Softwareentwicklung 4

11. Einheit

Dominik Dolezal

Höhere Lehranstalt für Informationstechnologie

1. Dezember 2015

Inhalt

tgm
Die Schule der Technik

Einführung

tgm
Die Schule der Technik

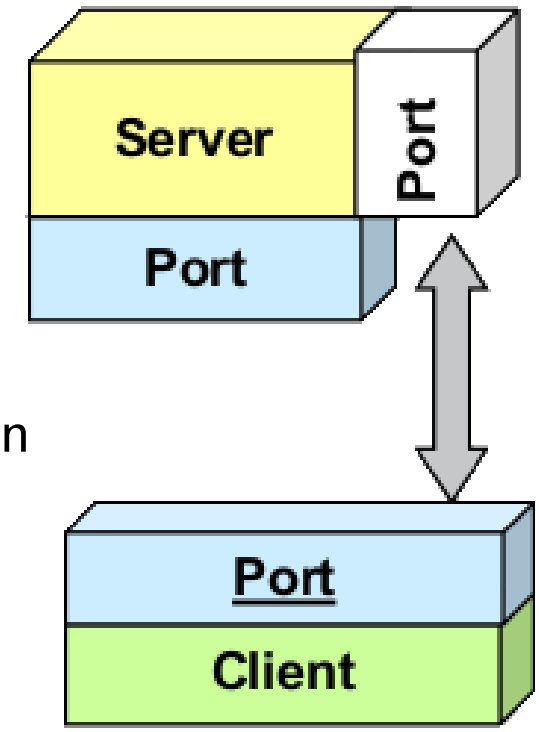
Realität	**Netzwerk**
EmpfängerIn / Name	Host / Name
Land Stadt / Ort Postleitzahl Straße / Gasse Stiege Hausnummer Stockwerk	IP-Adresse
Türnummer	Port

Einführung

tgm
Die Schule der Technik

Ein Socket...

- ... ist ein Kommunikations-endpunkt
- ... wird über IP-Adresse und Port eindeutig identifiziert
- ... nimmt eingehende Verbindungen entgegen
- ... verbindet sich mit seinem Gegenstück



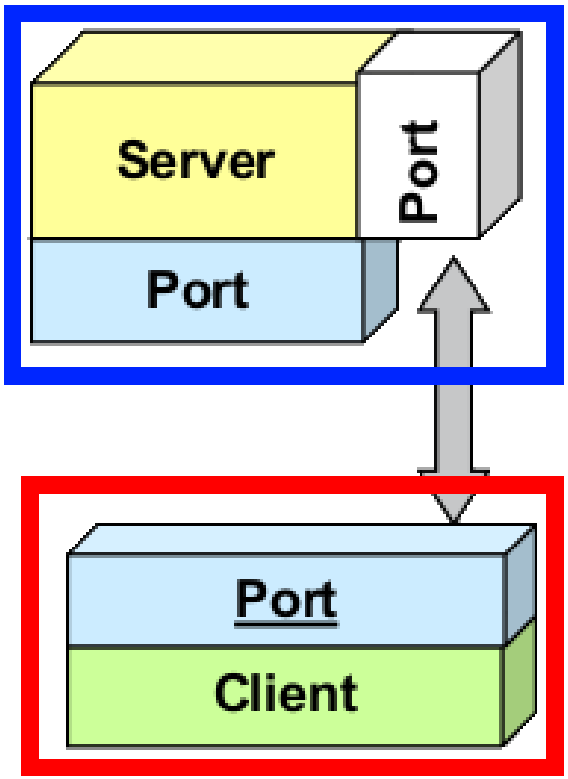
Unter einer IP-Adresse können mehrere Sockets erreichbar sein!

Server-Socket vs. Client-Socket

tgm
Die Schule der Technik

Sowohl **Server** als auch **Client** benötigen einen eindeutigen Kommunikationsendpunkt

⇒ Wir unterscheiden zwischen **Server-Socket** und **Client-Socket**

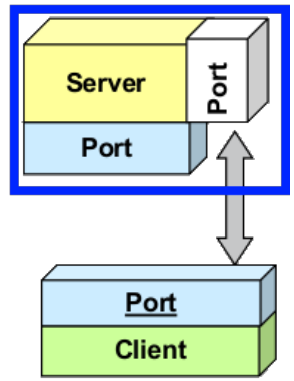


Server-Socket vs. Client-Socket

tgm
Die Schule der Technik

Server-Socket

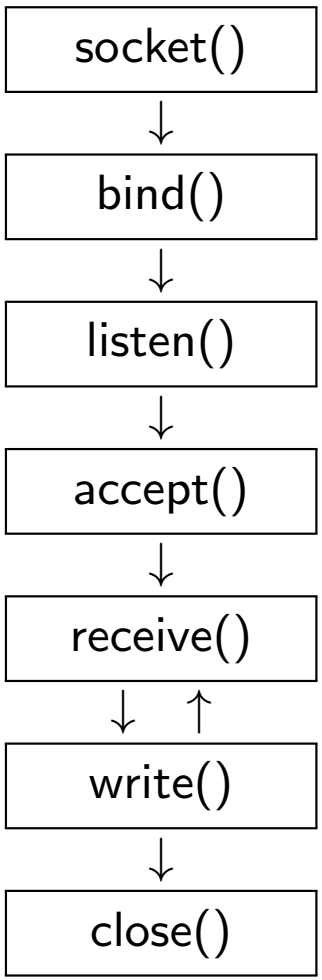
- Wartet und „horcht“ auf eingehende Verbindungen durch Clients (engl. „to listen“)
- Wird daher auch **passiver** Socket genannt
- Muss einer IP-Adresse und einem Port zugeteilt sein („Binding“)
- Empfängt Daten von Clients, führt Berechnungen durch und sendet das Ergebnis zurück
- Typischerweise Multi-Threaded – Warum?



Server-Socket vs. Client-Socket

tgm
Die Schule der Technik

Server-Socket



Socket erstellen

Adresse und Port zuweisen

Auf Verbindungen horchen

Nächste Verbindung entgegen nehmen (**blockierend**)

Nachricht empfangen (**blockierend**)

Nachricht senden (tw. **blockierend**)

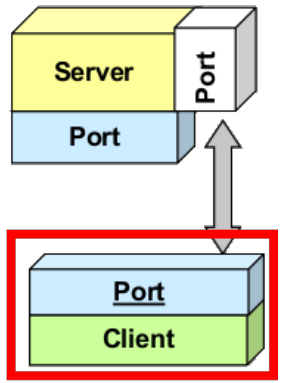
Socket schließen

Server-Socket vs. Client-Socket

tgm
Die Schule der Technik

Client-Socket

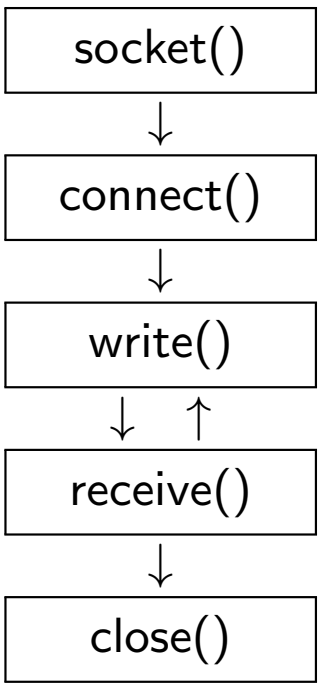
- Verbindet sich mit Server-Socket und initiiert den Verbindungsaufbau
- Wird daher auch **aktiver** Socket genannt
- Muss IP-Adresse und Port des Servers wissen
- Sendet Daten an Server und empfängt die Resultate
- Multi-Threaded oder Single-Threaded



Server-Socket vs. Client-Socket

tgm
Die Schule der Technik

Client-Socket



Socket erstellen

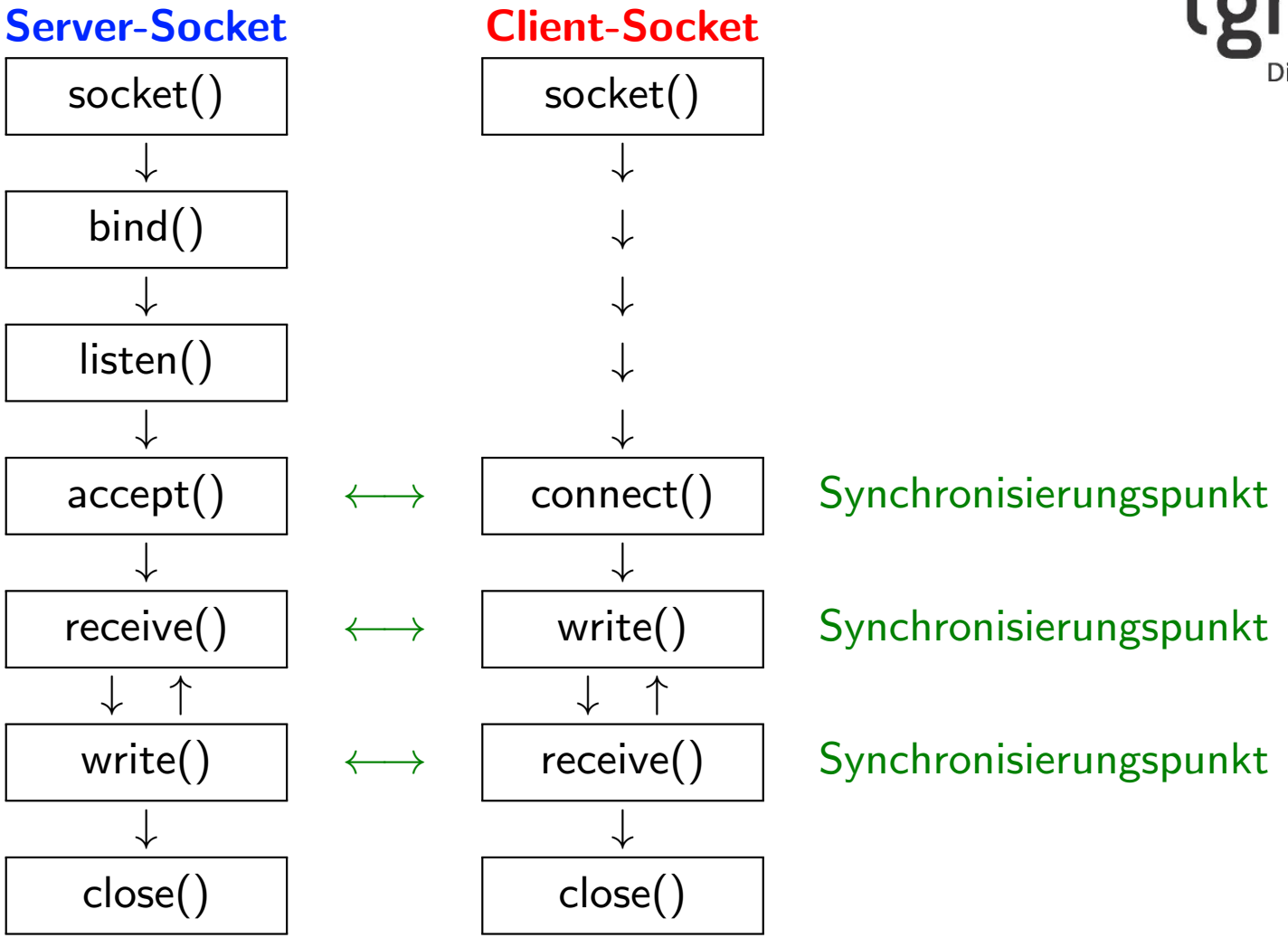
Mit Server-Socket verbinden (**blockierend**)

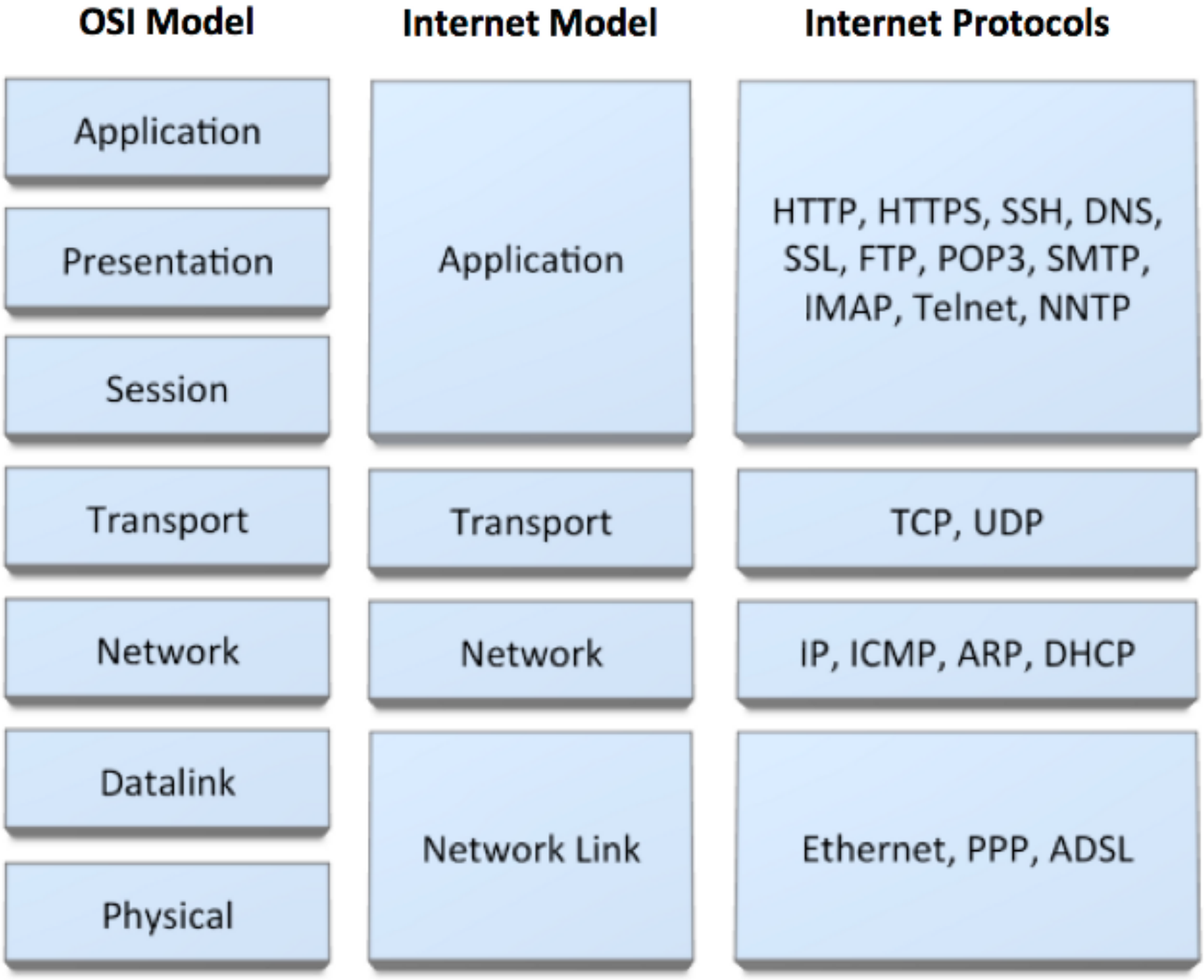
Nachricht senden (tw. **blockierend**)

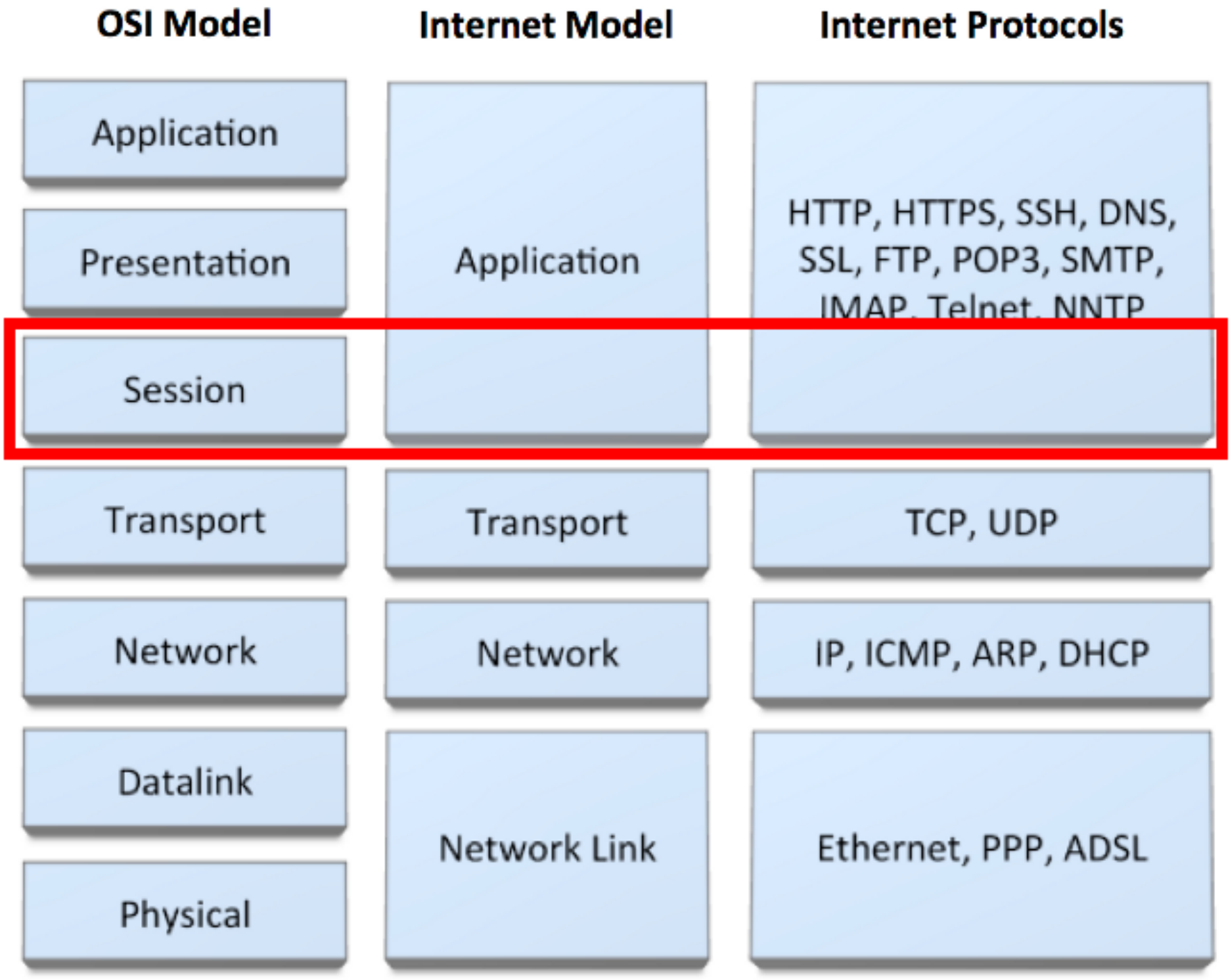
Nachricht empfangen (**blockierend**)

Socket schließen

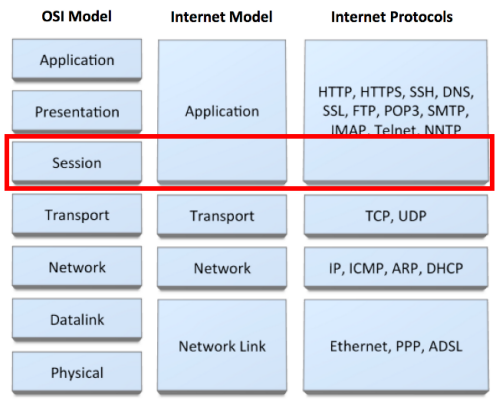
Server-Socket vs. Client-Socket




OSI Model
Internet Model
Internet Protocols
Application
Presentation
Session
Transport
Network
Datalink
Physical
Application
Transport
Network
Network Link
HTTP, HTTPS, SSH, DNS, SSL, FTP, POP3, SMTP, IMAP, Telnet, NNTP
TCP, UDP
IP, ICMP, ARP, DHCP
Ethernet, PPP, ADSL


OSI Model
Internet Model
Internet Protocols
Application
Presentation
Session
Transport
Network
Datalink
Physical
Application
Network Link
HTTP, HTTPS, SSH, DNS,
SSL, FTP, POP3, SMTP,
IMAP, Telnet, NNTP
TCP, UDP
IP, ICMP, ARP, DHCP
Ethernet, PPP, ADSL

Protokoll

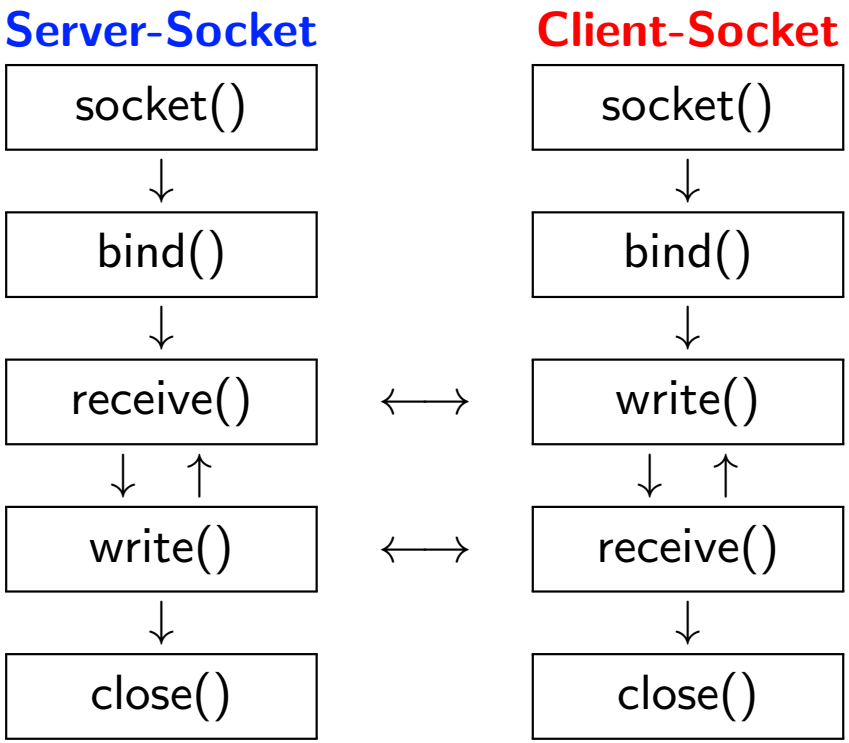


- Sockets setzen auf der vierten Schicht auf und befinden sich selbst auf Layer 5
- Je nach verwendetem Protokoll auf Schicht 4 unterscheiden wir zwischen
  - TCP-Sockets („Stream“)
  - UDP-Sockets („Datagram“)

- Das zugrunde liegende Protokoll wirkt sich maßgeblich auf die Programmierung von Sockets aus
- Bisher wurden TCP-Sockets besprochen (accept, connect, ...)
- Bei UDP-Sockets vereinfacht sich das Verfahren

UDP-Sockets

tgm
Die Schule der Technik



Anwendungsgebiete

tgm
Die Schule der Technik

TCP-Sockets	UDP-Sockets

Anwendungsgebiete

tgm
Die Schule der Technik

TCP-Sockets	UDP-Sockets
Geschäftsanwendungen	Audio-Streaming
WWW	Video-Streaming
Webservices	Computerspiele
Login	Alive-Messages
Datenbankanwendungen	Live-Daten (Aktien, Wetter, ...)

- Grundsätzlich wird meist TCP verwendet, außer die Übertragung muss nicht sichergestellt werden

Beispiel: Server

tgm
Die Schule der Technik

```
int main()
{
  try
  {
    io_service io_service;
    // Fuer neue Verbindungen und Bindung
    tcp::acceptor acceptor(io_service,
        tcp::endpoint(tcp::v4(), 1234));
    while (true)
    {
      tcp::socket socket(io_service);
      cout << "Warte auf Client..." << endl;
      // Auf neuen Client warten (blockiert)
      acceptor.accept(socket);
```

Beispiel: Server

tgm
Die Schule der Technik

```
        string message = "Hallo Welt!";
        boost::system::error_code ignored_error;
        // Nachricht senden
        write(socket, boost::asio::buffer(message),
            ignored_error);
        cout << "Verbindung zu Client geschlossen" <<
            endl;
      } // Verbindung beenden
    }
    catch (exception& e)
    {
      cerr << e.what() << endl;
    }
    return EXIT_SUCCESS;
}
```

Beispiel: Client

tgm
Die Schule der Technik

```
int main(int argc, char* argv[])
{
  try
  {
    io_service io_service;
    tcp::socket socket(io_service);
    // Mit Server-Socket verbinden
    socket.connect(tcp::endpoint(ip::address::from_string("
        1234));
    // Server hat Verbindung angenommen
    while(true)
    {
      // Zwischenspeicher fuer Server-Antwort
      array<char, 128> buf;
      boost::system::error_code error;
```

Beispiel: Client


```
        // Daten von Socket einlesen
        size_t len = socket.read_some(buffer(buf),
            error);
        if (error == error::eof)
          break;// Server hat Verbindung beendet
        else if (error)
          throw boost::system::system_error(error);
        cout << "Server: ";
        cout.write(buf.data(), len);
        cout << endl;
      }
    }
    catch (exception& e)
    {
      cerr << e.what() << endl;
    }
    return EXIT_SUCCESS;
}
```

Übung

tgm
Die Schule der Technik

Erweitere das Programm!

- Lade das Programm herunter und bringe es zum Laufen
- Erweitere das Programm, sodass der Client beliebige Textnachrichten über die Konsole einliest
- Der Server gibt die übertragenen Textnachrichten in der Konsole aus
- Über „`exit`“ schließt der Server die Verbindung
- Den Code findest du auf GitHub

Zusammenfassung

tgm
Die Schule der Technik

- Sockets sind Kommunikationsendpunkte
- Ein Socket wird über IP-Adresse und Port identifiziert
- Wir unterscheiden zwischen Server-Socket und Client-Socket
- Sockets setzen auf Schicht 4 des OSI-Modells auf
- Es gibt TCP- und UDP-Sockets
- TCP- und UDP-Sockets haben unterschiedliche Anwendungsgebiete