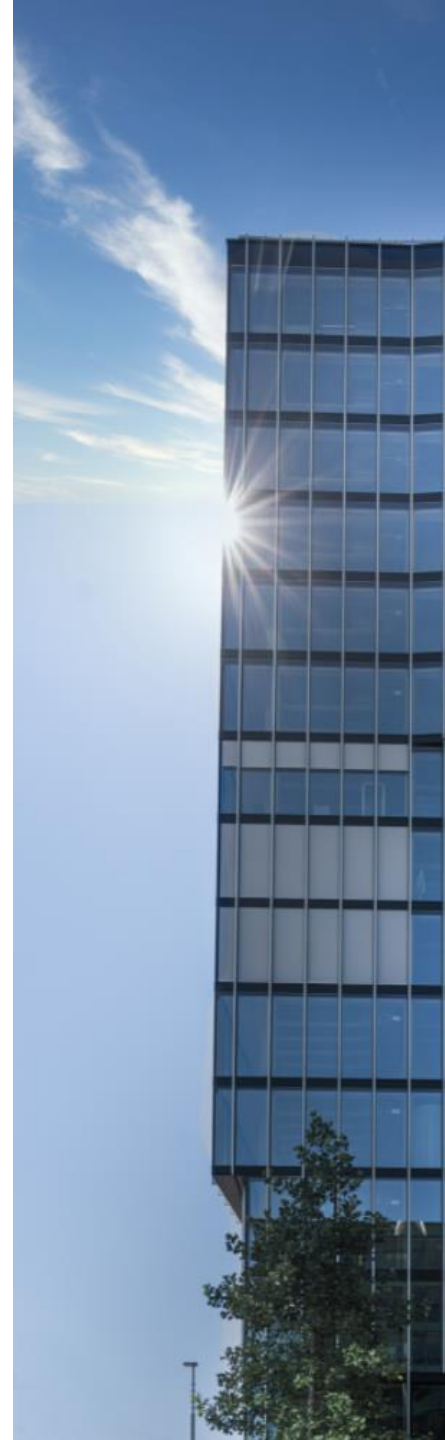


C# in Action

Socket-Kommunikation

Einführung

Roger Diehl



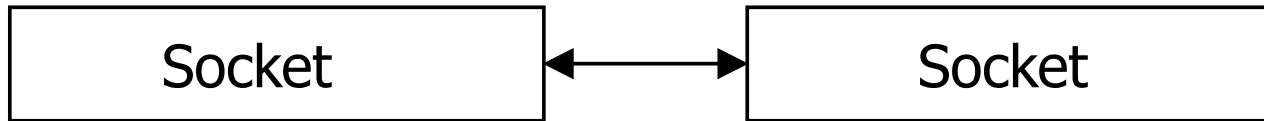
Hinweis

Für die Modulendprüfung sind die Inhalte der Unterlagen relevant.
Diese Folien sind eine Zusammenfassung der Unterlagen:

C# in Action, Teil 2 – Parallele und Verteilte Programmierung

Socket Prinzip

- Die Socket-Kommunikation ist eine Interprozesskommunikation
- Sie ist eine Ende-zu-Ende Kommunikation



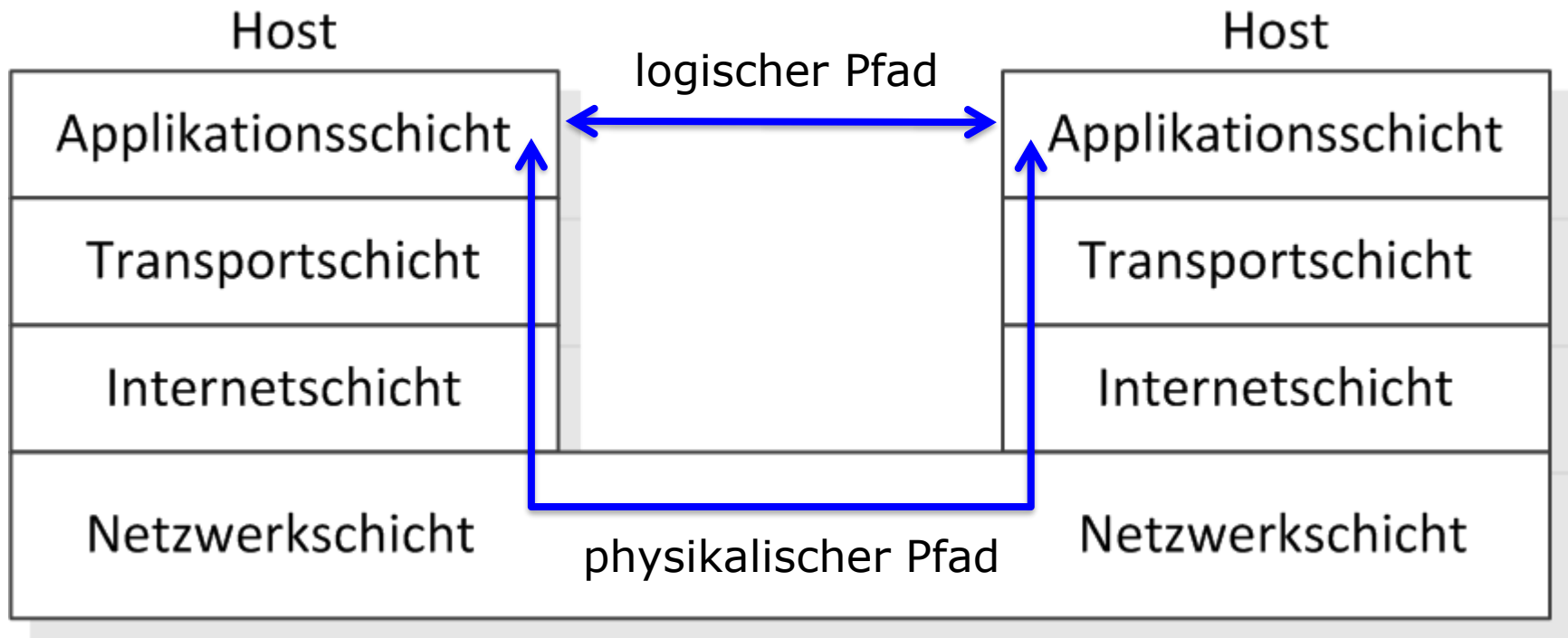
- Sockets stellen eine Abstraktion eines Datenendpunktes dar
 - Der Benutzer braucht sich nicht um technische Details der Datenübertragung zu kümmern.

Was kann ein Socket?

1. Verbindung zu einem entfernten Prozess aufbauen
2. Daten senden
3. Daten empfangen
4. Verbindung beenden
5. Einen Port (Applikation) binden
6. An einem Port auf Verbindungswunsch hören
7. Verbindungswunsch am Port akzeptieren

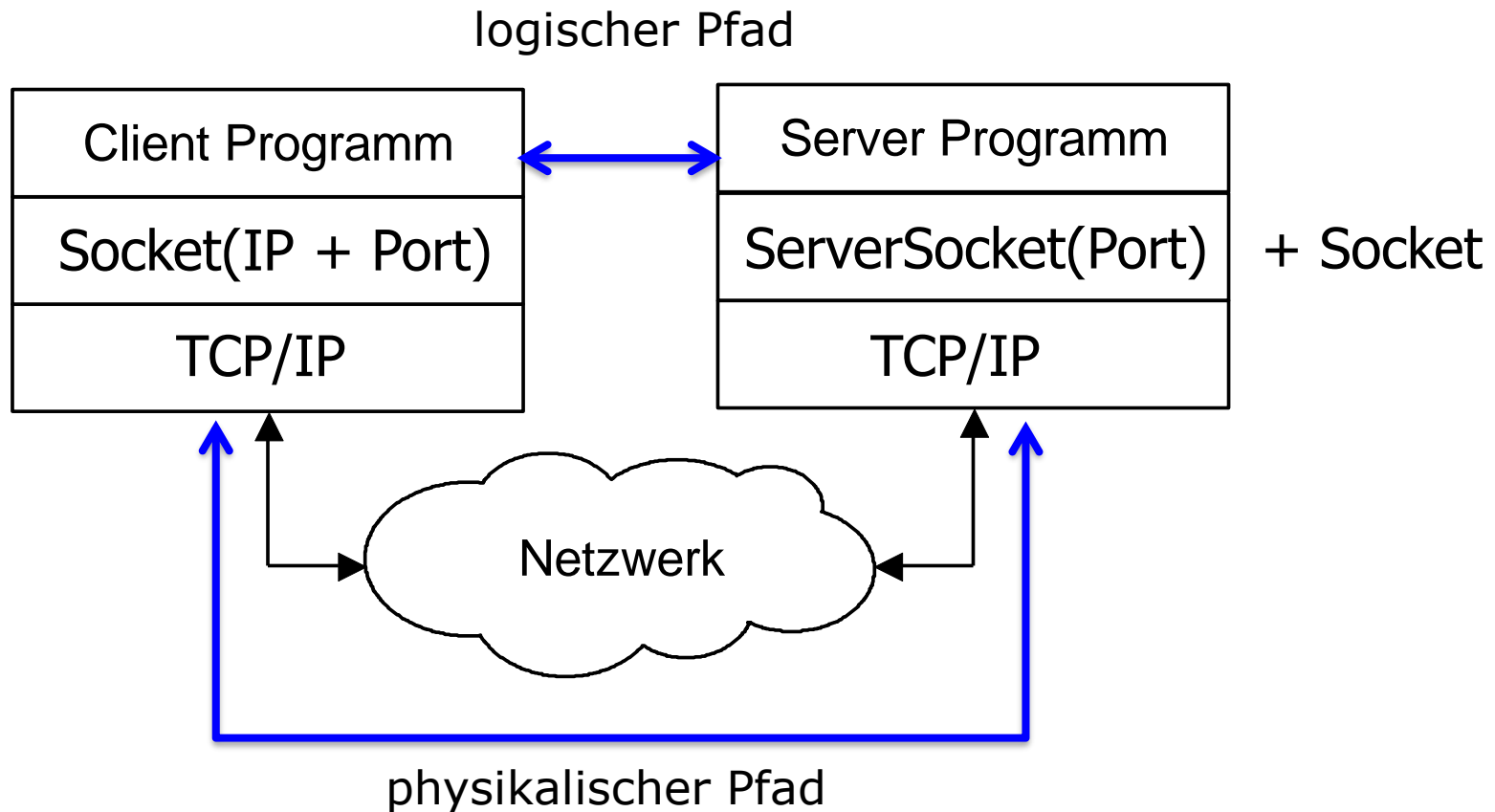
Kommunikation

- Physikalisch gehen die Daten der Host-zu-Host Kommunikation durch alle Schichten.
- Logisch gehen die Daten von Applikation zu Applikation. Die Kommunikationsdetails sind für die Applikation transparent.



Sockets für Client und Server

- Wir werden Socket Kommunikation für TCP-Client und TCP-Server betreiben



Erzeugen eines Sockets

TcpClient kapselt Socket mit TCP Protokoll



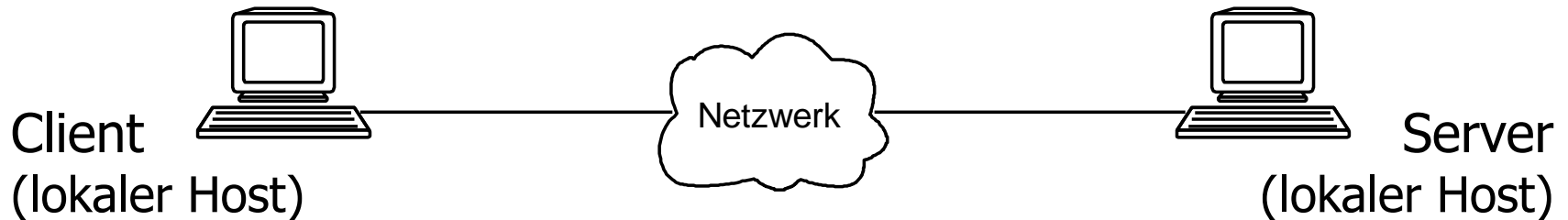
```
TcpClient theClient;  
try {  
    theClient = new TcpClient(host,port);  
}  
catch (ArgumentOutOfRangeException) {  
    // Port ausserhalb des erlaubten Bereichs  
}  
catch (SocketException) {  
    // Fehler beim Zugriff auf den Socket  
    // siehe: SocketException.ErrorCode  
}
```

Socket Eigenschaften

■ Informationen über die Socket-Verbindung

```
try {  
    theClient = new TcpClient(host,port);  
    Socket theSocket = theClient.Client;  
    //...  
    theSocket.LocalEndPoint;  
    theSocket.RemoteEndPoint;  
    theSocket.ProtocolType;  
    //...  
    theClient.Close();  
}  
catch (Exception e) {  
    // Fehler...  
}
```

Was stellen Sie fest, wenn Sie dieses Programm testen?




Server Socket


- Ein typisches Server Muster

```
try {  
    TcpListener listen = new TcpListener (port);  
    listen.Start();  
  
    while (...) {  
        TcpClient client = listen.AcceptTcpClient();  
        //...Kommunikation mit Client  
        client.Close();  
    }  
}  
catch (Exception e) {  
    // Fehler  
}
```


TcpListener (ServerSocket)
erstellen und starten



In einer Iteration auf
Verbindungswunsch warten

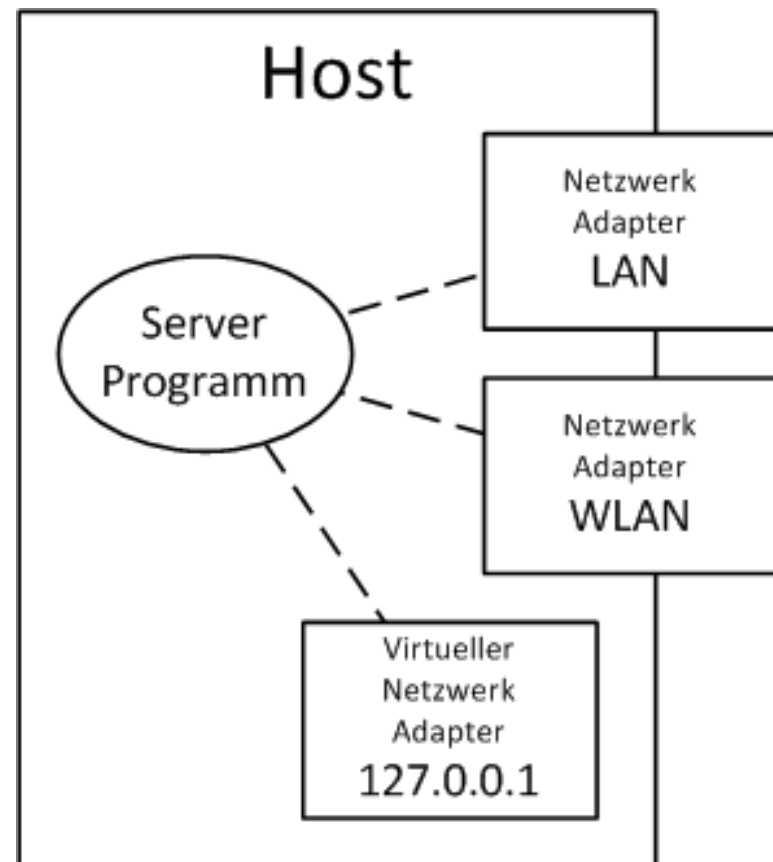


Wenn Verbindungswunsch
akzeptiert, dann
Kommunikation mit Client
durchführen



Netzwerk Adapter NIC (network interface card)

- Heutige Rechner besitzen mehr als ein Netzwerk Adapter für den Anschluss an ein Computernetzwerk, z.B. für:
 - LAN (Local Area Network)
 - WLAN (Wireless LAN)
 - Virtuelles Netzwerk
- Ein Server muss wissen, auf welchem Netzwerk Adapter er auf Verbindungen warten soll.
- Beim Erstellen des Server Sockets oder Datagramm Sockets kann (muss) der Netzwerk Adapter angegeben werden.



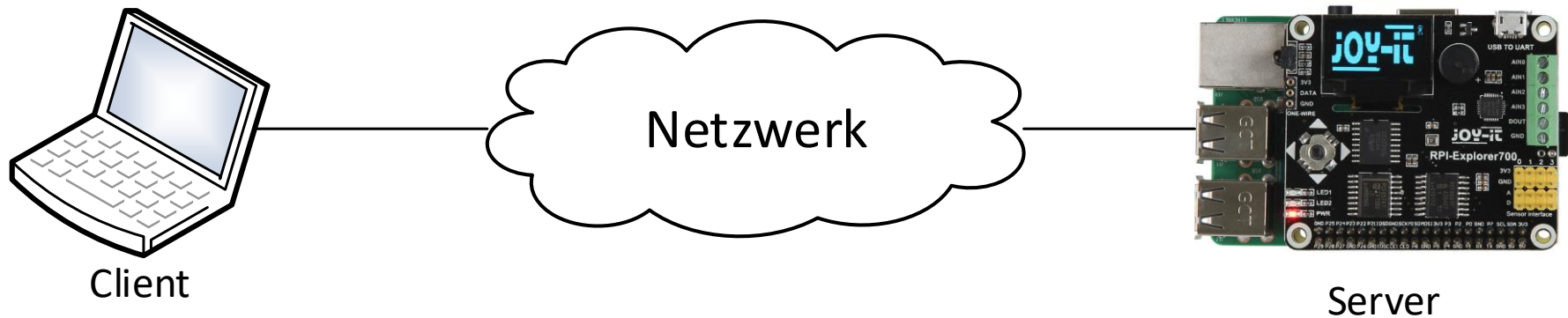
Simpler Day Time Server

- Der Day Time Server nimmt auf Port 13 einen Verbindungswunsch entgegen und sendet dem Client einen String mit Datum und Zeit.

```
public class SimpleDayTimeServer {  
    TcpListener (ServerSocket)  
        erstellen und starten  
  
    public static void Main() {  
        TcpListener listen = new TcpListener(13);  
        listen.Start();  
        while (true) {  
            auf Verbindungswunsch warten  
            TcpClient client = listen.AcceptTcpClient();  
            TextWriter tw =  
                Kommunikation  
                (Stream) zum  
                Client erstellen  
                → new StreamWriter(client.GetStream());  
            tw.Write(DateTime.Now.ToString());  
            tw.Flush();  
            client.Close();  
            Senden von Datum  
            und Zeit zum Client  
            Übergibt die Zeichen  
            an den Socket  
        }  
    }  
}
```

Übung: Socket Implementationen

Der Raspberry Pi mit dem Explorer 700 dient als Day Time Server. Auf dem Laptop oder PC läuft das telnet Programm, womit wir die Zeit auf dem Raspberry Pi abfragen.



- **Aufgabe A:** Betrieb mit dem SimpleDayTimeServer
- **Aufgabe B:** Buzzer ertönt bei korrektem Verbindungsaufbau
- **Aufgabe C:** Client (telnet) gibt einen Integer Wert mit, der die Beep Dauer des Buzzers steuert

Fragen?