits . digits . digits
port ::=            digits
```

# Uniform Resource Locator

## BNF-Syntax der HTTP-URL

```
URI ::=             scheme ':' hier-part [ '?' query ] [ '#' fragment ]

scheme ::=          'http' | ...
hier-part ::=       '//' authority path-rooted
authority ::=       [ userinfo '@' ] host [ ':' port ]
host ::=            hostname | IPv4address | IPv6address

hostname ::=        { domainlabel '.' }* toplabel
domainlabel ::=     alphadigit { { alphadigit | '-' }* alphadigit }?
toplabel ::=        alpha { { alphadigit | '-' }* alphadigit }?

path-rooted ::=     { '/' segment }*
path ::=            { segment-r }? { '/' segment }*
segment ::=         { uchar | ':' | '@' }*
segment-r ::=       { uchar | '@' }+

query ::=           { uchar | ':' | '@' | '/' | '?' }*
fragment ::=        { uchar | ':' | '@' | '/' | '?' }*

userinfo ::=        { uchar | ':' }*
IPv4address ::=     digits . digits . digits . digits
port ::=            digits
```

# Uniform Resource Locator

## BNF-Syntax der HTTP-URL

```
URI ::=              scheme ':' hier-part [ '?' query ] [ '#' fragment ]

scheme ::=           'http' | ...
hier-part ::=        '//' authority path-rooted
authority ::=        [ userinfo '@' ] host [ ':' port ]
host ::=             hostname | IPv4address | IPv6address

hostname ::=         { domainlabel '.' }* toplabel
domainlabel ::=      alphadigit { { alphadigit | '-' }* alphadigit }?
toplabel ::=         alpha { { alphadigit | '-' }* alphadigit }?

path-rooted ::=      { '/' segment }*
path ::=             { segment-r }? { '/' segment }*
segment ::=          { uchar | ':' | '@' }*
segment-r ::=        { uchar | '@' }+

query ::=            { uchar | ':' | '@' | '/' | '?' }*
fragment ::=         { uchar | ':' | '@' | '/' | '?' }*

userinfo ::=         { uchar | ':' }*
IPv4address ::=      digits . digits . digits . digits
port ::=             digits
```

# Uniform Resource Locator

## BNF-Syntax der HTTP-URL

```
URI ::=             scheme ':' hier-part [ '?' query ] [ '#' fragment ]

scheme ::=          'http' | ...
hier-part ::=       '//' authority path-rooted
authority ::=       [ userinfo '@' ] host [ ':' port ]
host ::=            hostname | IPv4address | IPv6address

hostname ::=        { domainlabel '.' }* toplabel
domainlabel ::=     alphadigit { { alphadigit | '-' }* alphadigit }?
toplabel ::=        alpha { { alphadigit | '-' }* alphadigit }?

path-rooted ::=     { '/' segment }*
path ::=            { segment-r }? { '/' segment }*
segment ::=         { uchar | ':' | '@' }*
segment-r ::=       { uchar | '@' }+

query ::=           { uchar | ':' | '@' | '/' | '?' }*
fragment ::=        { uchar | ':' | '@' | '/' | '?' }*

userinfo ::=        { uchar | ':' }*
IPv4address ::=     digits . digits . digits . digits
port ::=            digits
```

# Uniform Resource Locator

## BNF-Syntax der HTTP-URL

```
URI ::=             scheme ':' hier-part [ '?' query ] [ '#' fragment ]

scheme ::=          'http' | ...
hier-part ::=       '//' authority path-rooted
authority ::=       [ userinfo '@' ] host [ ':' port ]
host ::=            hostname | IPv4address | IPv6address

hostname ::=        { domainlabel '.' }* toplabel
domainlabel ::=     alphadigit { { alphadigit | '-' }* alphadigit }?
toplabel ::=        alpha { { alphadigit | '-' }* alphadigit }?

path-rooted ::=     { '/' segment }*
path ::=            { segment-r }? { '/' segment }*
segment ::=         { uchar | ':' | '@' }*
segment-r ::=       { uchar | '@' }+

query ::=           { uchar | ':' | '@' | '/' | '?' }*
fragment ::=        { uchar | ':' | '@' | '/' | '?' }*

userinfo ::=        { uchar | ':' }*
IPv4address ::=     digits . digits . digits . digits
port ::=            digits
```

Bemerkungen:

- Die Aufteilung von URIs in disjunkte Mengen von URLs und URNs gilt mittlerweile als überholt. Stattdessen betrachtet man die Menge der URIs aufgeteilt durch (1) die Protokolle für URLs (`http`, `ftp`, etc.) und (2) die Namspace-Identifier für URNs (`isbn`, `doi`, etc.). [RFC 3305:2]
- Die angegebene BNF basiert auf der BNF für URIs [RFC 3986: 3, Appendix] und für URLs [RFC 1738:5]. In der vorliegenden Vereinfachung erlaubt die BNF die Bildung typischer URLs, nicht nur im HTTP-Kontext.
- `alpha`, `alphadigit`, `alphadigit` und `uchar` sind Zeichenmengen, die Buchstaben, Ziffern und ggf. zusätzliche Sonderzeichen (ohne Sonderbedeutung) enthalten und unterschiedliche Repräsentationen erlauben.

# Uniform Resource Locator

## Namensauflösung URN → URL

- URN-Syntax [RFC 2141:2] :

  ```
  URN ::= 'urn:' namespace-id ':' namespace-specific-string
  ```

- Beispiel: `urn:ISBN:0-262-01210-3`

# Uniform Resource Locator

## Namensauflösung URN → URL

- URN-Syntax [RFC 2141:2] :

  ```
  URN ::= 'urn:' namespace-id ':' namespace-specific-string
  ```

- Beispiel: `urn:ISBN:0-262-01210-3`

# Uniform Resource Locator

## Namensauflösung URN → URL

- URN-Syntax [RFC 2141:2] :

  ```
  URN ::= 'urn:' namespace-id ':' namespace-specific-string
  ```

- Beispiel: `urn:ISBN:0-262-01210-3`

Namensauflösung URN → URL mittels Resolver Discovery Service [RFC 2276] :

# Uniform Resource Locator

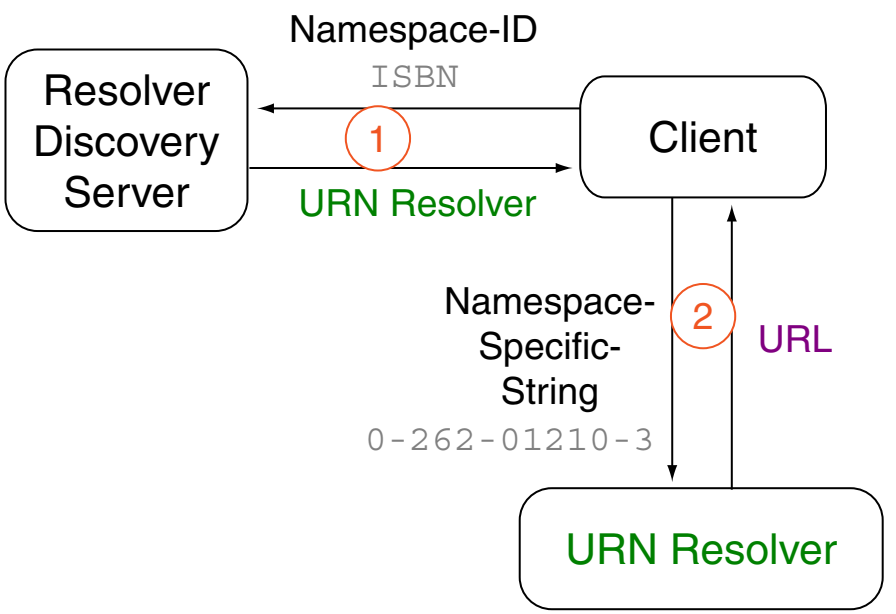
## Namensauflösung URN → URL

- URN-Syntax [RFC 2141:2] :

  ```
  URN ::= 'urn:' namespace-id ':' namespace-specific-string
  ```

- Beispiel: `urn:ISBN:0-262-01210-3`

Namensauflösung URN → URL mittels Resolver Discovery Service [RFC 2276] :



# Uniform Resource Locator

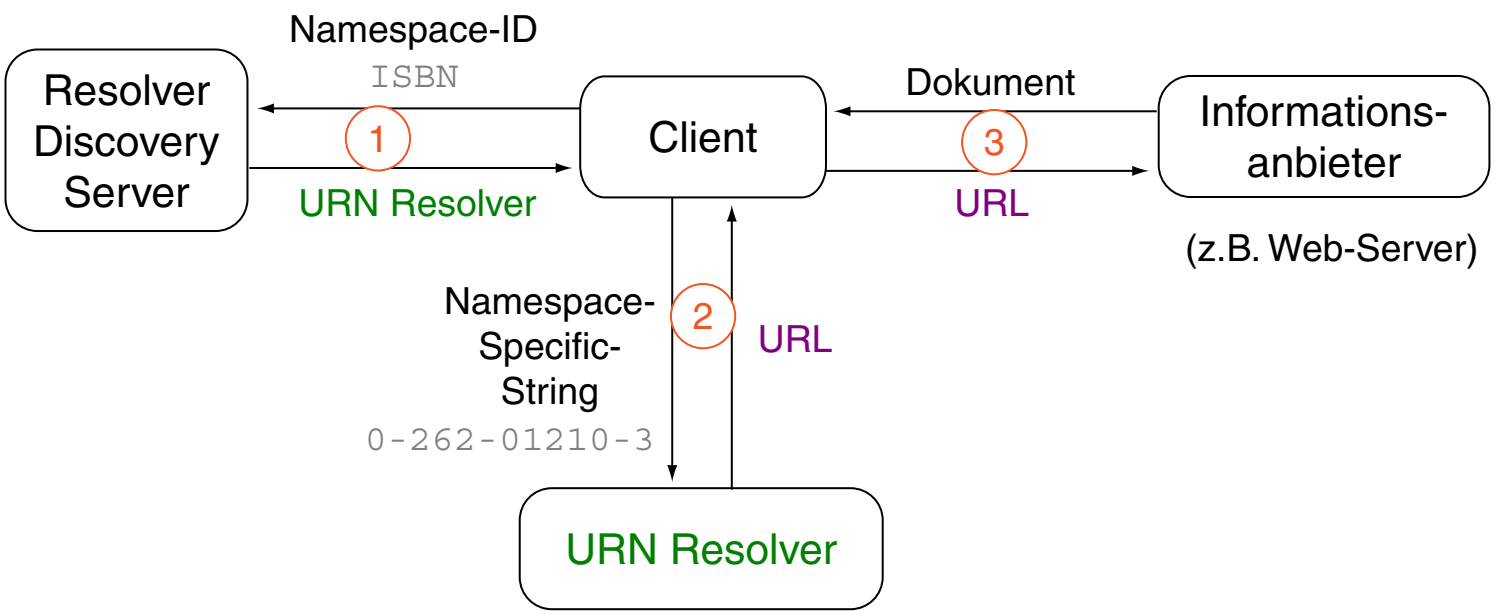
## Namensauflösung URN → URL

- URN-Syntax [RFC 2141:2] :

  ```
  URN ::= 'urn:' namespace-id ':' namespace-specific-string
  ```

- Beispiel: `urn:ISBN:0-262-01210-3`

Namensauflösung URN → URL mittels Resolver Discovery Service [RFC 2276] :



Bemerkungen:

- Ein Ort (URL) einer Ressource kann sich ändern, ihr Name (URN) nicht. Ziel ist es, Änderungen des Ortes einer Ressource automatisch nachzuvollziehen. Hierfür ist ein Dienst erforderlich, der aus dem Namen einer Ressource deren Standort ermitteln kann.

- Zurzeit werden im WWW fast ausschließlich URLs zur Identifikation von Ressourcen verwendet; hinsichtlich der Standardisierung eines Dienstes zur Abbildung URN $\rightarrow$ URL konnte sich bislang nicht geeinigt werden: Die Category von RFC 2276 (URN Resolution) ist „Informational“, die von RFC 2141 (URN Syntax) ist „Standards Track“.

- Namensauflösung URN $\rightarrow$ URL bei *Digital Object Identifiern*, DOI, [www.doi.org] :

  1. Für den Namespace-ID „DOI“ wird der URN-Resolver dx.doi.org herausgesucht.
  2. Der URN-Resolver ermittelt für den Namespace-Specific-String `10.1007/s10579-010-9115-y` die URL `https://link.springer.com/article/10.1007%2Fs10579-010-9115-y`.
  3. Die URL verweist auf das Dokument „Stein/Lipka/Prettenhofer: Intrinsic Plagiarism Analysis“, das durch den in der URL spezifizierten Web-Server ausgeliefert wird.

  Die Schritte 2 und 3 werden auf der Web-Seite webis.de > publications durch Klicken des [doi]-Links transparent ausgeführt.