FACHHOCHSCHULE
TECHNIKUM WIEN

TCP/IP Client Server Architektur

Franz Hollerer

Thomas M. Galla

Lehrziel

FACHHOCHSCHULE TECHNIKUM WIEN

- Programmieren von TCP/IP Client/Server Anwendungen basierend auf dem Socket API
- Praktische Anwendung der in den Lehrveranstaltungen Computernetze und Betriebssysteme erworbenen Kenntnisse

Ports und Sockets

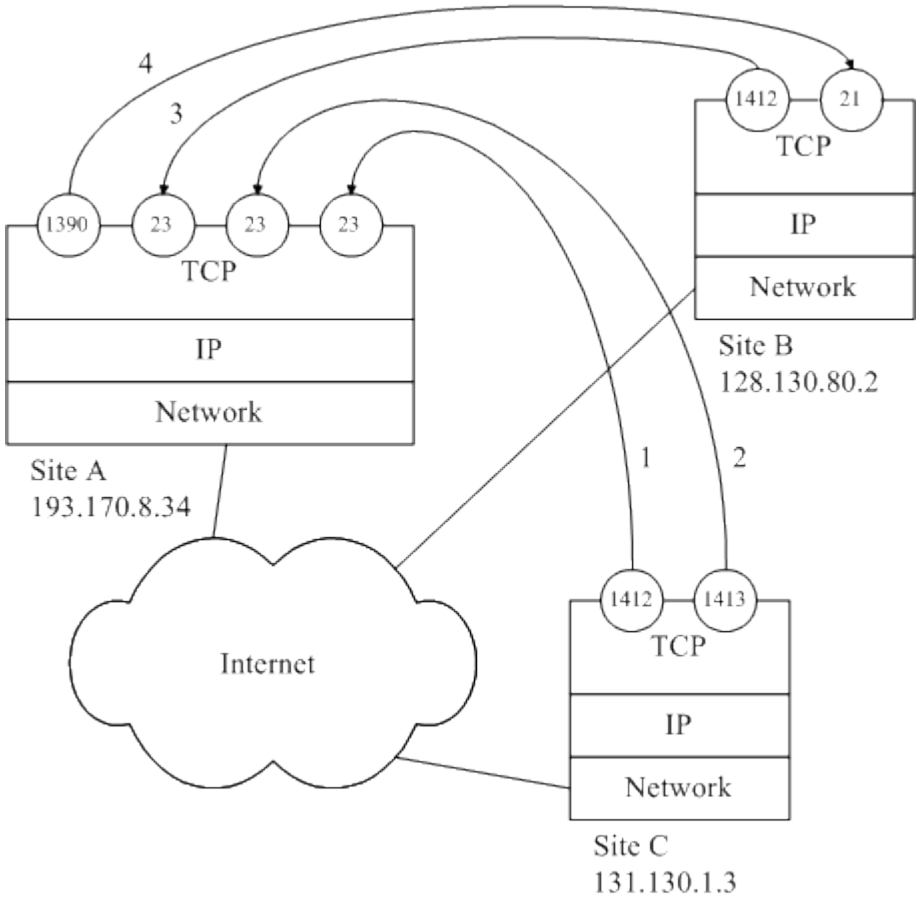
FACHHOCHSCHULE TECHNIKUM WIEN

- Wie finden Client und Server zueinander?
- Der Dienst auf einem Server wird über einen well known **port** (“Anschlussbuchse”) identifiziert.
- Der Server selbst über die IP-Adresse
- Kombination von IP-Adresse:Port wird als **Socket** bezeichnet
- Client-Server Verbindung ist durch ein **Socket-Paar** (Server-Socket, Client-Socket) eindeutig definiert

Ports und Sockets (cont.)

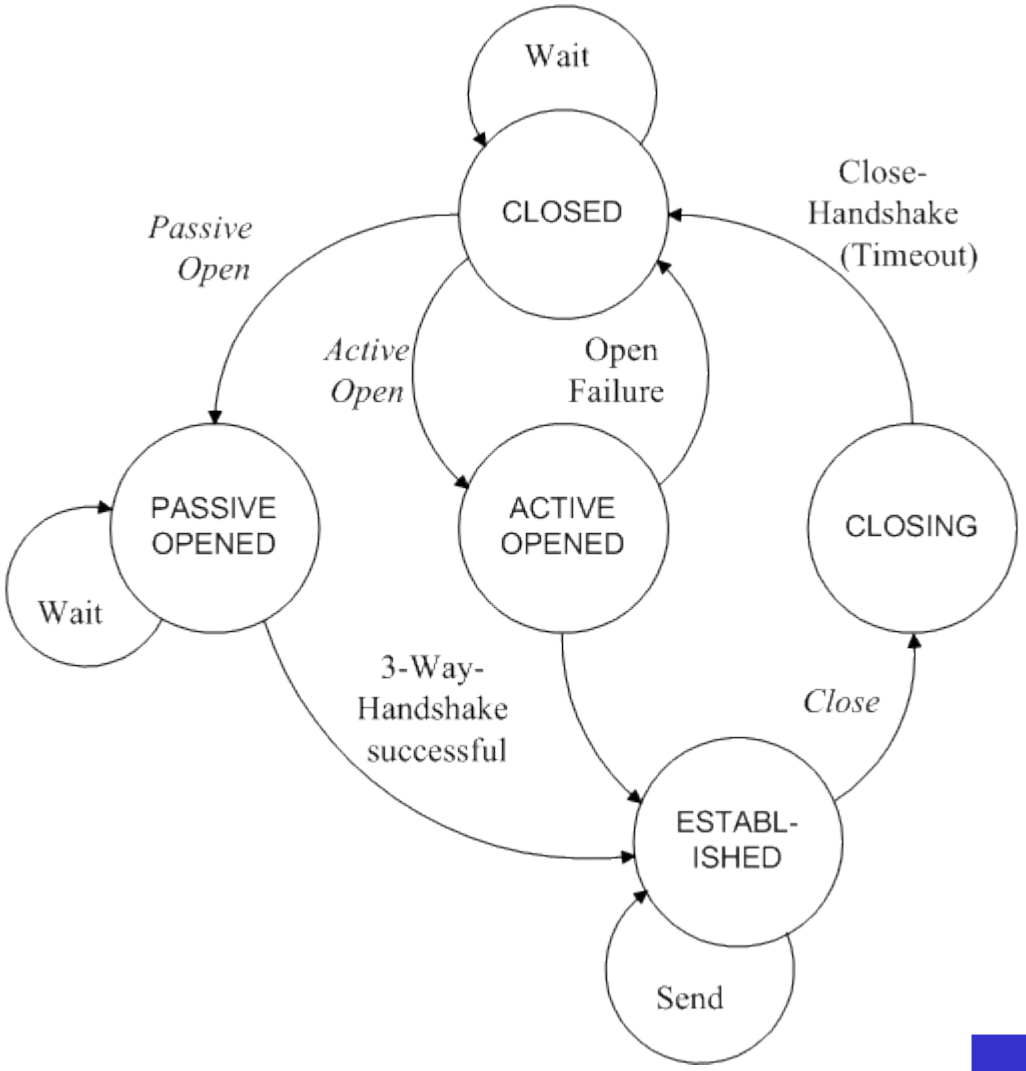
FACHHOCHSCHULE TECHNIKUM WIEN

Connection	Client-InAddr	Client-Port	Server-InAddr	Server-Port
1	131.130.1.3	1412	193.170.8.34	23
2	131.130.1.3	1413	193.170.8.34	23



TCP State Machine (grob)

FACHHOCHSCHULE TECHNIKUM WIEN



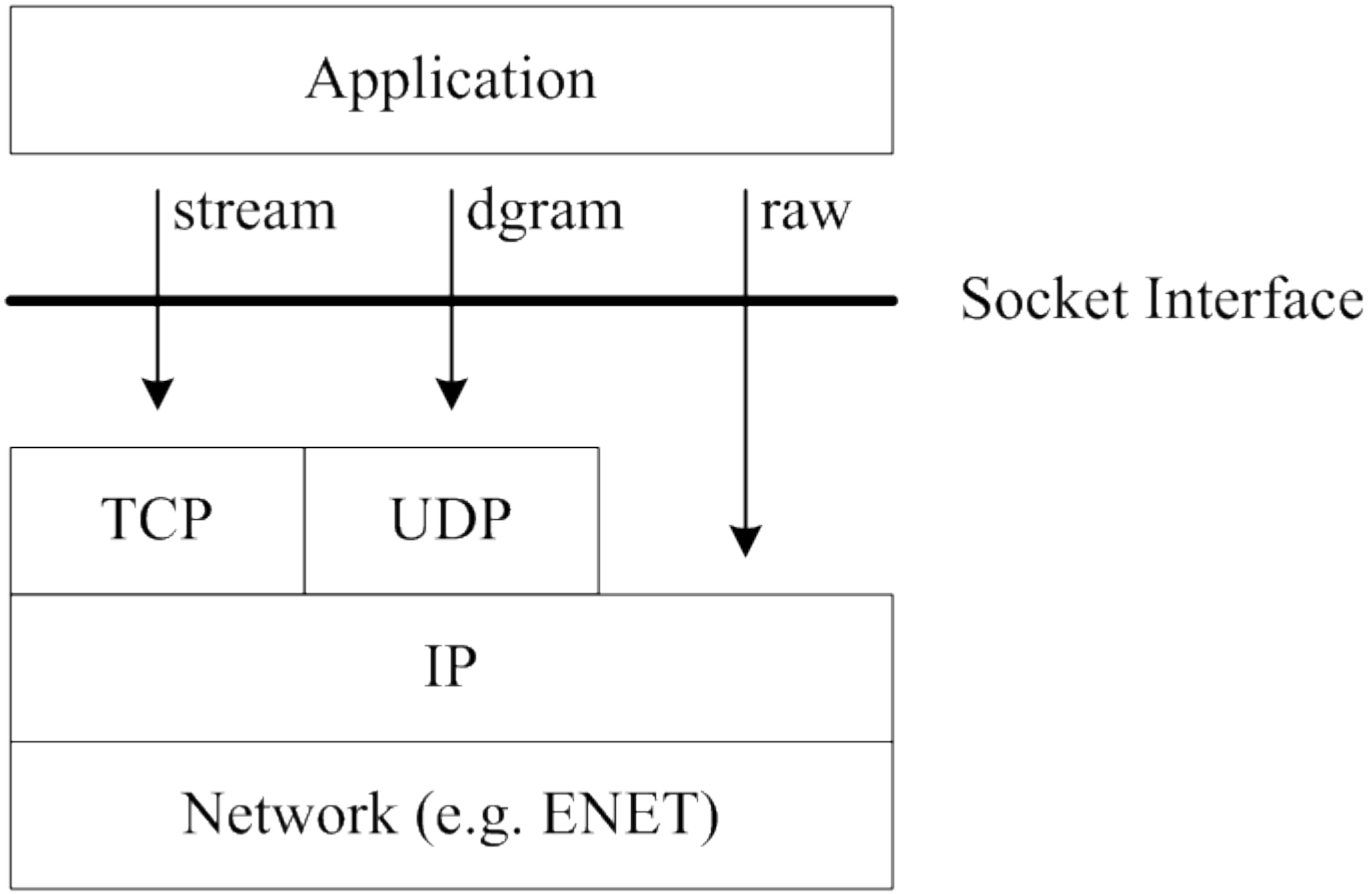
Sockets

FACHHOCHSCHULE TECHNIKUM WIEN

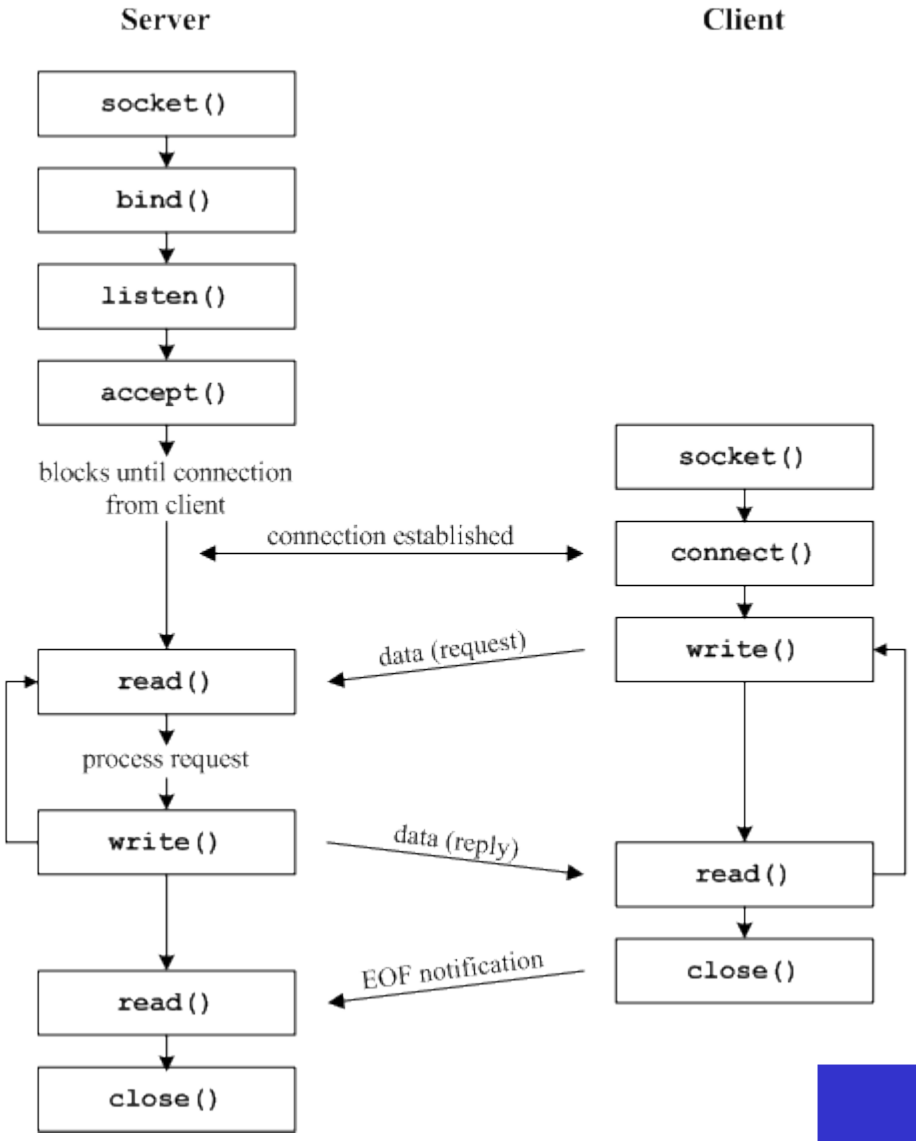
- Begriff “socket” unterschiedlich verwendet
  - **Socket** = Kombination **IP-Adresse:Port**
  - Socket API (Application Programming Interface) bzw. Socket Interface
  - `socket()` SysCall
- $\Rightarrow$ auf Kontext achten!!

Socket Interface

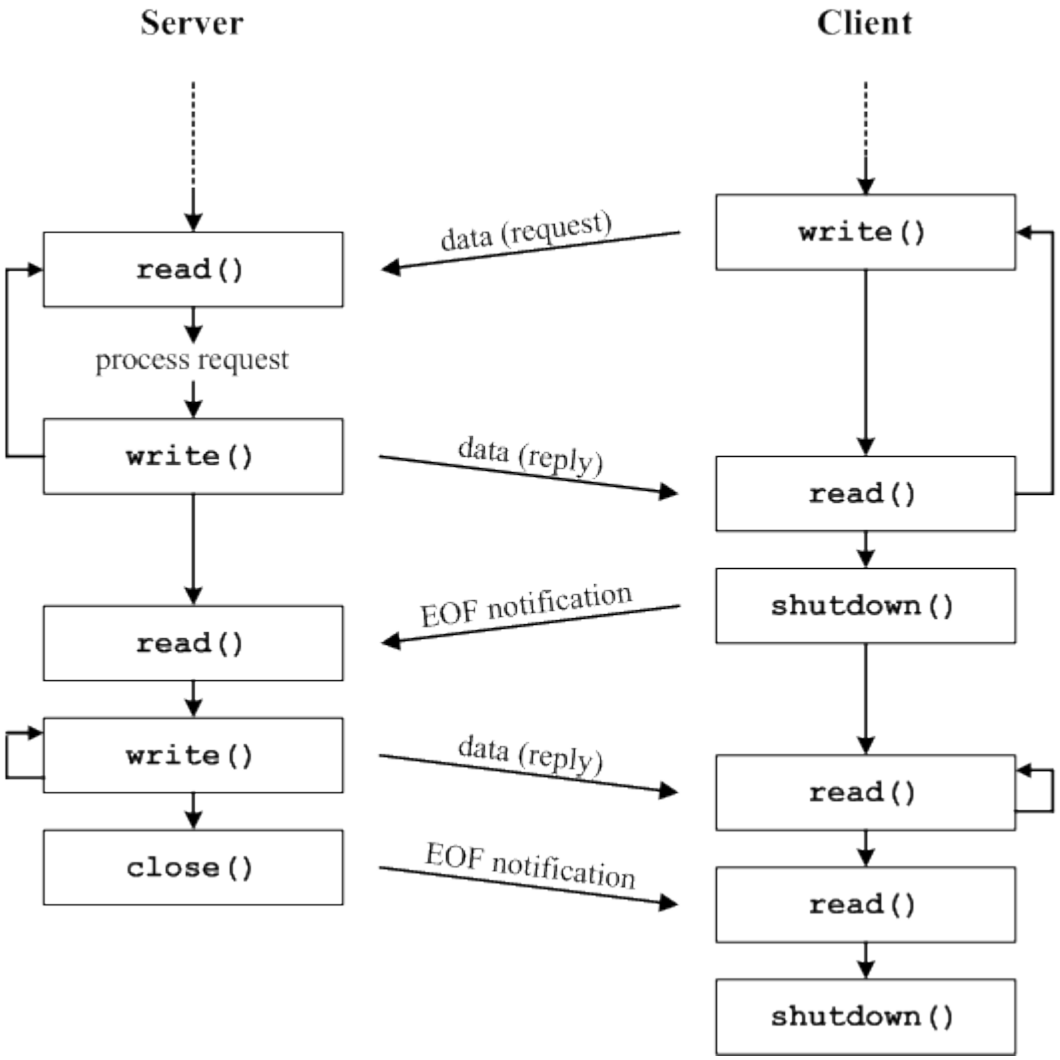
FACHHOCHSCHULE TECHNIKUM WIEN



TCP Server/Client SysCalls

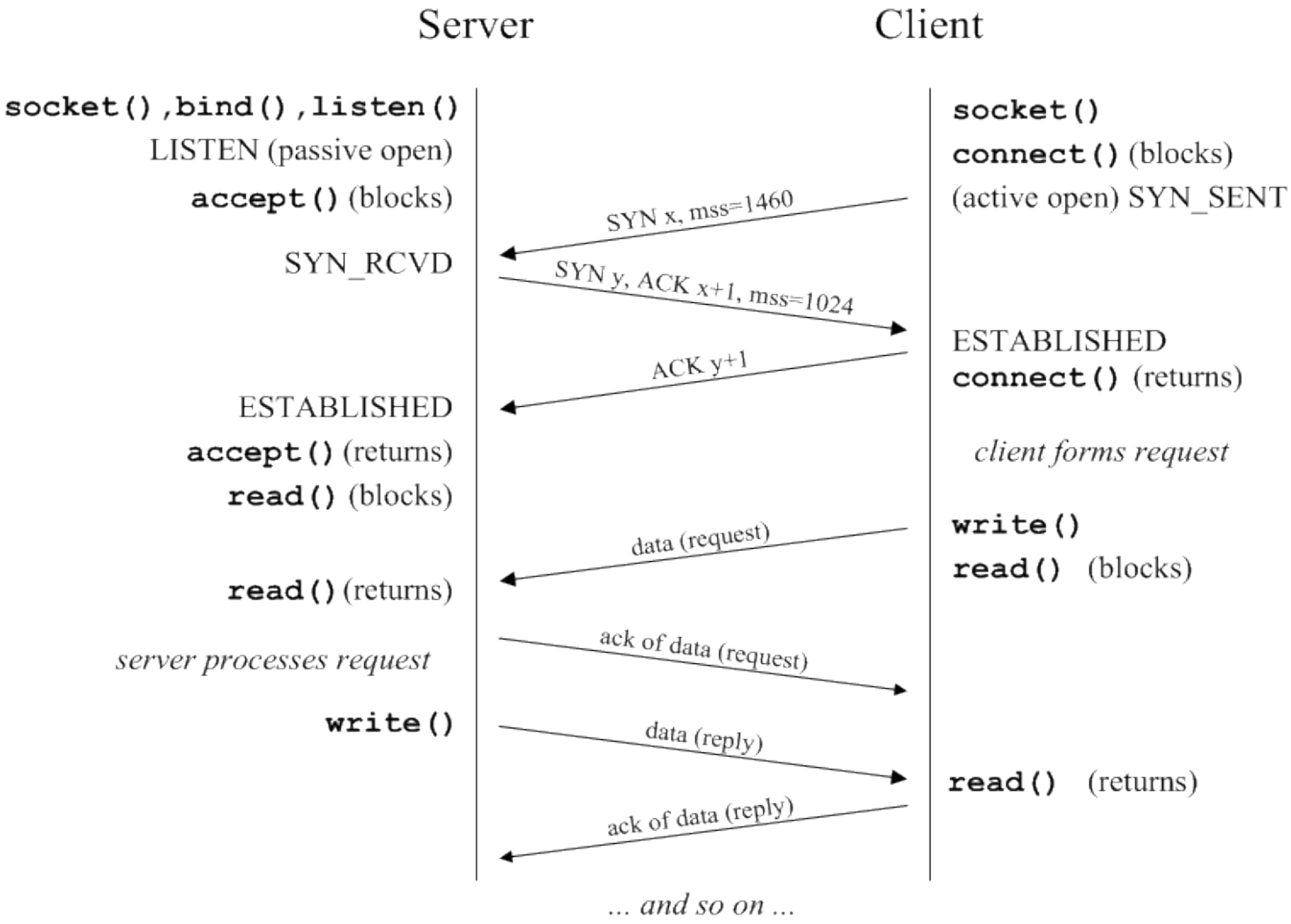


TCP Server/Client SysCalls (cont.)



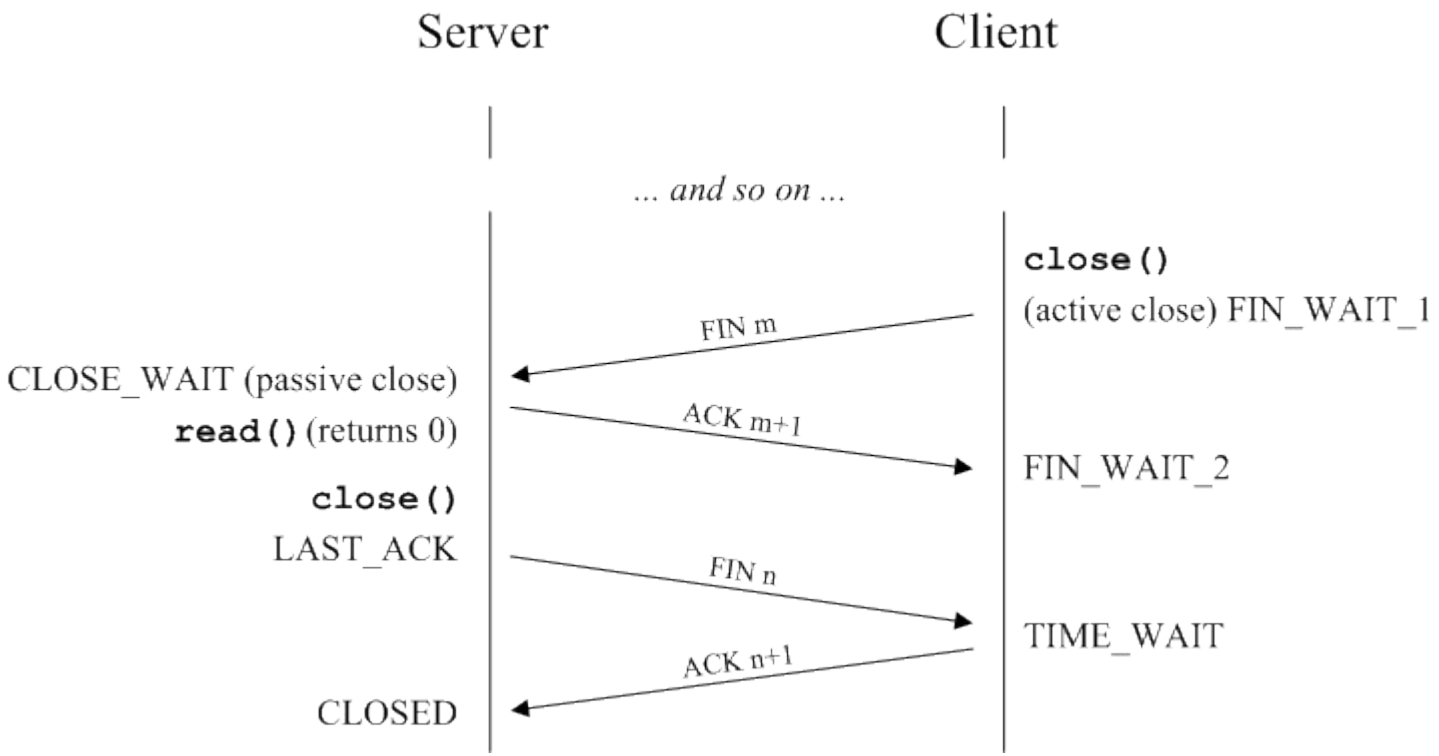
TCP Datenfluss

FACHHOCHSCHULE TECHNIKUM WIEN

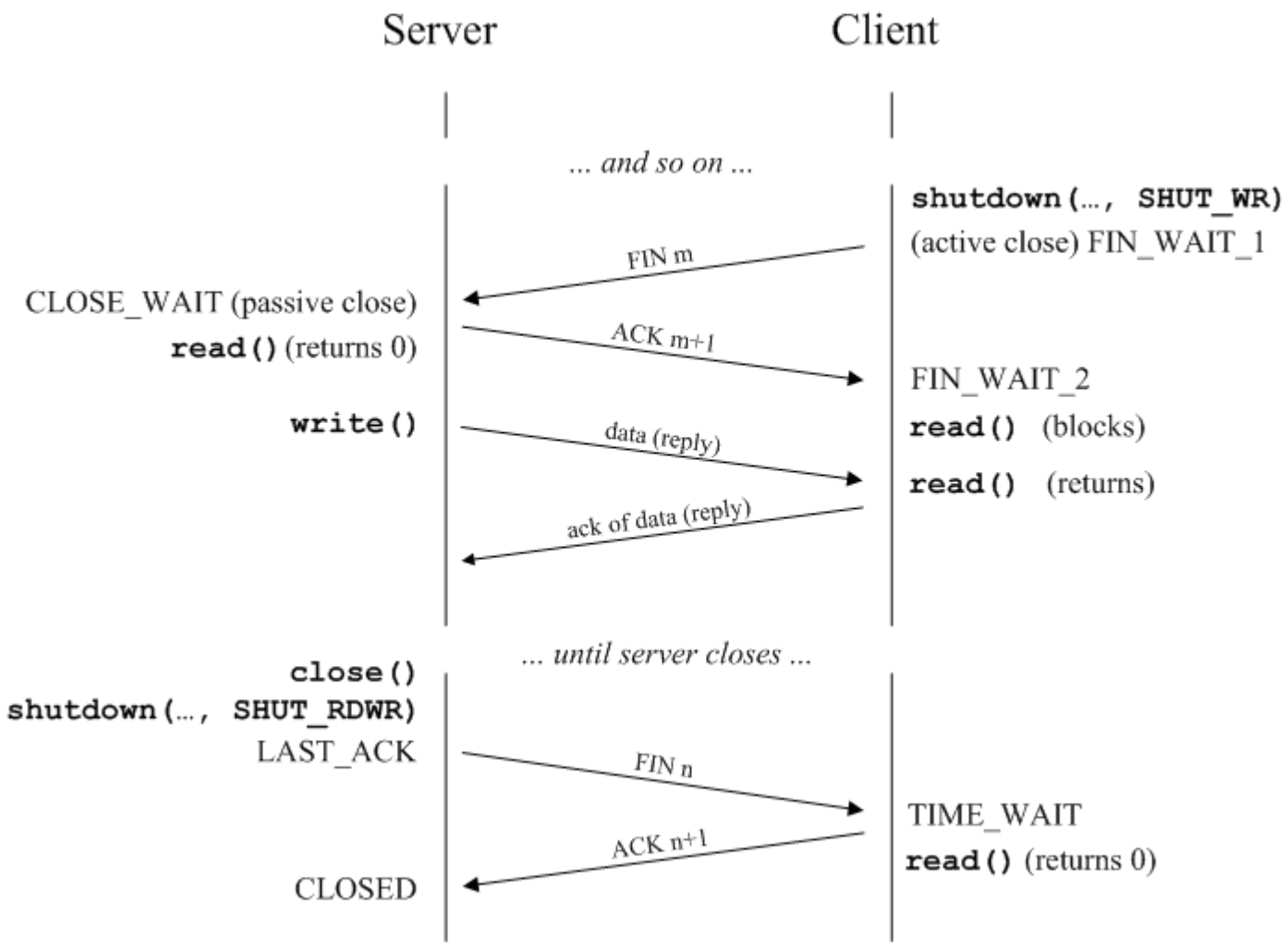


TCP Datenfluss (`close()`)

FACHHOCHSCHULE TECHNIKUM WIEN



TCP Datenfluss (`shutdown()`)



Austauch von strukturierten Daten

FACHHOCHSCHULE TECHNIKUM WIEN

- TCP ermöglicht Übertragung von Byte-Streams
- Übertragen von strukturierte Daten (d.h., Records) ⇒ Markierung/Erkennung von Feldgrenzen, Recordgrenzen, Requestgrenzen
  - **Sender:** Markierung dieser Grenzen
  - **Empfänger:** Erkennen dieser Grenzen

Markierung von Grenzen (1)

- Fixe Länge (d.h., Anzahl der Bytes pro Feld, Record, Request fix und bekannt)
- Länge als Teil des Feldes, Records, Requests (an bekannter Position)
  - z.B., HTTP (RFC 2616): Länge des Gesamtrequests ist Teil der HTTP Headers
- Expliziter Terminator
  - z.B., SMTP (RFC 5321): '.' in eigener Zeile
  - z.B., Newline als Terminator für ein Feld

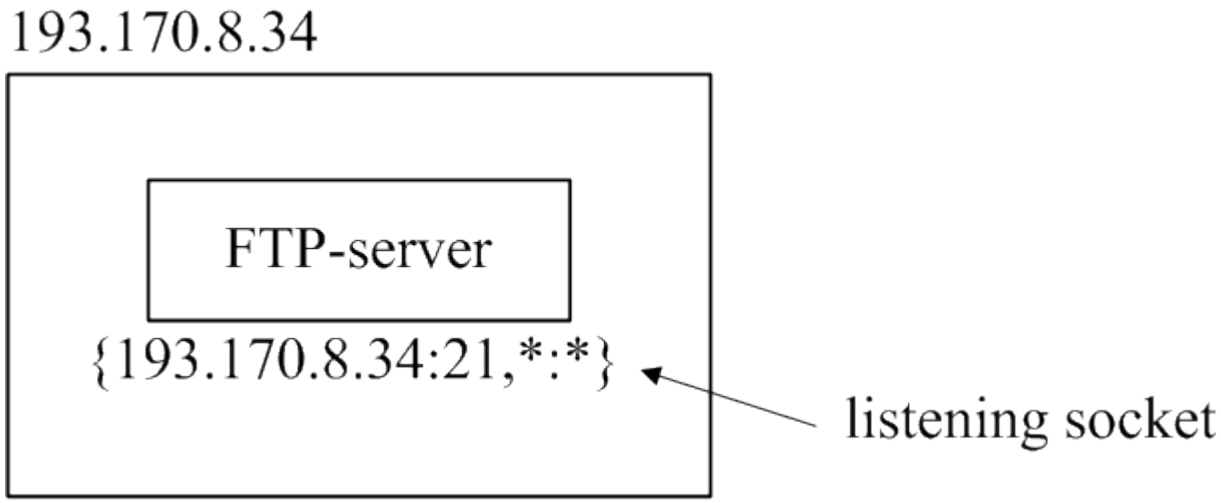
Markierung von Grenzen (2)

- EOF, d.h., Terminierung der Connection via **`close()`/`shutdown()`**
  - z.B., ECHO (RFC 862)

Verbindungsaufbau

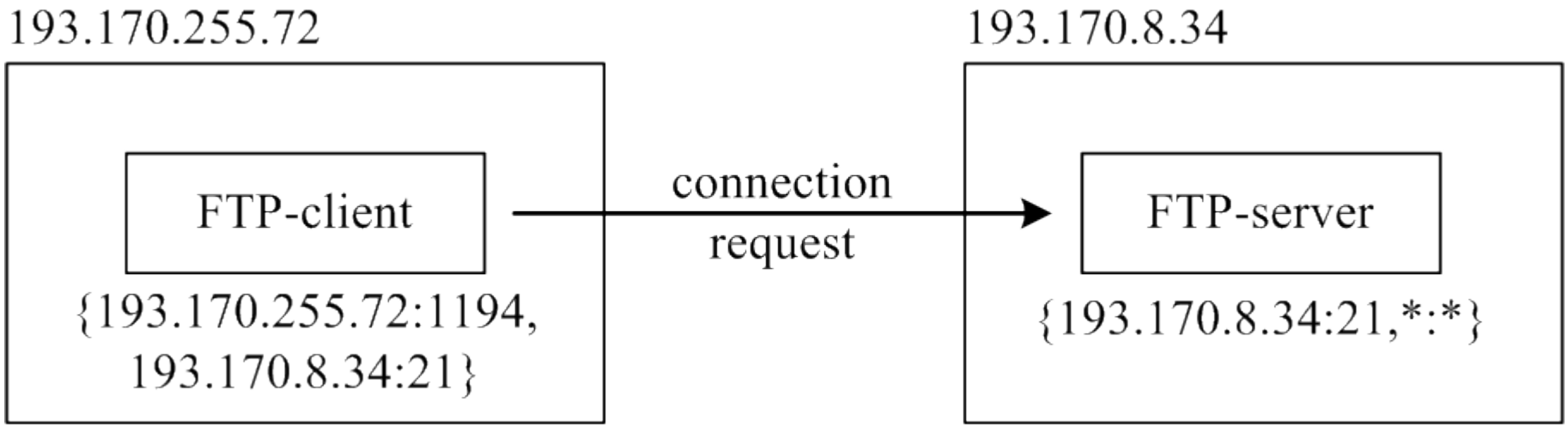
FACHHOCHSCHULE TECHNIKUM WIEN

- Server nach **passive open**, **listening socket** vorhanden



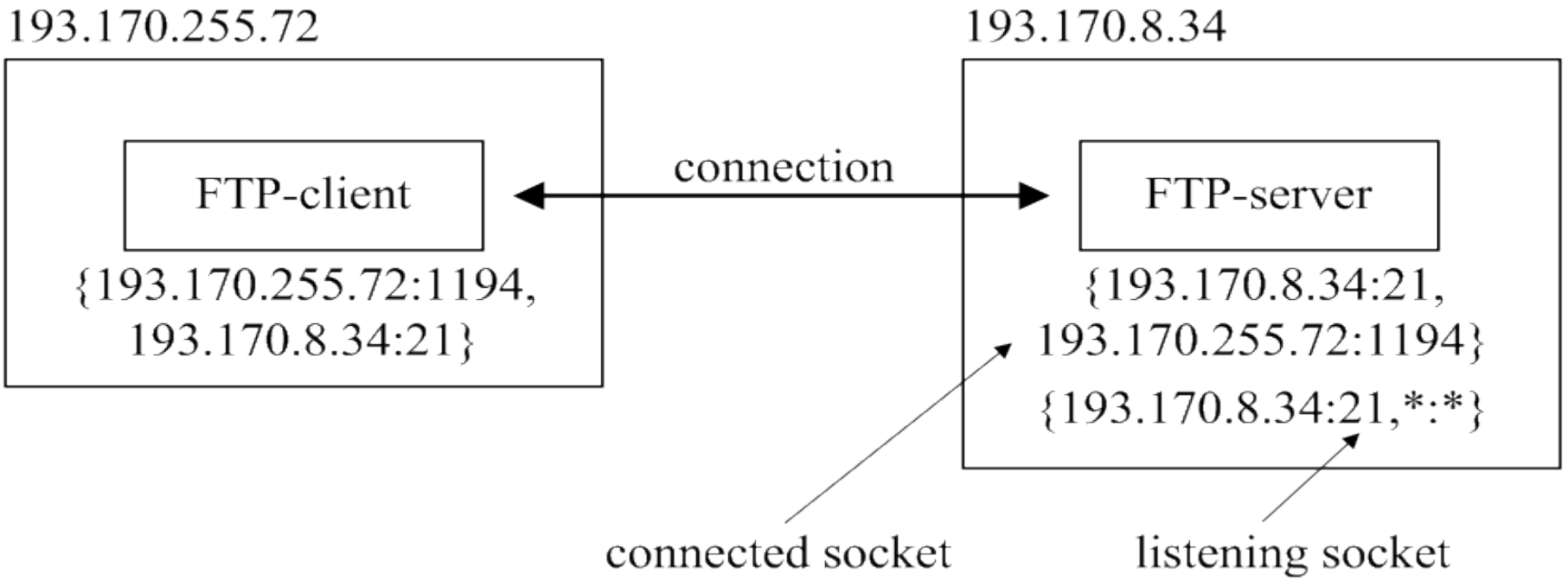
Verbindungsaufbau (cont.)

- nach der Verbindungsanforderung durch den Client



Verbindungsaufbau (cont.)

- nach Abschluss des Verbindungsaufbaus: Server verfügt über **vollständiges Socket-Paar**



Servertypen

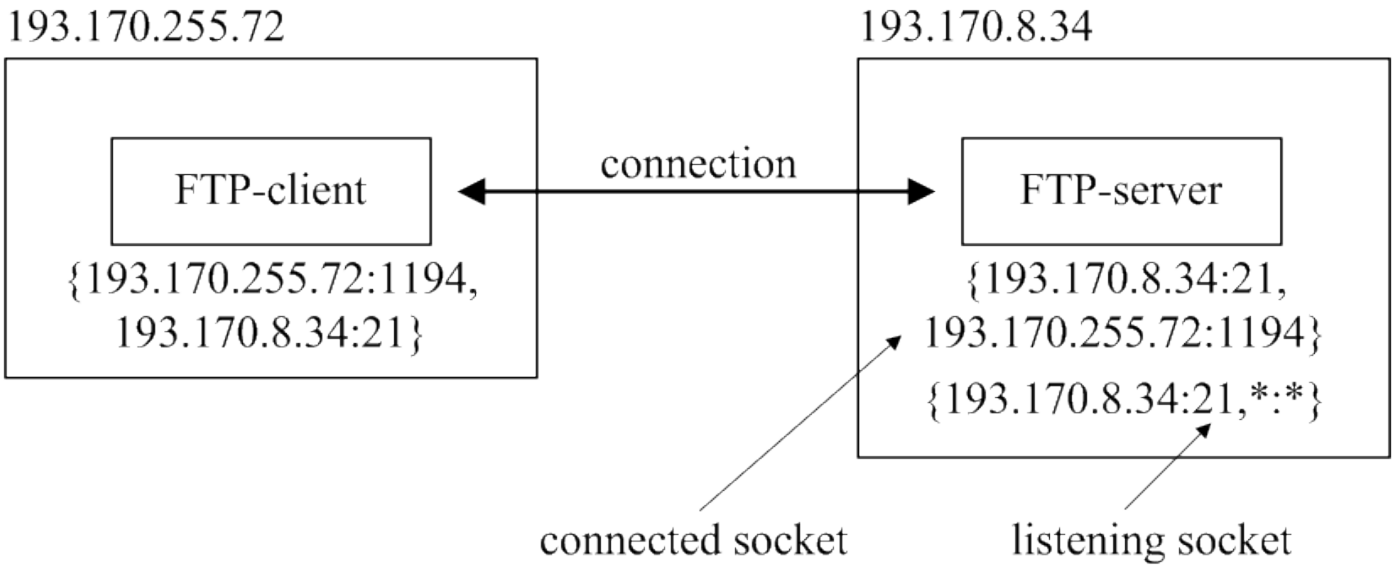
FACHHOCHSCHULE TECHNIKUM WIEN

- iterative server
  - Simple-Server, wie vorher beschrieben
- concurrent server
  - Forking-Server, Pre-Forking-Server
    - Spawning-Server
  - Threading-Server, Pre-Threading Server
  - Multipexing Server (with **`poll()`** or **`select()`**)

Forking Server

FACHHOCHSCHULE TECHNIKUM WIEN

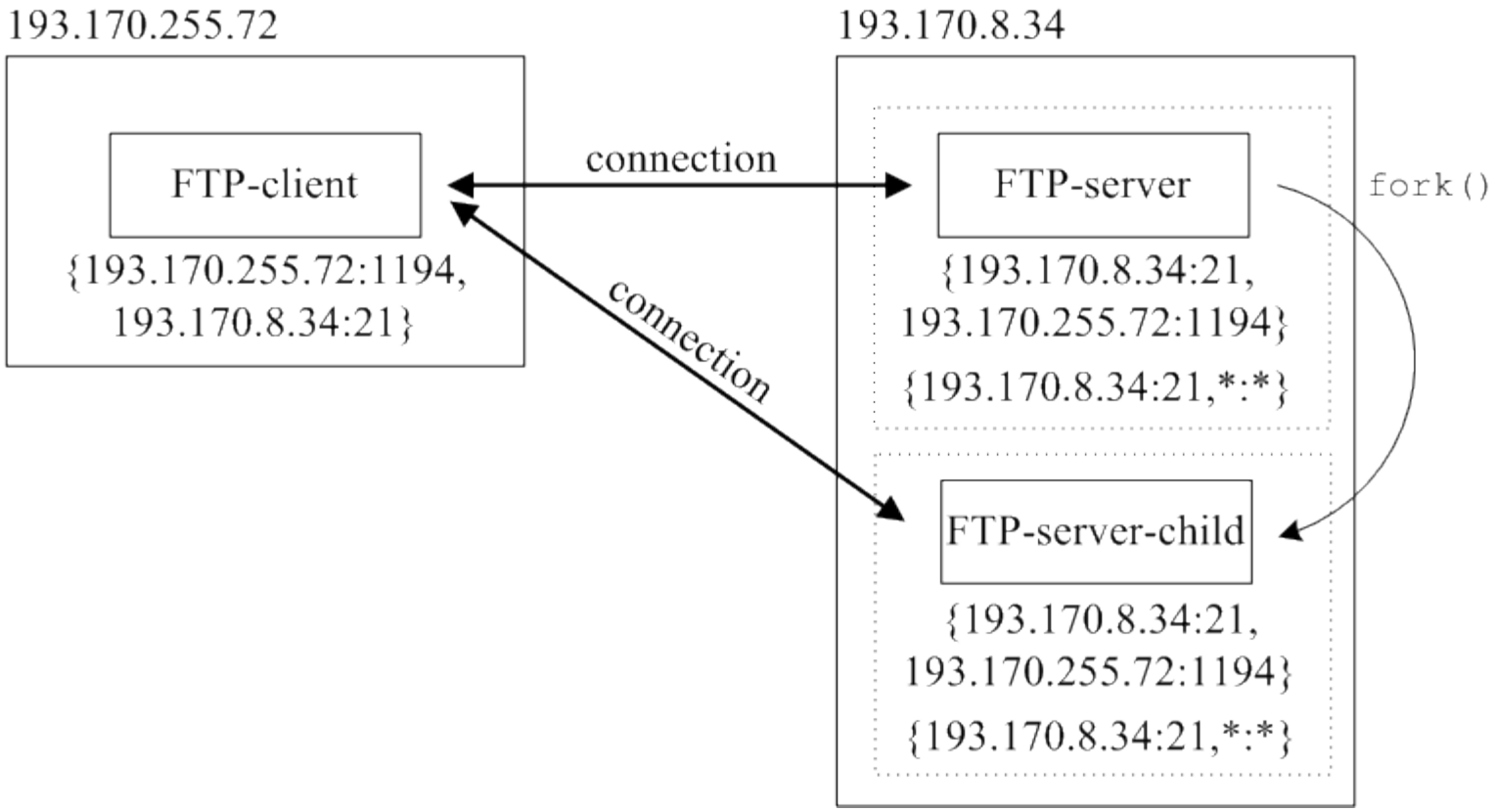
- Verbindungsaufbau wie bei Simple-Server



Forking Server (cont.)

FACHHOCHSCHULE TECHNIKUM WIEN

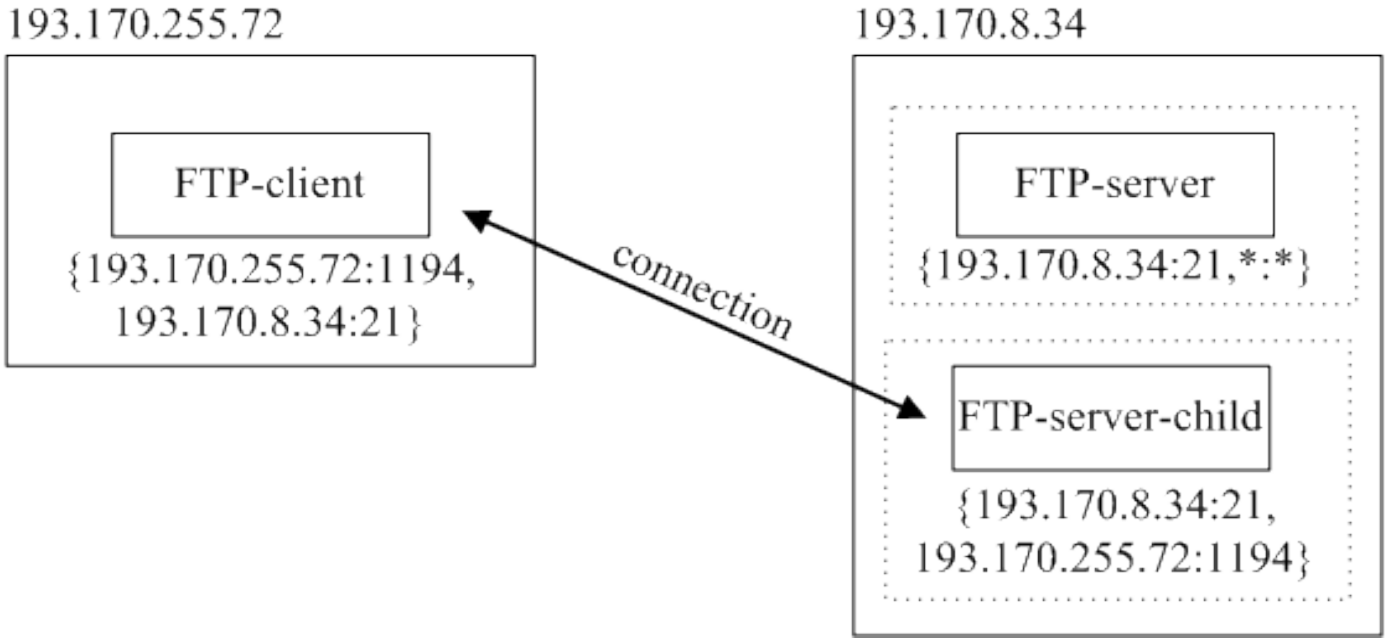
- SysCall **`fork()`** dupliziert u.a. alle *file descriptoren*, und damit auch alle *sockets*



Forking Server (cont.)

FACHHOCHSCHULE TECHNIKUM WIEN

- Server Vaterprozess schließt *connected socket*
- Server Kindprozess den *listening socket*



Spawning Server

FACHHOCHSCHULE TECHNIKUM WIEN

- Verbindungsaufbau immer ähnlich
- minimaler Forking-Server wartet auf Verbindungsanforderung
- nach Abschluss des Verbindungsaufbaus wird wie beim Forking-Server ein Server Kindprozess erzeugt
- Server Kindprozess startet den eigentlichen Server Dienst / “Business Logic” (**`exec()`** SysCall Familie)
- “Business Logic” weiß u.U. gar nicht, dass sie über eine Netzwerkverbindung kommuniziert
  - stdin und stdout wird vor **exec()** auf socket umgeleitet

Spawning Server, Beispiel

- inetd – internet “super-server”, xinetd
- viele Dienste werden über inetd gestartet:
  - ftp
  - telnet
  - rsh
  - ....

Zombie Prozess

FACHHOCHSCHULE TECHNIKUM WIEN

- Was ist ein Zombie Prozess?
  - (Linux: **ps** zeigt **<defunct>**, siehe **`man ps`**)
- Mögliche Probleme
- Wie kann man einen Zombie Prozess verhindern?
  - Vaterprozess muss Status abfragen: **`waitpid()`**
  - am besten im Signal-Handler für **`SIGCHLD`**
- siehe Beej's Guide to Network Programming

Projekt: Simple Bulletin Board

FACHHOCHSCHULE TECHNIKUM WIEN

- Phase 1
  - Lektoren stellen fertige Executables für Client und Server zur Verfügung (inkl. Manual Pages)
  - Studierende analysieren mittels tcpdump das Protokoll zwischen Client und Server
  - Abgabe: Protokollanalyse (PDF)
- Phase 2
  - Studierende implementieren Client und testen diesen gegen den Server der Lektoren
- Phase 3
  - Studierende implementiere Server und testen diesen den Client der Lektoren (sowie gegen den eigenen Client)
  - Abgabe: Source Client und Server (inkl. Abgabegespräch)

tcpdump

FACHHOCHSCHULE TECHNIKUM WIEN

sudo /usr/sbin/tcpdump -vv -X -s 0 -i lo port <port>

Weiterführende Literatur

FACHHOCHSCHULE TECHNIKUM WIEN

- Beej's Guide to Network Programming - Using Internet Sockets (http://beej.us/guide/bgnet/)
- Unix Network Programming, volumes 1-2 by W. Richard Stevens. Published by Prentice Hall. ISBNs for volumes 1-2: 013141155142, 013081081943.
- TCP/IP Illustrated, volumes 1-3 by W. Richard Stevens and Gary R. Wright. Published by Addison Wesley. ISBNs for volumes 1, 2, and 3 (and a 3-volume set): 020163346947, 020163354X48, 020163495349, (020177631650).
- Advanced Programming in the UNIX Environment by W. Richard Stevens. Published by Addison Wesley. ISBN 020143307952.
- RFC (Request for Comments) series. ISSN 2070-1721 (http://www.rfc-editor.org/RFCoverview.html)