

# Praktikum Augmented Reality auf mobilen Geräten

WinSock-Programmierung

20. Juni 2006

Department of Informatics | Technische Universität München

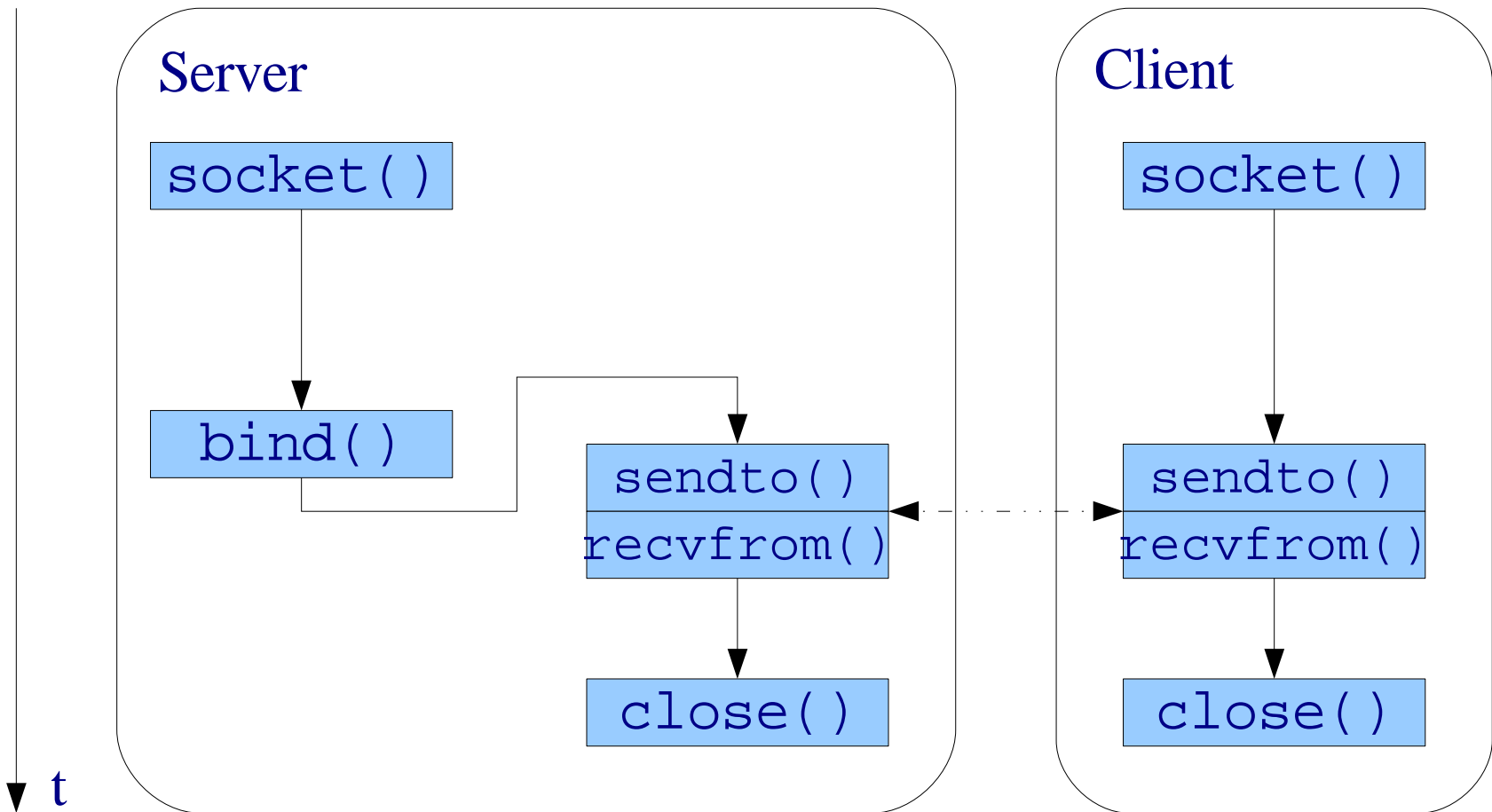
# Berkeley Sockets

- weitverbreitetes Programmiermodell
- Socket ist Start- oder Endpunkt einer Verbindung
- von Interesse: nur TCP- oder UDP-Sockets

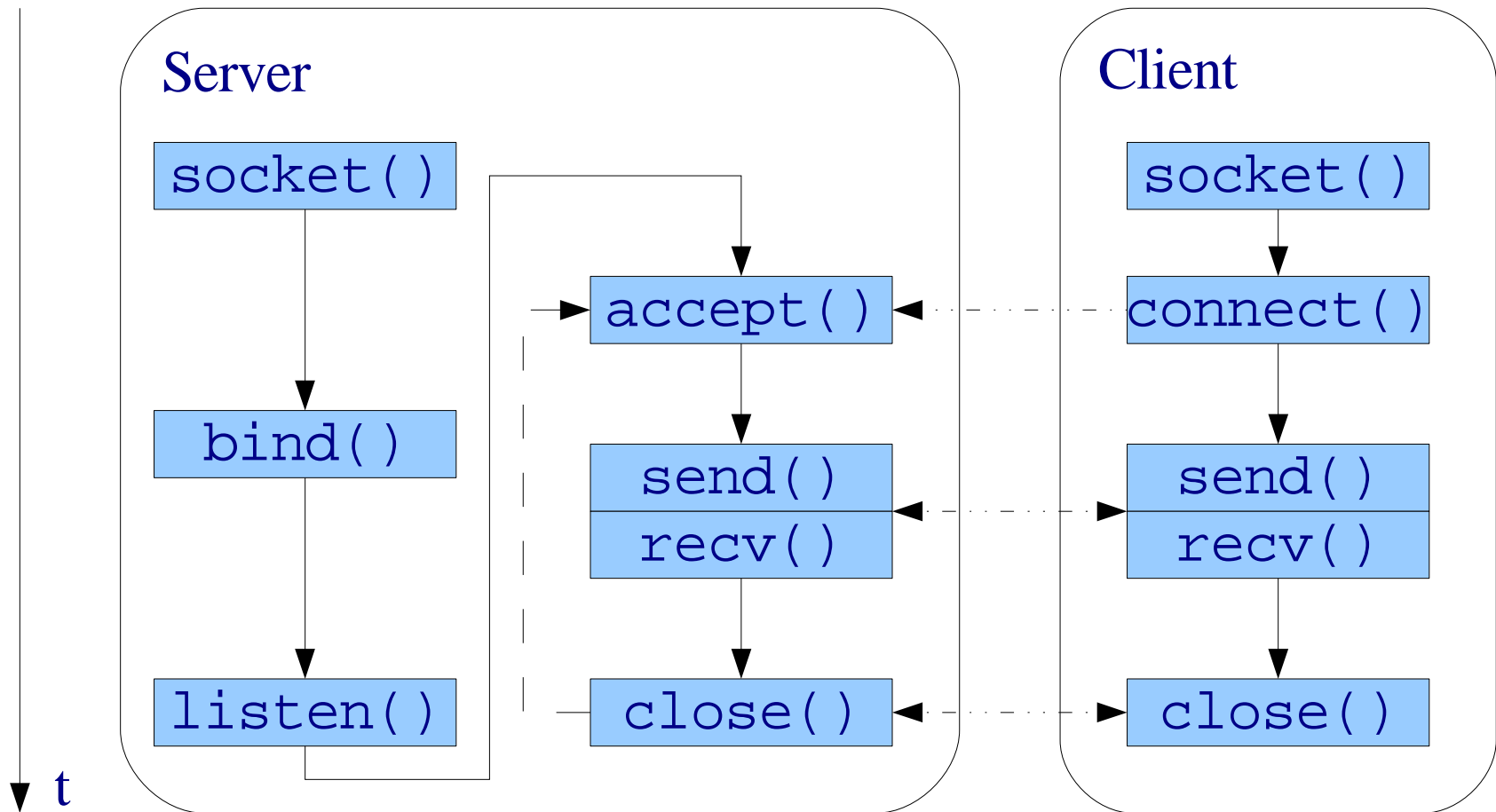
# Unterschiede TCP und UDP

- TCP
  - verbindungsorientiert
  - fehlerkorrigierend
  - automatische Fragmentierung
  - höhere Latenz
- UDP
  - verbindungslos
  - keine Garantie für Ankunft
  - geringere Latenz

# Ablauf einer UDP-Kommunikation



# Ablauf einer TCP-Kommunikation



# WinSock-Eigenheiten

- Aufruf von `WSAStartup`

```
if (WSAStartup (MAKELWORD(2,2), &WSAData) != 0)
{
    MessageBox (NULL, TEXT("WSAStartup failed!"), TEXT("Error"), MB_OK);
    return FALSE;
}
```

- anstatt `close()`: `closesocket()`
- Rückgabewert: immer 0 oder `SOCKET_ERROR`  
Fehlerwert via `WSAGetLastError`

# Weitere Anmerkungen

- Sockets arbeiten mit IP-Adressen
  - für DNS: siehe z.B. `gethostbyname()`
- Sockets sind i.A. blockierend
  - Programm bleibt stehen, solange keine Daten da
  - Problem in Verbindung mit DirectX
- Lösung: non-blocking mode
  - Verhalten umschaltbar mit `ioctlsocket()`
  - Statusabfrage mit `select()`

## Weitere Anmerkungen (2)

Problem: Verzögerungen bei vielen kleinen Paketen

- Grund: Nagle-Algorithmus kombiniert mehrere kleine Pakete
- abschaltbar via `setsockopt()`

Beispiel:

```
static const BOOL one = TRUE;
```

```
..
```

```
setsockopt( socket, IPPROTO_TCP, TCP_NODELAY,  
           &one, sizeof(BOOL) );
```

# Weitere Anmerkungen (3)

Informationen im Web: wie üblich bei MSDN und der Referenz