.NET Networking 1

Proseminar Objektorientiertes Programmieren mit .NET und C#

Matthias Jaros

Institut für Informatik
Software & Systems Engineering

Agenda

- Motivation
- Protokolle
- Sockets
- Anwendung in .NET C#
  - Codebeispiele
  - Hilfsklassen
- Demo
- Höhere API's
  - .NET Remoting
  - MSMQ

Wozu überhaupt verteilte Anwendungen?

- Nicht verteilte Anwendungen im Allgemeinen sicherer und performanter

Vorteile von verteilten Anwendungen:

- Nutzer können Dienste und Anwendungen in Anspruch nehmen unabhängig vom aktuellen Standort.
- Gemeinsame Nutzung von Ressourcen
  - ➔ Netzwerkdrucker
  - ➔ Dateiserver, Datenbank

- Bestes Beispiel: Internet (www,email,...)

Aufgabenverteilung im Netzwerk

Client/Server-Modell

- Server (engl: to serve) bietet Dienste an
- Client nimmt Dienste in Anspruch

Peer-to-Peer

- Programm ist Server und Client zugleich.

Internet Protocol (IP)

- Erste vom Übertragungsmedium unabhängige Schicht
- Dient der Adressierung von Computern in Netzwerken
- IP Version 4 sehr verbreitetes Protokoll
- Nachfolger ist IP Version 6 da größerer Adressbereich
- TCP wie UDP basieren auf IP
- Lokaler Computer hat immer 127.0.0.1

Transmission Control Protocol

Vorteile
- Verbindungsorientiert
- Reihenfolge der Pakete garantiert
- Drei-Wege-Handschlag
- Flusssteuerung

Nachteile
- Viel Overhead
- Langsam

User Datagram Protocol

Vorteile
- Minimal
- Schnell
- Weniger Overhead

Nachteile
- Keine Sequenznummern
- Verbindungslos
- Keine Flusssteuerung

Ports

- Nötig um eine bestimmte Anwendung ansprechen zu können
- Im TCP/UDP Header stehen Quell- und Zielport

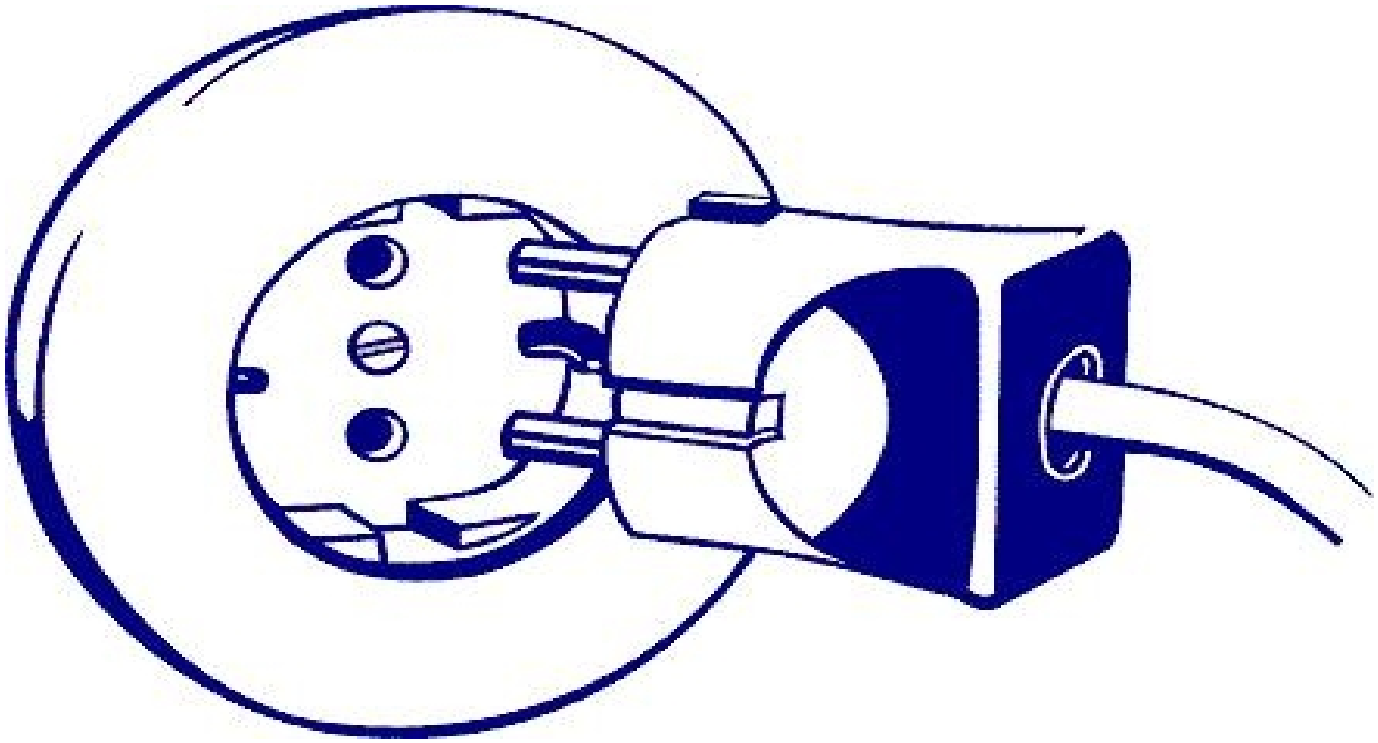
- Drei Unterscheidbare Bereiche
  - ➔ System Ports 0-1023 durch IANA standardisiert
  - ➔ User Ports 1024-49151
  - ➔ Dynamic Ports 49152-65535

Anwendungsgebiete

TCP
- FTP 21
- SFTP 22
- HTTP 80
- Telnet 23
- SSH 22
- SMTP 25

UDP
- Allgemein VoIP
  - RTP
  - RTSP 554
- Domain Name System 53
- Youtube

Sockets



Bild(3)

Sockets

- Mehrere unterscheidbare Arten
  - Raw-Sockets
  - Stream-Sockets
  - Datagram-Sockets

- Entwickelt von der Berkeley Universität
- Schnittstelle zwischen Programm und Rechnernetz
- Bidirektionale Kommunikation möglich

Nützliches

IPAddress

- Speichert IP eines Computers in einem Netzwerk
- Hat intern loopback und broadcast IPv4/v6 Adresse gespeichert

IPEndPoint

- Speichert IP und Port des lokalen und gegebenenfalls des Servers ab

Codebeispiele in C# .NET

- Socket als TCP Server Beispiel

```
IPEndPoint lokaleIP = new IPEndPoint(IPAddress.Any, port);
Socket tcpserver = new Socket(AddressFamily.InterNetwork,
SocketType.Stream, ProtocolType.Tcp);
tcpserver.Bind(lokaleIP);
tcpserver.Listen();
```

- Socket als TCP Client Beispiel

```
IPEndPoint ziel_adresse = new IPEndPoint(ip, port);
Socket tcpsocket =
   new Socket(ziel_adresse.AddressFamily, SocketType.Stream, ProtocolType.Tcp);
tcpsocket.Connect(ziel_adresse);
//Daten senden
tcpsocket.Send(bytesSent, bytesSent.Length, 0);
//Daten emfangen
bytes = s.Receive(bytesReceived, bytesReceived.Length, 0);
```

Codebeispiele in C# .NET

- UDP Sender Socket

```
Socket udpsocket = new Socket (AddressFamily.InterNetwork, SocketType.Dgram,
                               ProtocolType.Udp);
IPAddress ziel_IP = IPAddress.Parse ("127.0.0.1");
IPEndPoint ziel_adresse = new IPEndPoint (ziel_IP, 12345);
udpsocket.SendTo (sende_puffer, ziel_adresse);
```

Hilfsklassen

- Kapseln Details über TCP/UDP Verbindungen
- Bieten einfachere und benutzerfreundlichere Schnittstelle

```
TcpClient client = new TcpClient();
client.Connect(IPAddress.Parse("127.0.0.1"), 8000);
```

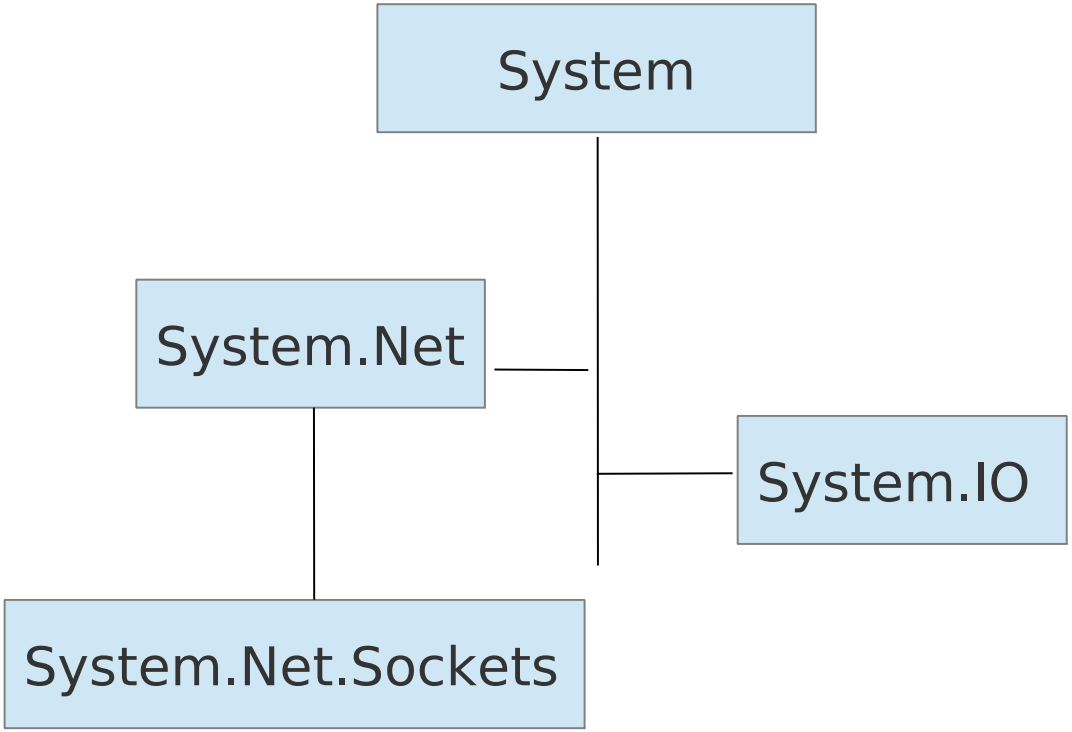
- UdpClient Hilfsklasse als Server und Client nutzbar

Hilfsklassen

```
TcpListener listener = new TcpListener(IPAddress.Parse("127.0.0.1"),
    8000);
listener.Start();
TcpClient client = listener.AcceptTcpClient();
//.... daten austauschen
client.Close();
listener.Close();
```

```
UdpClient listener = new UdpClient (listenPort);
IPEndPoint sender_group = new IPEndPoint (IPAddress.Any, listenPort);
receive_byte_array = listener.Receive(ref sender_group)
```

Bibliotheken



DEMO

.NET Remoting

- Framework für Kommunikation
- Früher Distributed COM zuständig für Netzwerkkommunikation
- Erlaubt Erstellung von Objekten über ein Rechnernetz
- Objekte können vom Server oder Client erstellt werden
- Wurde größtenteils von WCF abgelöst

Microsoft Message Queuing

- Kurz MSMQ
- Warteschlangen System
- Wird für Kommunikation zwischen Anwendungen benutzt
- Kommunikation selbst dann möglich, wenn eine Anwendung offline
  - ➔ Zwischenspeichern der Daten
- Benutzt TCP und UDP Verbindungen
- Benutzer muss sich nicht mit niedrigen API's (TCP,UDP) befassen

Referenzen

- Jay Hilyard & Stephen Teilhet, 'C# Kochbuch', 2006
- http://msdn.microsoft.com
- Bilder
  - Bild(1): http://www.tcp-ip-info.de/glossar/glossar_t.htm
  - Bild(2): http://upload.wikimedia.org/wikipedia/commons/9/98/Tcp-handshake.svg
  - Bild(3): http://alischirasi.blogsport.de/2009/07/18/iran-demo-mit-der-steckdose/