

Client/Server-Systeme

Prof. Dr.-Ing. Wilhelm G. Spruth

WS 2006/2007

Teil 2

Sockets

Literatur

„Socket-Schnittstelle“

J. Martin, J. Leben:

„TCP/IP Netzwerke“.
Prentice Hall, 1994, Kapitel 18.

D. E. Comer, D. L. Stevens:

„Internetworking with TCP/IP“.
Prentice Hall, 1991, Band 1, Kapitel 21.

D. E. Comer, D. L. Stevens:

„Internetworking with TCP/IP“.
Prentice Hall, 1991, Band 3, Kapitel 5.

J. Bloomer:

„Power Programing with RPC“.
O´Reilly&Assiciates, 1992, S. 125.

**Schicht 7
Anwendung**

Telnet, FTP, SMTP

**Schicht 6
Darstellung**

**Schicht 5
Sitzung**

**Schicht 4
Transport**

**TCP Transmission Control
Protokoll**

**Schicht 3
Netzwerk**

**IP
Internet Protokoll**

**Schicht 2
Sicherung**

**Ethernet, Tokenring
HDLC, ATM**

**Schicht 1
Bitübertragung**

Kupfer, Glasfaser, drahtlos

Einordnung von TCP/IP in das OSI Schichtmodell

**Schicht 7
Anwendung**

**SNMP, Telnet, FTP
vom Benutzer geschriebene
Anwendungen**

**Schicht 6
Darstellung**

**XDR, ASN.1, Kompression,
Sicherheit, Authentifizierung**

**Schicht 5
Sitzung**

**RPC (Remote Procedure Call)
Sockets, DCE, CORBA, RMI, SOAP**

**Schicht 4
Transport**

**TCP
Transmission Control Protokoll**

**Schicht 3
Netzwerk**

IP Internet Protokoll

**Schicht 2
Sicherheit**

**Ethernet, Tokenring
HDLC, ISDN, ATM**

**Schicht 1
Bitübertragung**

Kupfer, Glasfaser, drahtlos

Einordnung von TCP/IP in das OSI Schichtmodell

Socket

Kommunikationsendpunkt, bestehend aus Netzwerk-ID, Rechner-ID und Portnummer

Port

Eindeutige Zuordnung der TCP oder UDP Pakete zur nächst höheren Schicht 5

Beispiel

Socket	Telephon
Netzwerk-ID	Ortsvorwahl
Rechner-ID	Telephon Nr.
Portnummer	Nebenstelle

Sockets: Schnittstelle zu Netzwerkprotokollen

Sockets sind eine allgemeine Schnittstelle zu unterschiedlichen Transportprotokollen der Schicht 4 (z.B. NetBIOS, SNA, IPX) und verbergen weitgehend Architektur des verwendeten Transportprotokolls. Sie sind die Defacto Standard Schnittstelle zu den TCP/IP Stacks in allen Betriebssystemen. Bei einem Austausch des Transportprotokolls muß Applikationsprogramm nur geringfügig verändert werden.

Ein Socket ist ein Endpunkt einer Kommunikationsverbindung. Die Verbindung ist sowohl lokal als auch entfernt möglich. Bei einer entfernten Verbindung wird über den Socket eine Anwendung mit einem Transportmechanismus (z.B. TCP oder UDP) verbunden.

Für TCP- und UDP-Protokolle ist die Socketadresse eindeutig durch die Kombination Internetadresse plus Portnummer definiert.

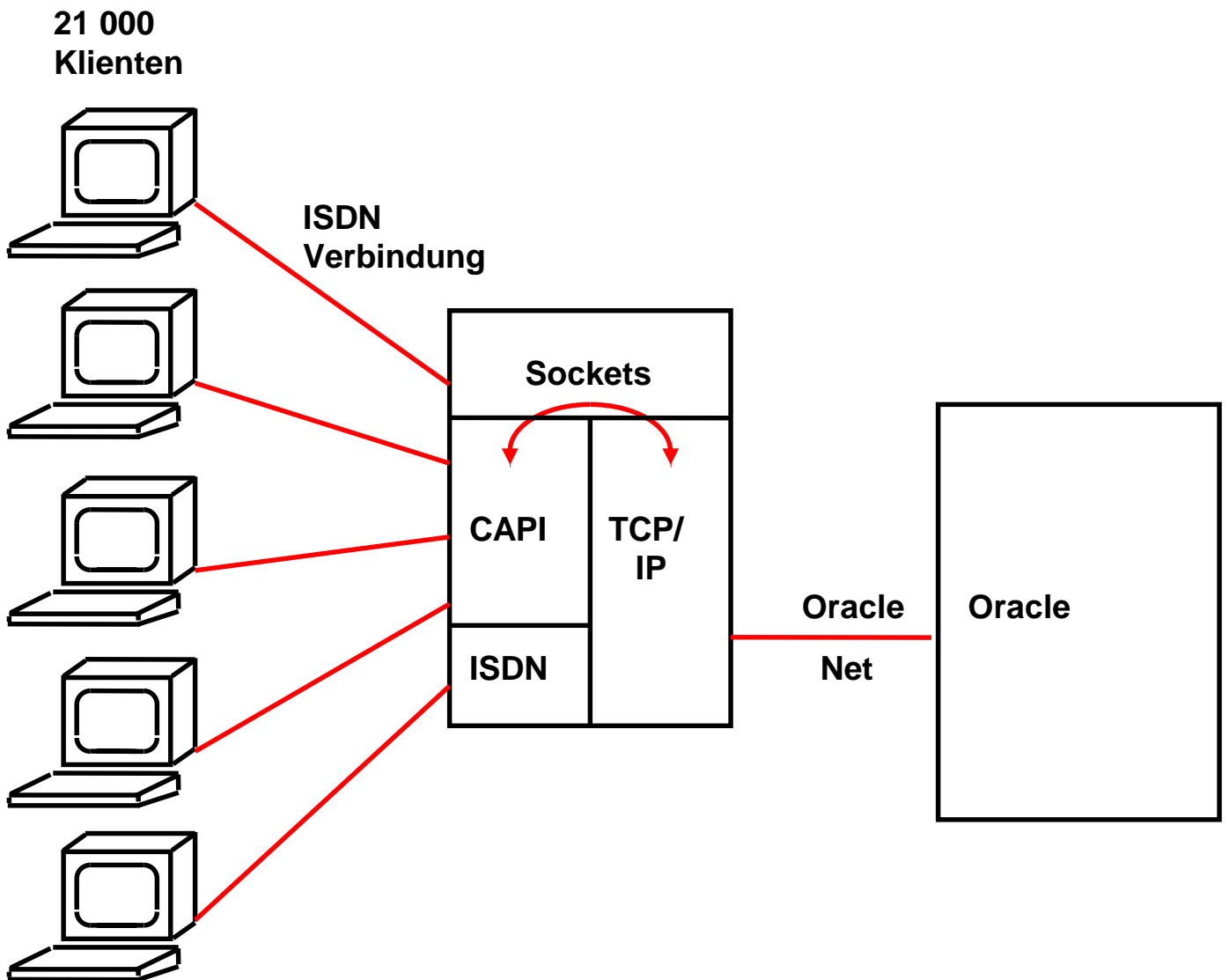
Der Socket Kommunikationsendpunkt besteht aus der IP Adresse (Netzwerk-ID und Rechner-ID) und der Portnummer. Ein Port ist eine eindeutige Zuordnung der TCP oder UDP Pakete zur nächst höheren Schicht 5.

Eine TCP/IP Verbindung ist immer eindeutig durch 6 Größen bestimmt:

IP Adresse des Senders	IP Adresse des Empfängers
Port des Senders	Port des Empfängers
Protokoll des Senders	Protokoll des Empfängers

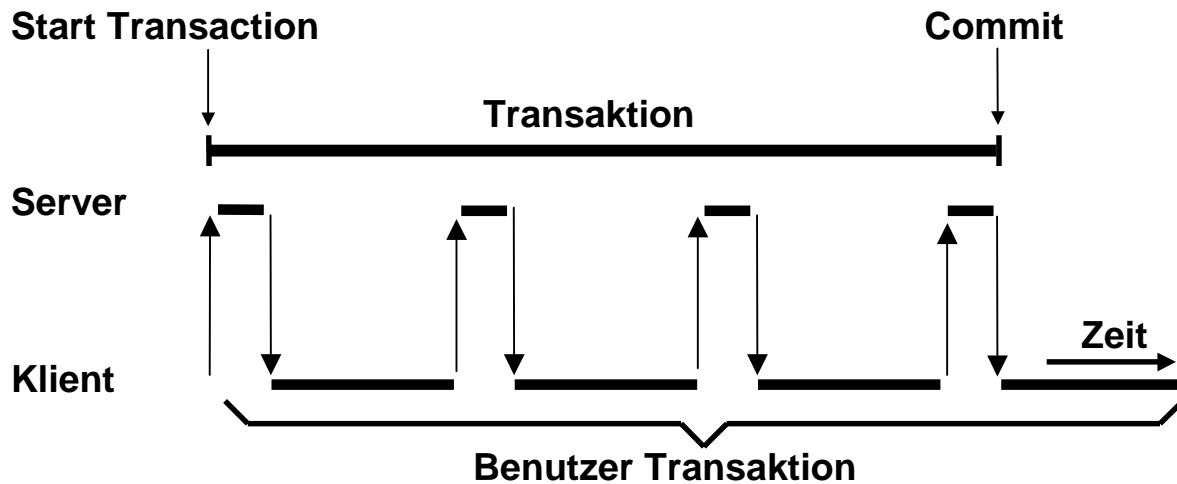
Beispiel

Socket	Telephon
Netzwerk-ID	Ortsvorwahl
Rechner-ID	Telephon Nr.
Portnummer	Nebenstelle



Ticket Service der Deutschen Post

Verbindungsloses Protokoll Verbindungsorientiertes Protokoll



Sitzung - Session

1. Open
2. Do Work
3. Close

Beispiel: Internet Zugriff auf Amazon, ebay benutzt verbindungsloses http Protokoll.

Lösung:

Sitzungsstatus (session state) entweder auf dem Klienten oder auf dem Server speichern.

Drei Typen von Sockets

Stream Sockets (SOCK_STREAM) für zuverlässige verbindungsorientierte Duplex-Verbindungen. Werden im Internet-Bereich durch das TCP Protokoll unterstützt.

TCP verwendet die Abstraktion einer "Verbindung", die aus zwei Endpunkten, definiert durch 2 Integer Paare (IP-Adresse/Portnummer) besteht. Der Zustand einer Verbindung wird durch den "Transmission Control Block" (TCB, nicht zu verwechseln mit dem "Task Control Block des Betriebssystems) gesteuert. Der TCB wird beim Aufbau einer Verbindung erzeugt, während der Dauer der Verbindung modifiziert, und beim Abbau der Verbindung wieder gelöscht.

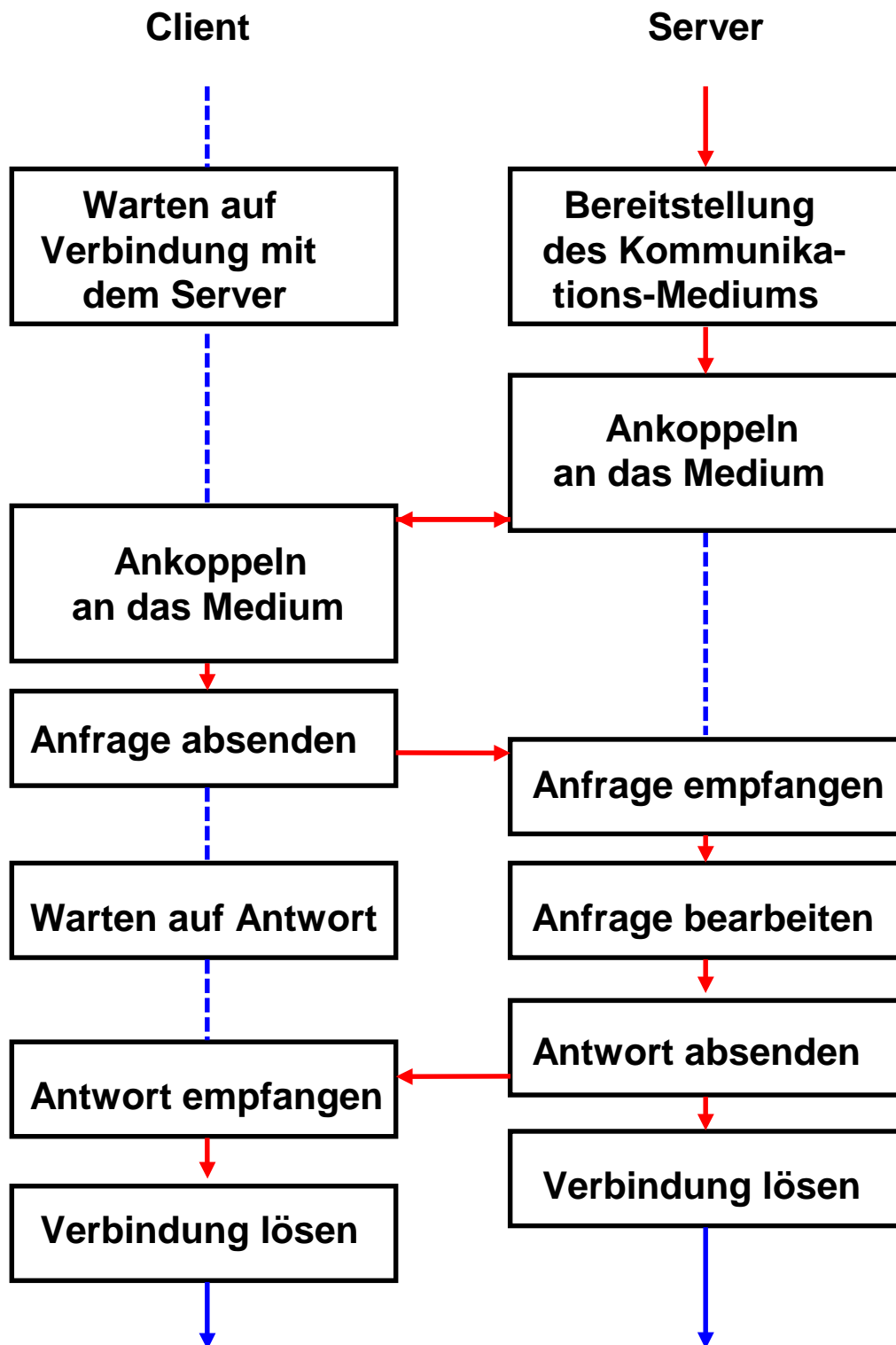
Datagram Sockets (SOCK_DGRAM) für verbindungslose, nicht-garantierte Nachrichtenübertragung. Werden im Internet-Bereich durch das UDP Protokoll unterstützt.

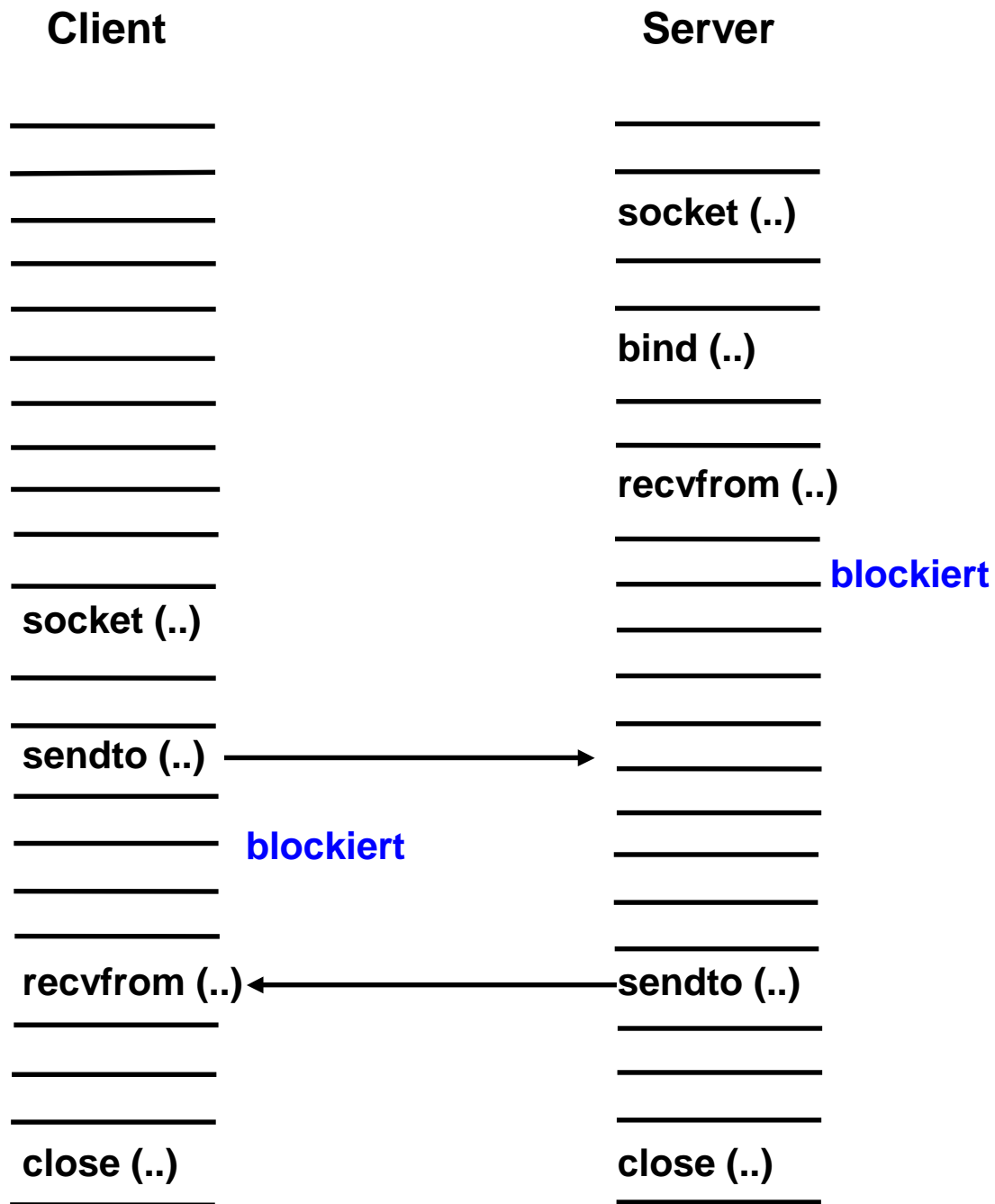
UDP betrachtet einen Port als eine FIFO Warteschlange, in die es IP Datagramme überträgt.

Raw Sockets (SOCK_RAW) greifen direkt auf untere Ebenen der Netzwerkarchitektur zu. Sie ermöglichen Eigenentwicklungen von Transportmechanismen. Beispiel: „Light Weight Sockets“ für Switch-gekoppelte Parallelrechner.

Datagram Sockets

Einfache Client/Server-Transaktion



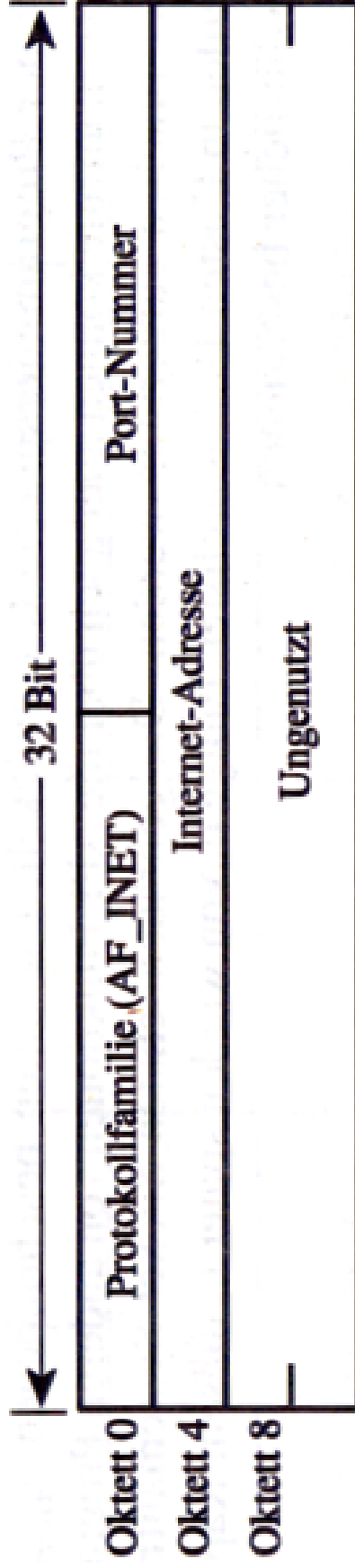


Verbindungsunabhängiges Socketprotokoll (Datagram Socket)

```

struct sockaddr_in {
    short  sin_family;   /* AF_INET          */
    u_short sin_port;    /* 16-Bit Port-Nummer */
    struct in_addr sin_addr; /* 32-Bit Internet-Adresse */
    char    sin_zero[8]; /* Netzwerk-Byte sortiert */
};

```



Datenstruktur für Socket Adressen

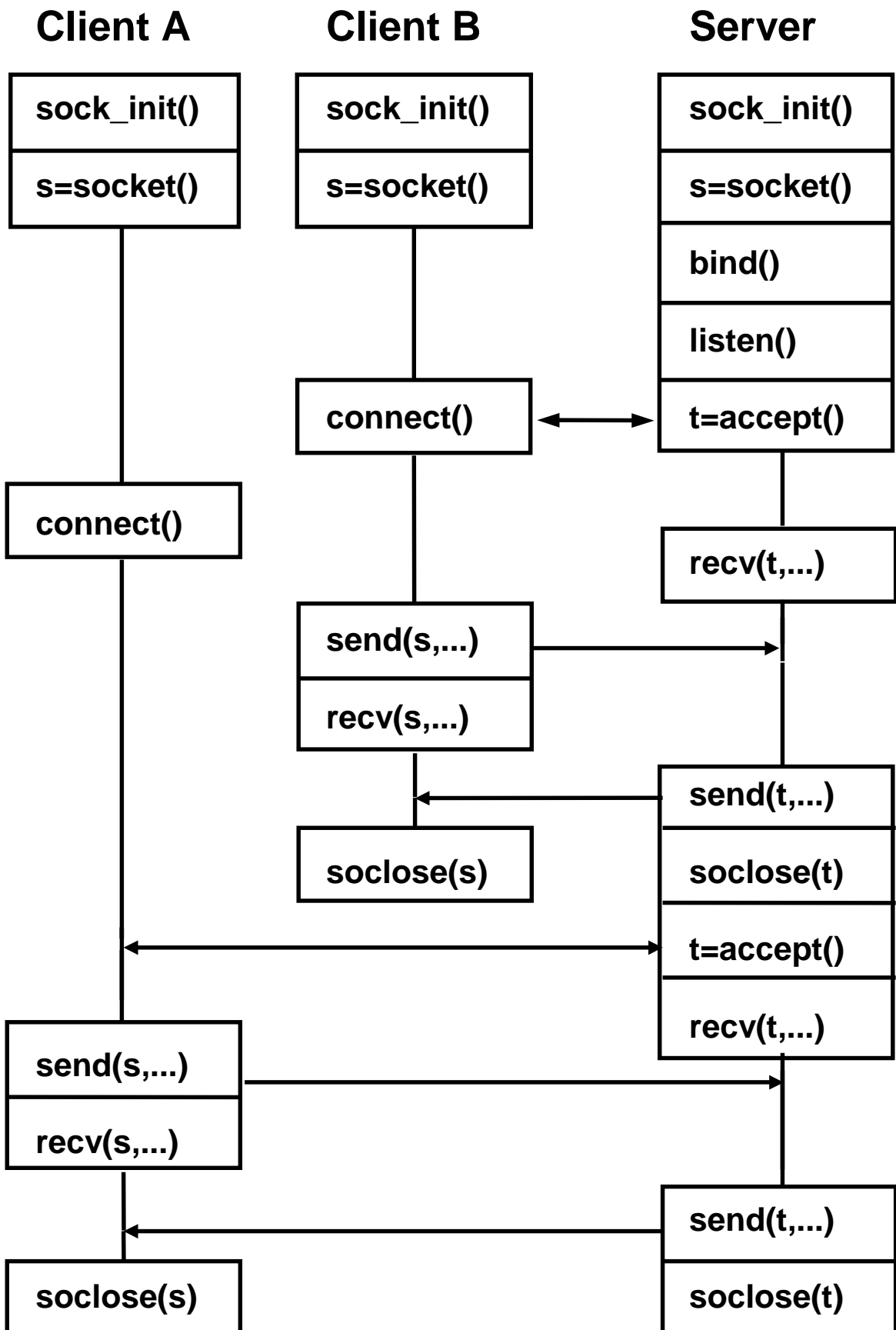
Stream Sockets

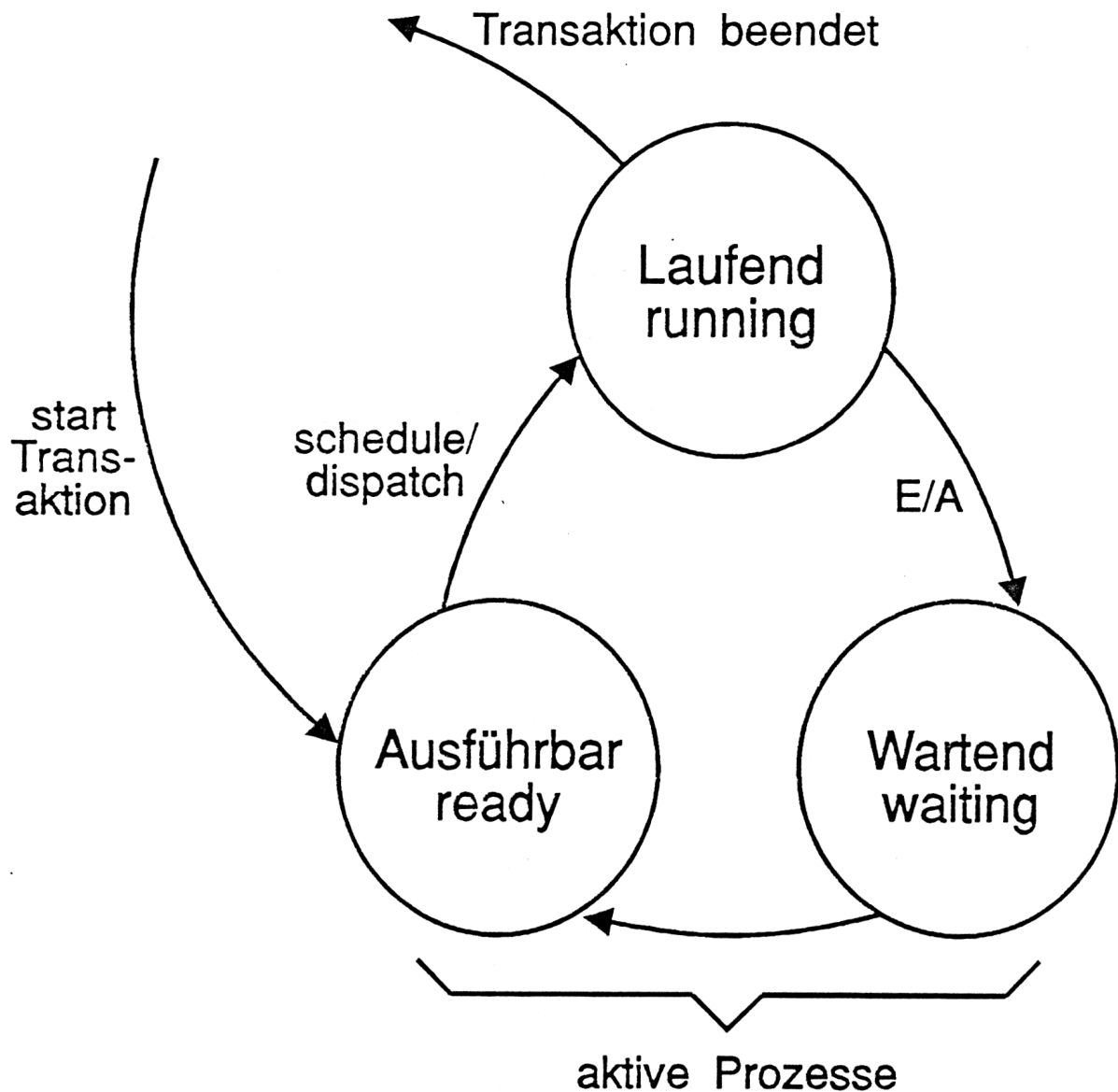
	Befehl
1. Socket erzeugen (Datenstruktur innerhalb des Adressenraums Datenbereich des Betriebssystems)	socket (...
2. Socket mit einem nach außen sichtbaren Namen versehen (mit einer TSAP Adresse)*	bind (...
3. Warteschlange erstellen, um eingehende Nachrichten zu speichern	listen (...
4. Auf Nachricht warten. Socket Deskriptor an child process weitergeben	accept (...
5. Verbindung zwischen 2 Sockets herstellen	connect (...
6. Daten senden, empfangen	send (... recv (...
7. Beenden	close (...

Stream Socket Transportdienst Aufrufe

- * Nachdem dies geschehen ist, kann der Name bekanntgegeben bzw. verteilt werden

Sequentieller Serverbetrieb





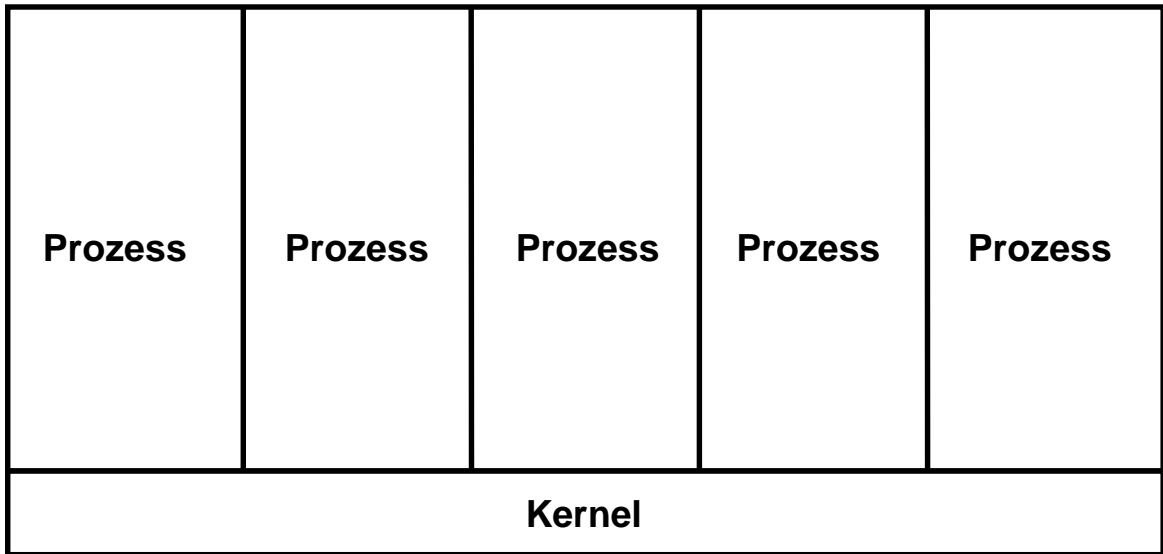
Ausführungsmodell für Transaktionen

Prozesse und Transaktionen verbringen den größten Teil der Verarbeitungsdauer mit dem Abschluss von E/A Operationen (Plattenspeicher).

Ohne Multiprogrammierung ist kein sinnvoller C/S Betrieb möglich. Zwei Alternativen:

- Mehrfache Prozesse in eigenen virtuellen Adressräumen
- Threads

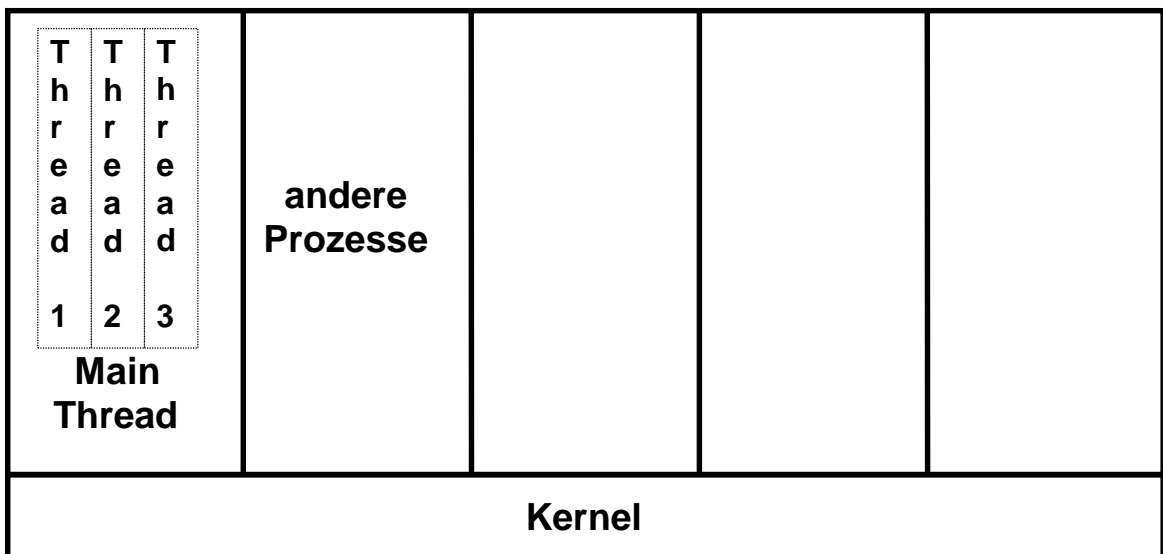
FF..FF



00..00

Prozess Ansatz

FF..FF

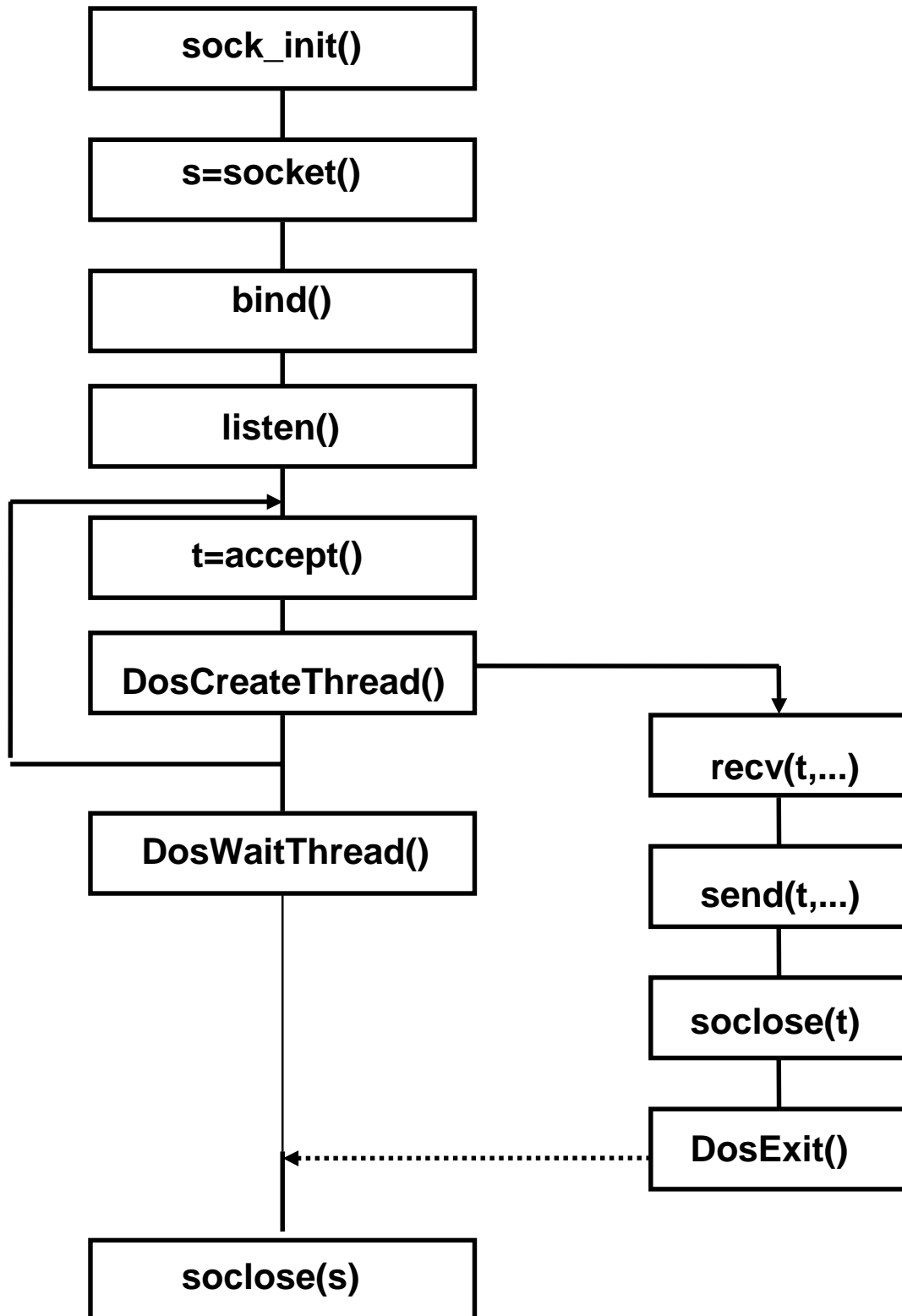


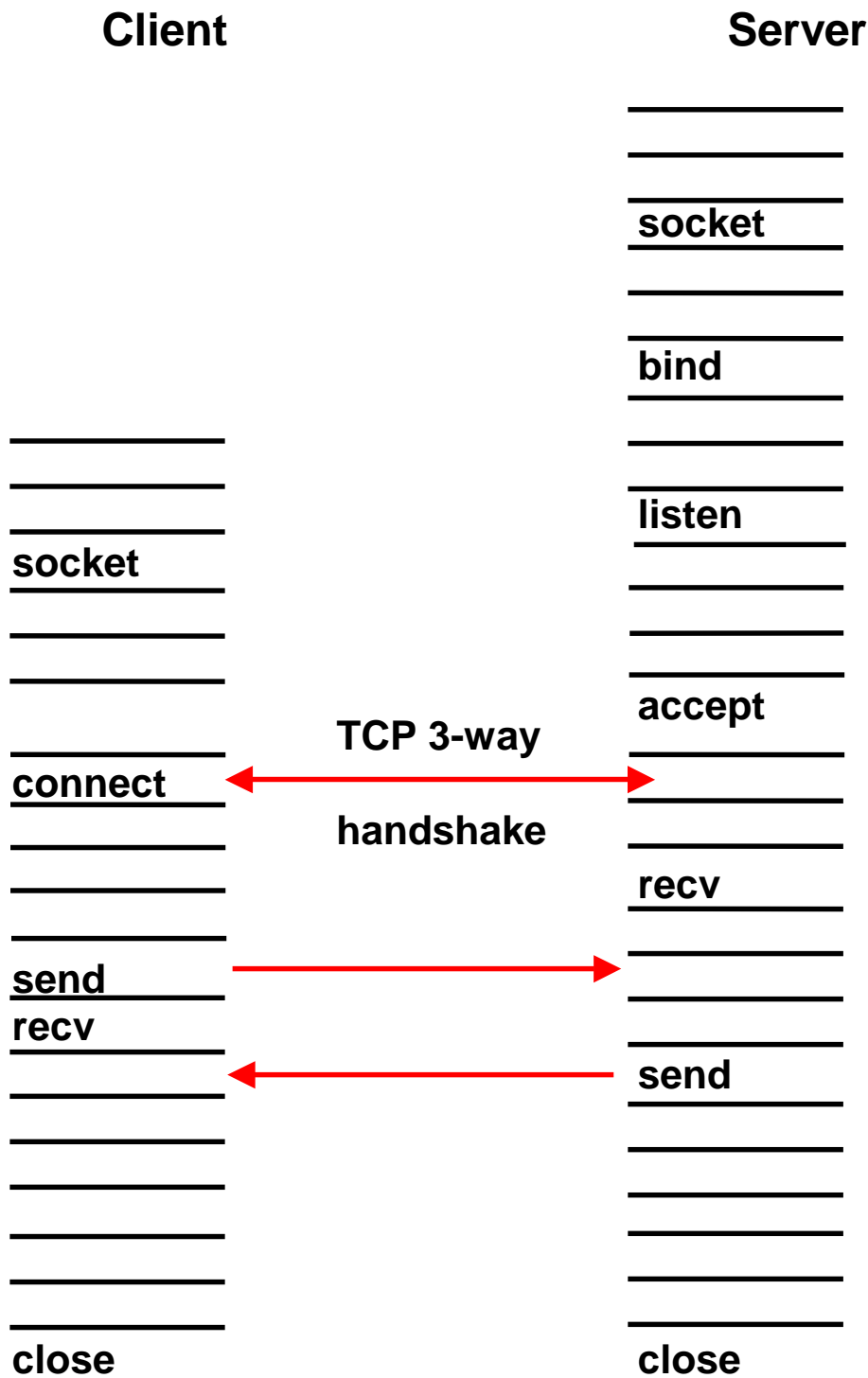
00..00

Thread Ansatz

Prozesse laufen in getrennten virtuellen Adressenräumen. Threads sind unabhängige Ausführungseinheiten, die innerhalb des gleichen virtuellen Adressenraums ablaufen. Ein Wechsel zwischen Threads erfordert deutlich weniger Aufwand als ein Wechsel zwischen Prozessen.

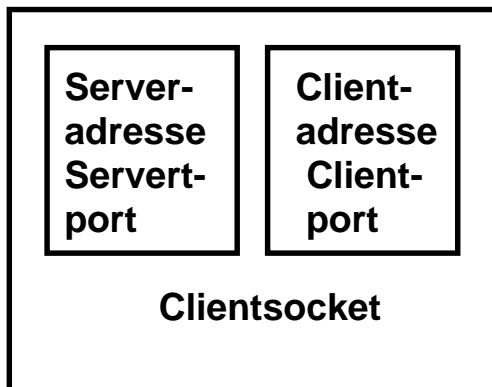
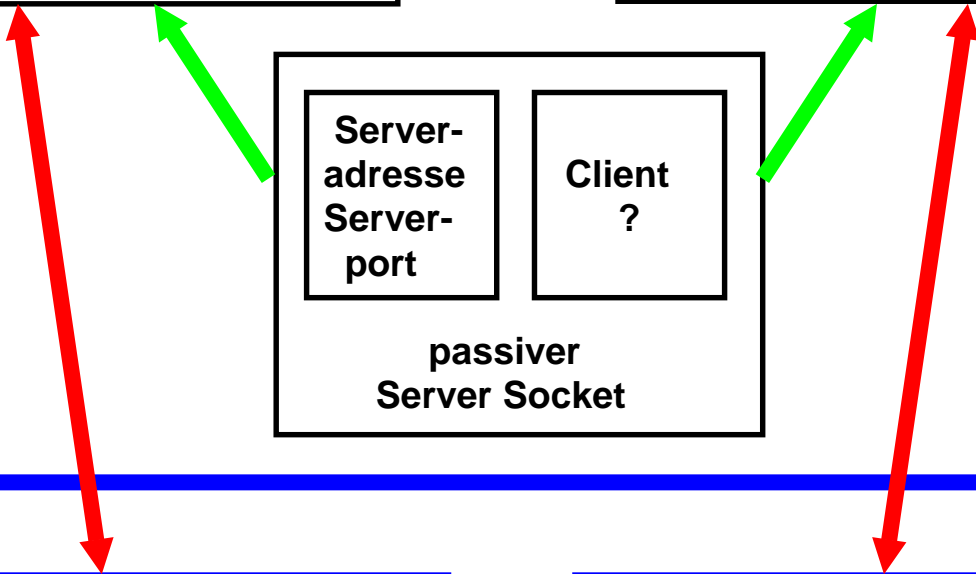
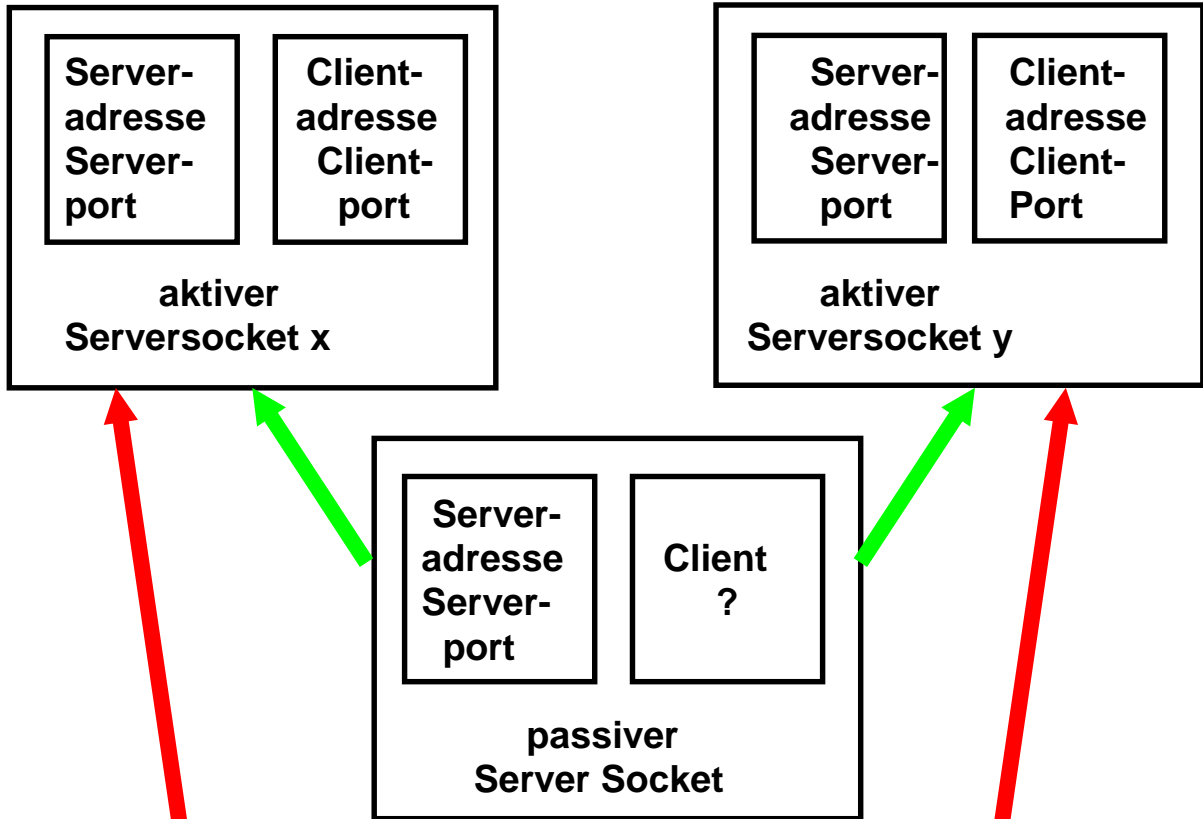
Paralleler Server



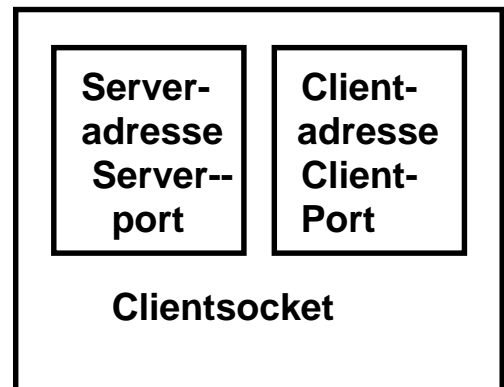


Verbindungsabhängiges Socketprotokoll

Server



Client I



Client II

Multiprogrammierter Betrieb

listen(r_sd, 5)

Mehrfache Kopien des Server Sockets erzeugen

Symbolische Konstanten für Socket Call Parameter

**BSD UNIX stellt vordefinierte symbolische
Konstanten und**

**Datenstrukturen zur Verfügung. Sie sind in 2 Header
files**

enthalten

**sys/types.h
sys/socket.h**

**und werden über # include Anweisungen
eingebunden.**

Beispiele für symbolische Konstanten sind:

**SOCK_STREAM
SOCK_DGRAM
AF_INET**

Server Process Using Sockets

```
/* Error checking has been omitted in this example,
 * for easy reading */
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>
#include <netinet/in.h>
#define PORTNUM 1503

main()
{
    int r_sd,                /* rendezvous socket */
        c_sd;              /* communications socket */
    char buffer[256];        /* to store data */
    struct sockaddr_in name; /* socket name structure */

    /* create socket */
    r_sd = socket(AF_INET, SOCK_STREAM, 0);

    /* bind socket */
    name.sin_family = AF_INET; /* Internet domain */
    name.sin_addr.s_addr = INADDR_ANY;
    name.sin_port = PORTNUM;
    bind (r_sd, (struct sockaddr *) &name, sizeof(name));

    /* accept client connections */
    listen(r_sd,5);

    do {
        /* The 2nd and 3rd arguments of the accept call
         * are not used here, but can be used
         * to obtain the client's address. */
        c_sd = accept ( r_sd, (struct sockaddr *) 0,
            (int *) 0 );
        /* read socket and then close it */
        read(c_sd, buffer, 256);
        close(c_sd);
    } while (1);
}
/* end of main */
```

Client Process Using Stream Sockets

```
/* Error checking has been omitted to allow for easy
 * reading. However, error checks should always be
 * implemented. */
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>
#include <netinet/in.h>
#define PORTNUM 1503
#define MESSAGE "Socket Test"

main()

int sd;                /* client socket descriptor */
struct sockaddr_in
    servername;        /* server socket name */
struct hostent *host;  /* server host struct */

/* create client socket */
sd = socket(AF_INET, SOCK_STREAM, 0);

/* Obtain network address of server.
 * "myhost" is used as server name.
 */
host = gethostbyname( "myhost" );

/* Initialize server socket address */
bcopy( host->h_addr,
        (char*) &servername.sin_addr, host->h_length);
servername.sin_family = AF_INET;
servername.sin_port = PORTNUM;
/* Attempt to connect to server */
connect (sd,(struct sockaddr *)&servername,
        sizeof(servername));

/* write data to socket; close socket */
write ( sd, MESSAGE, sizeof( MESSAGE ));
close(sd);
}
```

```

.
.
.
/* create client socket */

sd = socket(AF_INET, SOCK_STREAM, 0);

/* Obtain network address of server.
 * "dummy_host" is used as server name.
 */

host = gethostbyname( "myhost" );

/* Initialize server socket address */

bcopy( host->h_addr,
(char*) &servername.sin_addr,
host->h_length);

.
.
.

```

In der Linux-Datei /etc/hosts steht die Zuordnung aller 32 Bit Internet Adressen zu den entsprechenden symbolischen Host Adressen.

Mit `gethostbyname` wird die Internet Adresse des Rechners mit dem Namen `myhost` ermittelt.

Mit `bcopy` wird diese Adresse in die Socket-Struktur kopiert.

Resolver

Routine für die Umsetzung von Host-Namen und IP-Adressen

gethostbyname (...) **IP-Adresse eines Host-Namens wird ermittelt**

gethostbyaddress (...) **Host-Name einer IP-Adresse wird ermittelt**

gethostbyname (...) durchsucht Tabelle **/etc/hosts** *) nach verschiedenen Kriterien. Liefert Struktur

struct hostent {....

Umsetzung dieser Information ist Aufgabe der Anwendung

***) Unix : /etc/hosts**
WindowsXP
c:\windows\system32\drivers\etc\hosts

Command	Description
gethostbyaddr()	This call returns the symbolic name of a host address. The call will obtain that information from a network name server, if one is present. If a name server is not present, or unable to resolve the host name, the call searches the local TCPIP\ETC\HOSTS file until a matching host address is found or an EOF marker is reached. The call places the requested information in a hostent structure.
gethostbyname()	This call returns the address of a symbolic host name. The call will obtain that information from a network name server, if one is present. If a name server is not present, or unable to resolve the host name, the call searches the TCPIP\ETC\HOSTS file until a matching host name is found or an EOF marker is reached. The call places the requested information in a hostent structure.
gethostent()	This call reads the next line of the TCPIP\ETC\HOSTS file.
sethostent()	This call opens and rewinds the TCPIP\ETC\HOSTS file.
endhostent()	This call closes the TCPIP\ETC\HOSTS file.

statt

```
listen (r_sd, 5)
```

besser

```
if (listen (r_sd, 5) == -1)  
{  
    perror ("listen");  
    exit (1);  
}
```

Socket Implementierungen

Teil des Kernels in BSD

Andere Betriebssysteme wie MS-DOS, Windows, MAC-OS beinhalten Sockets als Bibliotheken

WinSock **Multivendor API, normiert die
Verwendung von TCP/IP
unter Windows**

Schwierigkeiten mit Sockets

Sockets nur Schnittstelle zur Transportschicht. Keine Unterstützung zur

**Kommunikationssteuerung
Datendarstellung**

Anwendungsprogrammierer muß selber Protokolle definieren und implementieren um

Kommunikationspartner finden	Name Server
Austausch von Nachrichten steuern	Sync Punkte
Eigenschaften des Transport- mechanismus berücksichtigen	Verbindungsorientiert verbindungslos
Wechsel der Transferrichtung steuern auf Fehler reagieren (z.B. Verbindungsabbruch)	Please Token Waisen

und

einheitliche Datendarstellung	XDR, ASN.1
--------------------------------------	-------------------

Bei komplizierten Anforderungen aufwendige und fehleranfällige Programmierung