Universität Bremen

# Praktische Informatik 1
# Netzwerkkommunikation


## Thomas Röfer

Cyber-Physical Systems
Deutsches Forschungszentrum für Künstliche Intelligenz

Multisensorische Interaktive Systeme
Fachbereich 3, Universität Bremen



Bild: ©Michael Ihle

Universität Bremen

# Netzwerkkommunikation: OSI-Schichtenmodell

| OSI-Schicht | | Einordnung | DoD-Schicht | Einordnung | Protokollbeispiel | Einheiten |
|---|---|---|---|---|---|---|
| 7 | Anwendungen (Application) | Anwendungs-orientiert | Anwendung | Ende zu Ende (Multihop) | HTTP<br>FTP<br>HTTPS<br>SMTP<br>LDAP<br>NCP | Daten |
| 6 | Darstellung (Presentation) | | | | | |
| 5 | Sitzung (Session) | | | | | |
| 4 | Transport (Transport) | Transport-orientiert | Transport | | TCP<br>UDP<br>SCTP<br>SPX | TCP = Segmente<br>UDP = Datagramme |
| 3 | Vermittlung (Network) | | Vermittlung | Punkt zu Punkt | ICMP<br>IGMP<br>IP<br>IPsec<br>IPX | Pakete |
| 2 | Sicherung (Data Link) | | Netzzugriff | | Ethernet<br>Token Ring<br>FDDI<br>ARCNET | Rahmen (Frames) |
| 1 | Bitübertragung (Physical) | | | | | Bits |

©Wikipedia

Universität Bremen

# Internet Protokoll (IP): IP-Adresse

- Adressierung in einem Netzwerk und zwischen Netzwerken
- 32 Bit lang (**IPv4**) oder 128 Bit lang (**IPv6**)
- **Schreibweise IPv4**: Bytes als Dezimalzahlen mit Punkten getrennt
  - Z.B. **134.102.224.17**, eigener Rechner: **127.0.0.1**
  - Besteht aus **Netzwerk-ID** und **Host-ID** und muss eindeutig sein
- **Schreibweise IPv6**: 16-Bit-Hexadezimalzahlen mit Doppelpunkten getrennt (Ein Bereich von Nullen kann durch **::** ersetzt werden)
  - Z.B. **2001:638:708:30e0::17**, eigener Rechner: **::1**

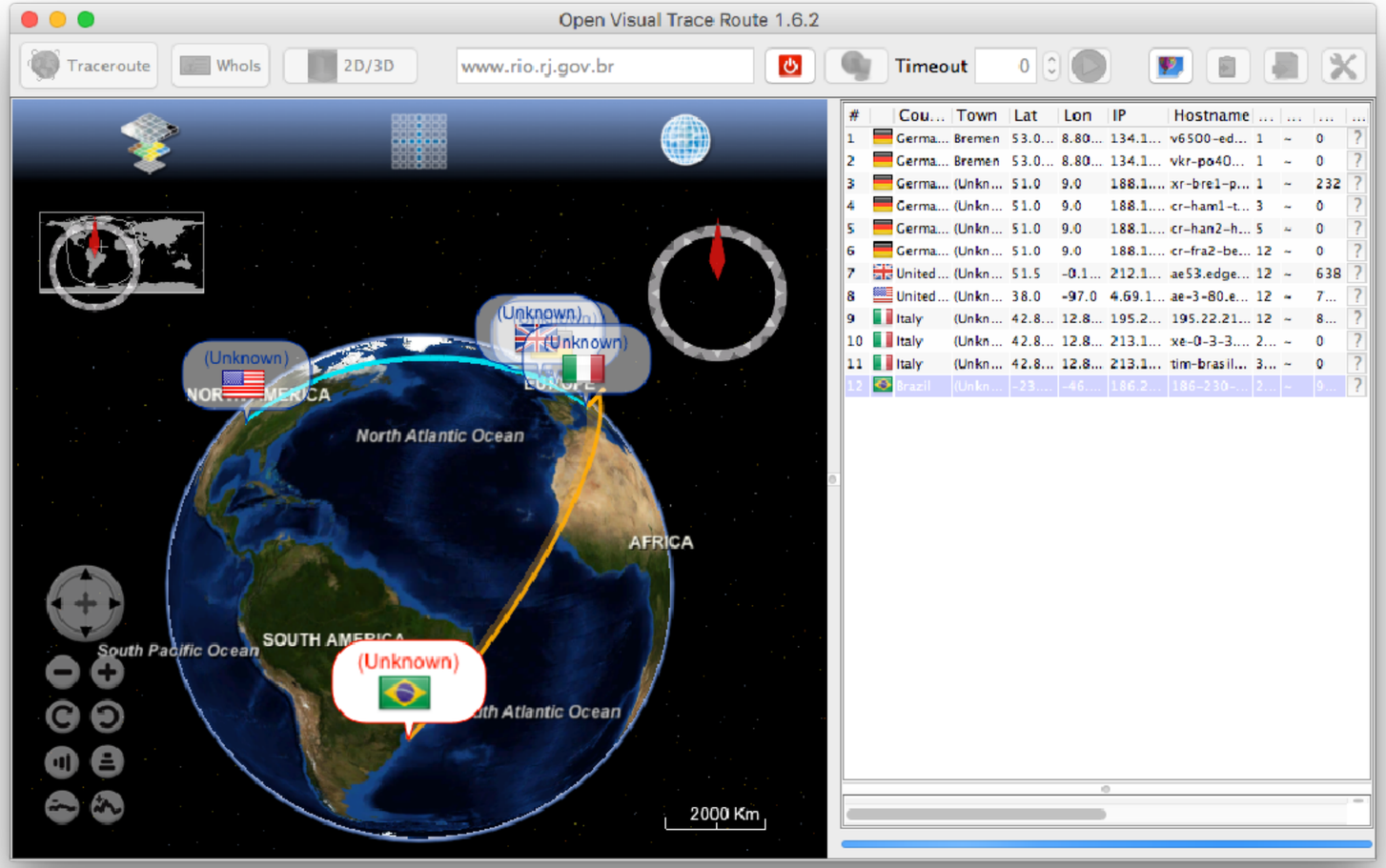
# Internet Protokoll (IP): IP-Routing

- Verbindungsaufbau zwischen verschiedenen Netzwerken
- Informationsaustausch mit Hilfe von **Routern**
- Ein Router ist in mehreren Netzen enthalten
- Ein Datenpaket wird zuerst an den Router übermittelt
- Kann Router Zielcomputer erreichen?
  - Ja: Paket wird direkt übermittelt
  - Nein: Nachschauen in Routing-Tabelle

©Wikipedia

Universität Bremen

# Trace Route: Demo



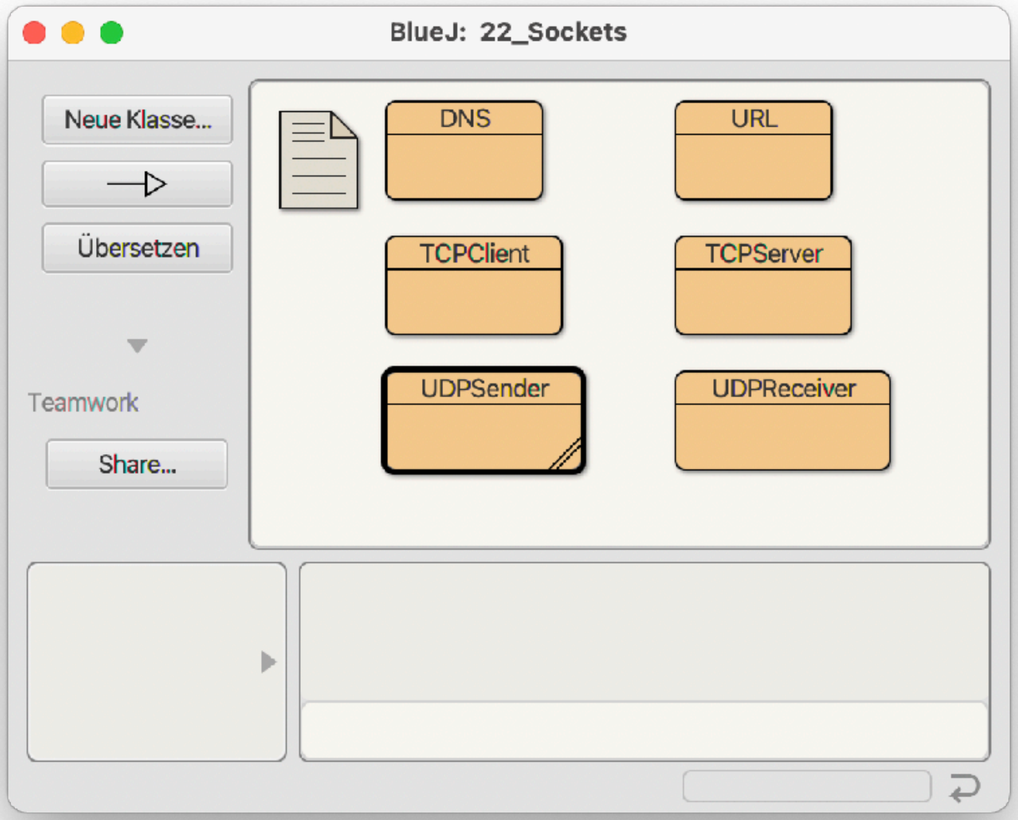
Universität Bremen

# Domain Name System (DNS)

- IP-Adressen sind schlecht zu merken
- Zuordnung von IP-Adressen zu DNS-Namen
- Ein DNS-Name besteht aus Computernamen (Host) und Domäne
  - Z.B. **x22.informatik.uni-bremen.de**

    Host Domäne
  - Eigener Rechner: **localhost**

# nslookup: Demo



Universität Bremen

# Domain Name System (DNS) in Java verwenden

- Namen auflösen

```
InetAddress ia = InetAddress.getByName("localhost");
String name = ia.getHostName(); // "localhost"
String address = ia.getHostAddress(); // "127.0.0.1"
```

  - Geht auch mit IP-Adressen

- **UnknownHostException**

  - Wenn Name falsch

  - oder Netzwerk nicht erreichbar

Universität Bremen

# Client-Server-Verbindungsschema

- Ein Rechner stellt einen bestimmten Dienst zur Verfügung (**Server**)
  - Wartet passiv, bis eine Verbindung zu ihm aufgebaut wird
- Ein anderer Computer möchte Dienst nutzen (**Client**)
  - Baut Verbindung zum Server auf
- Unterscheidung zwischen Client und Server schwierig, da Server auch wieder Client sein kann
- Ein Server soll mehrere Dienste anbieten können: **Ports**

Universität Bremen

# Ports

- Jeder Dienst auf dem Server läuft auf einem anderen Port
- Port-Nummern sind Ganzzahlen
  - In Gruppen **System** und **Benutzer** eingeteilt
  - System-Ports: 0-1023
  - Benutzer-Ports: 1024-65535
- Port-Nummern sind beliebig festlegbar
  - Inoffizielle Standards: SSH-Port 22, HTTPS-Port 443 usw.

Universität Bremen

# Transportprotokolle

- User Datagram Protocol (**UDP**)
  - Verbindungslos, nur einzelne Datenpakete werden verschickt
  - Keine Überprüfung, ob Pakete fehlerfrei angekommen sind (weniger Nutzlast)
  - Unterstützt Versand an mehrere Empfänger (**Broadcast**, **Multicasting**)
- Transmission Control Protocol (**TCP**)
  - Kommunikationskanal (point-to-point)
  - Daten können in beide Richtungen übertragen werden
  - Verbindungsorientiert und zuverlässig

Universität Bremen

# Sockets

- Kommunikationsschnittstelle zwischen zwei Programmen zum Austausch von Daten über das Netzwerk
  - Applikation fordert einen **Socket** vom Betriebssystem an
  - Anschließend können Daten über den Socket verschickt und empfangen werden
- Betriebssystem verwaltet Sockets und Verbindungsinformationen
  - **Datagramm-Sockets**: Senden einzelner Nachrichten
  - **Stream-Sockets**: Kontinuierlicher Zeichen-Datenstrom

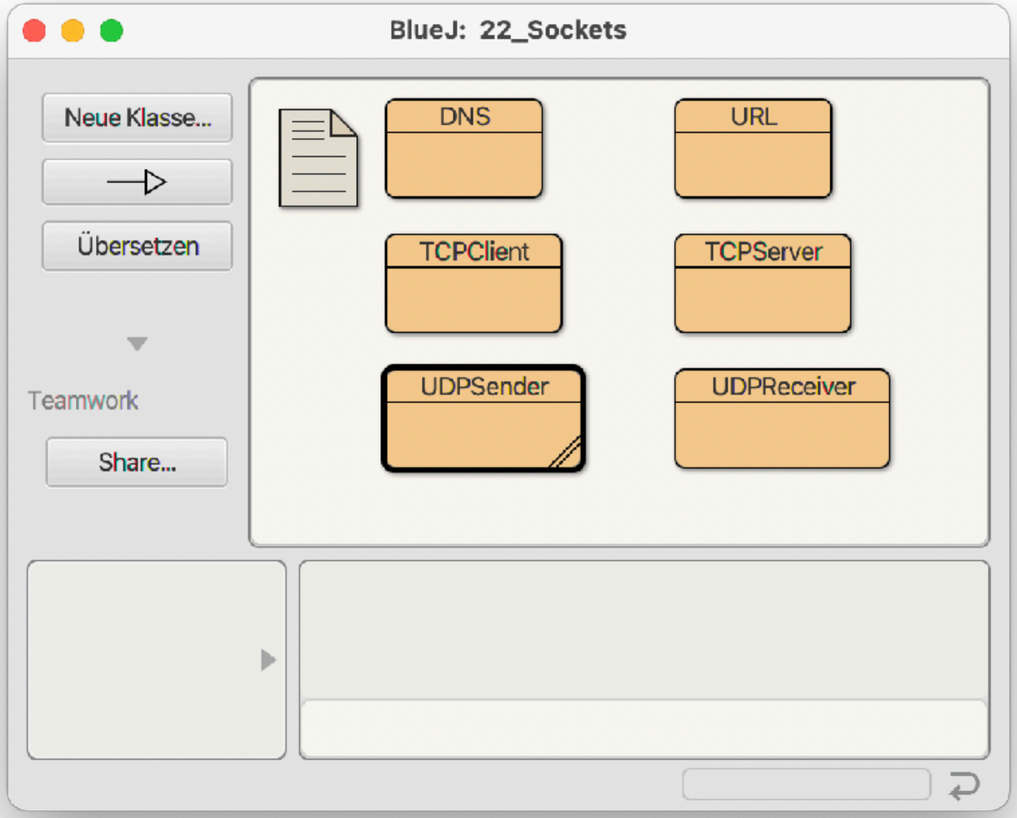
Universität Bremen

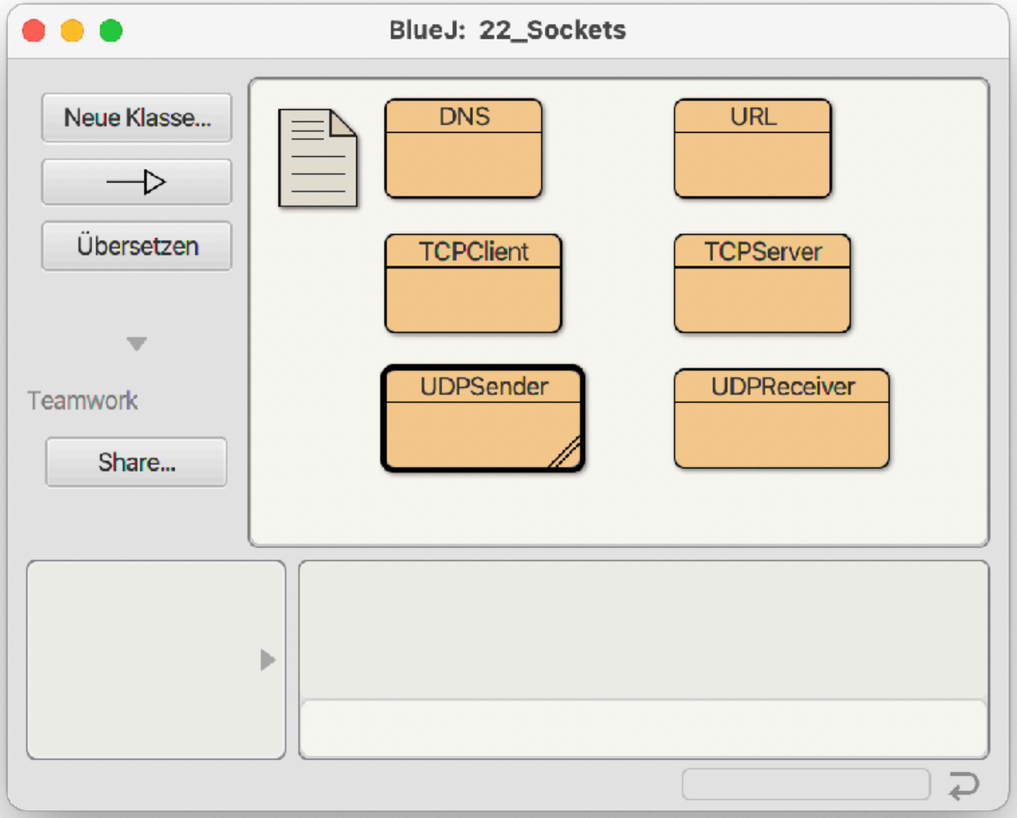
# User Datagram Protocol (UDP)

- Verschickt Daten als einfache Pakete
- Zustellung und Einhaltung der Reihenfolge der Pakete nicht garantiert
- Neben den Nutzdaten werden noch Verwaltungsinformationen gesendet
- In Java
  - Versenden/empfangen über **DatagramSocket**s
  - Pakete werden als Objekte des Typs **DatagramPacket** repräsentiert

| Quell-Port | Ziel-Port |
|---|---|
| Länge | Prüfsumme |
| Daten | |

# UDP: Demo


BlueJ: 22_Sockets
Neue Klasse...
Übersetzen
Teamwork
Share...
DNS
URL
TCPClient
TCPServer
UDPSender
UDPReceiver


BlueJ: 22_Sockets
Neue Klasse...
Übersetzen
Teamwork
Share...
DNS
URL
TCPClient
TCPServer
UDPSender
UDPReceiver

Universität Bremen

# UDP: Pakete

```
byte[ ] data = … // Daten
DatagramPacket packet = new DatagramPacket(data,
        data.length, InetAddress.getByName(address), port);
```

- Für Senden
  - Paket erzeugen mit Zieladresse und Ziel-Port
  - Broadcast: Enthält **address** 255er (z.B. **134.102.204.255**), wird das Paket in das gesamte Subnetz verbreitet (wenn Router mitspielt, **255.255.255.255** sollte eigentlich immer gehen)
- Für Empfang: Paket mit ausreichender Größe erzeugen (überschüssige Daten gehen sonst verloren)

```
DatagramPacket packet = new DatagramPacket(
        new byte[length], length);
```

Universität Bremen

# UDP: Senden und Empfangen

- Socket erzeugen und Paket versenden

```
DatagramSocket socket = new DatagramSocket();
socket.send(packet);
```

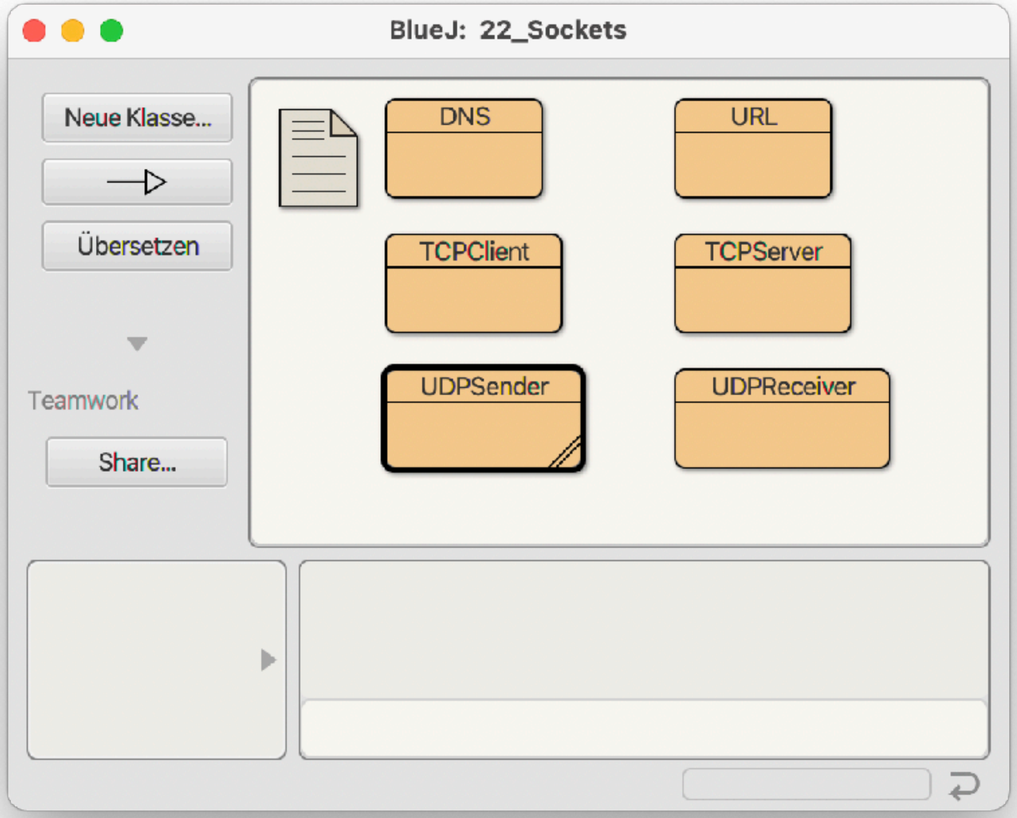
- Socket erzeugen und Paket empfangen

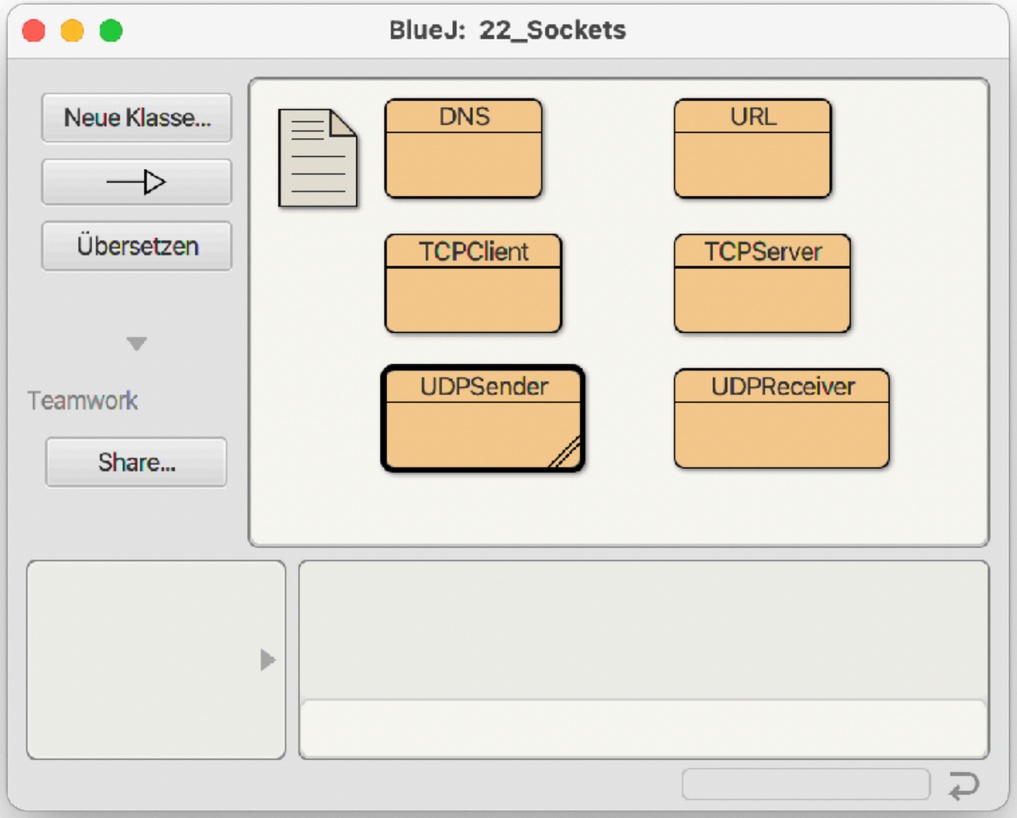
```
DatagramSocket socket = new DatagramSocket(port);
socket.receive(packet);
```

  - Ergebnis in **packet.getData()** und **packet.getLength()**
  - Port bleibt belegt, bis er geschlossen wird, also **socket.close()**

Universität Bremen

# TCP: Demo



Universität Bremen

# TCP-Client

- Socket zu IP-Adresse und Port des Servers

```
Socket socket = new Socket();
socket.connect(new InetSocketAddress(address, port));
```

- Kein Server am Port → **ConnectException**

- Sockets bestehen aus zwei Datenströmen

```
OutputStream out = socket.getOutputStream();
InputStream in = socket.getInputStream();
```

- Eingabe-Datenstrom blockiert, wenn keine Daten verfügbar sind

- **out.flush()** stellt sicher, dass Daten wirklich abgesendet werden

Universität Bremen

# TCP-Server

- Port für Dienst reservieren

```
ServerSocket server = new ServerSocket();
server.bind(new InetSocketAddress(port));
```

  - Ist der Port bereits belegt → **BindException**
  - Fehlende Berechtigung, einen Server zu betreiben (Port-Nummern von 0 bis 1023) → **SecurityException**

- Verbindungen akzeptieren

```
Socket socket = server.accept();
```

  - Auf Verbindung warten und Socket zurückliefern
  - Socket kann genauso wie beim Client verwendet werden
  - Wie mehrere Verbindungen gleichzeitig? → Multithreading (später…)

# Schließen von Sockets

- Schließen eines Client-Sockets signalisiert der Gegenseite das Ende der Verbindung

```
socket.close();
```

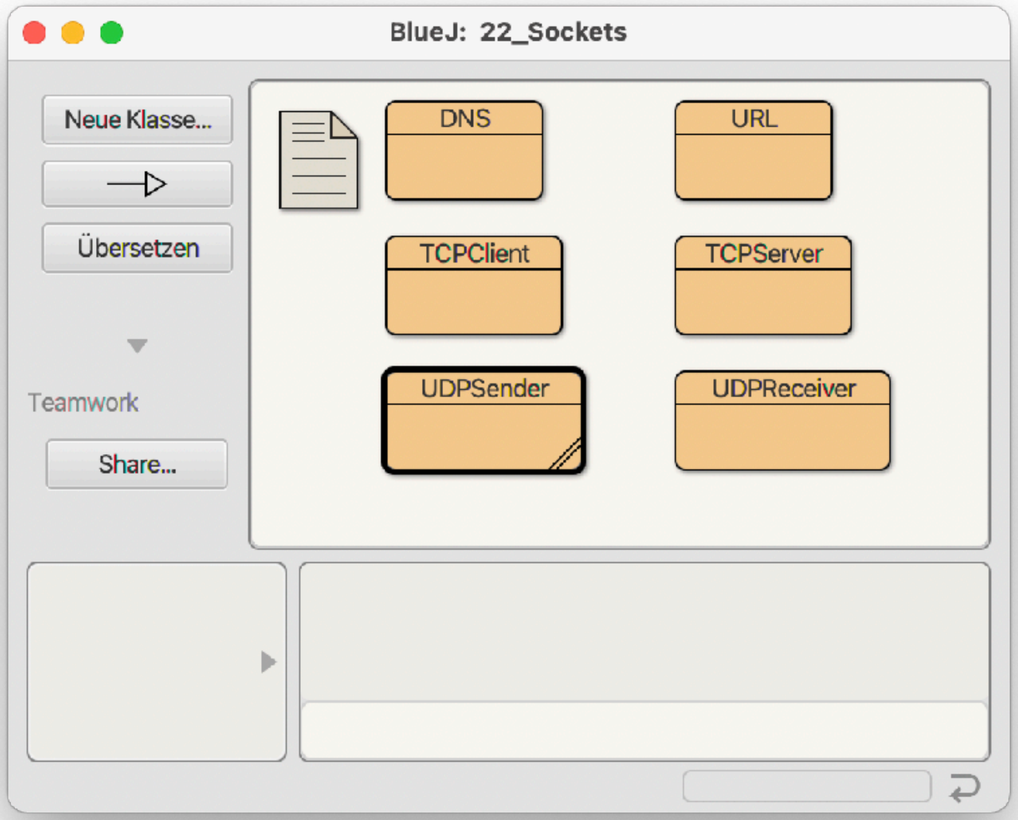
- Schließen eines Server-Sockets gibt den Port für erneute Verwendung frei
  - Ansonsten ist er blockiert

```
server.close();
```

- Passiert beim Abbauen des Objekts automatisch, aber Zeitpunkt ist zufällig (Garbage Collection)
- Beenden der JVM schließt alle Sockets

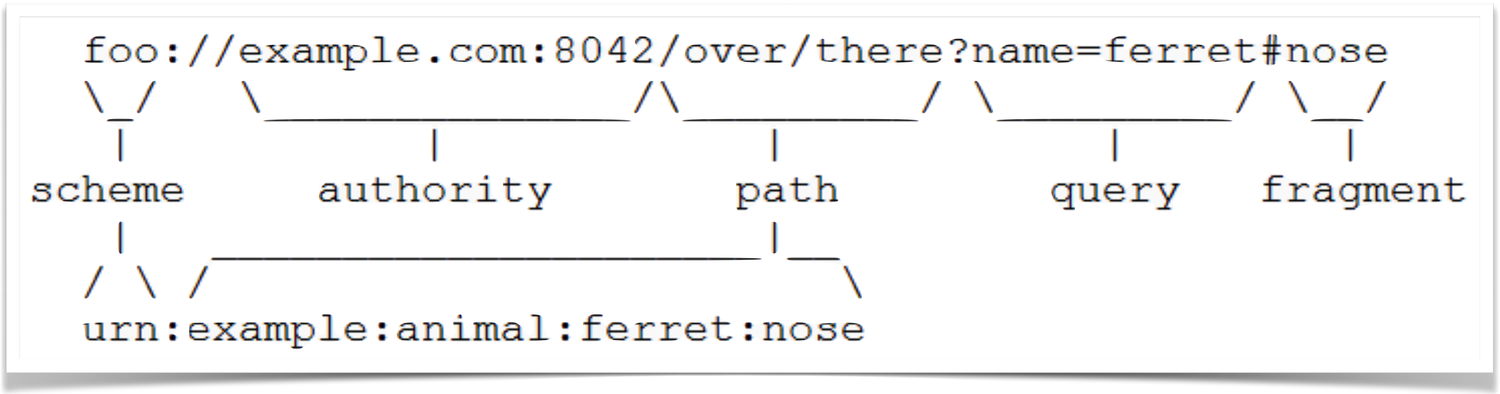
Universität Bremen

# URL: Demo



# URI, URL und URN

- **Uniform Resource Identifier** (URI): Identifikator für eine abstrakte oder physische Ressource
- **Uniform Resource Locator** (URL): URI, die den Ort einer Ressource im Netz angibt (z.B. **https:…**)
- **Uniform Resource Name** (URN): URI, die eine Ressource anhand eines Namens bezeichnet (z.B. **urn:isbn**)



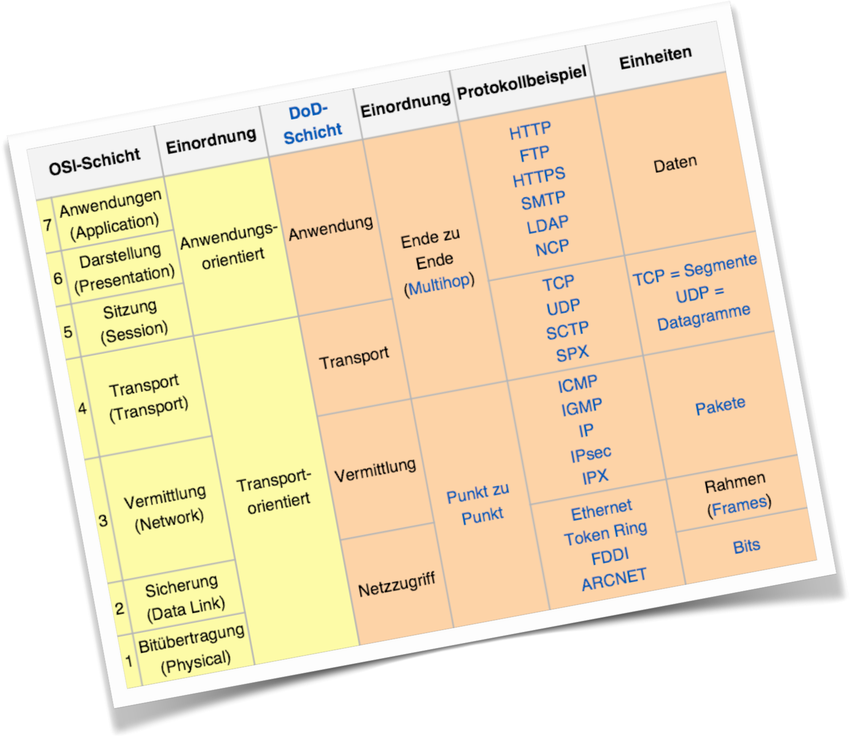
Universität Bremen

# URL und URLConnection

- **URL**-Objekte repräsentieren eine URL und besitzen Methoden, um Teile davon abzufragen
  - **getProtocol()**, **getHost()**, **getPort()**, **getPath()** …
  - Die Methode **openConnection()** liefert eine **URLConnection** zurück
- **URLConnection** repräsentiert eine Verbindung zum Server
  - Die Methoden **getInputStream()** und **getOutputStream()** liefern Datenströme zum Lesen und Schreiben
- **URL**-Objekte besitzen auch die Methode **openStream()**, die eine Abkürzung für **openConnection().getInputStream()** ist

# Zusammenfassung der Konzepte

- **IP-Adresse** und **Port**
- **Domain Name System**
- **UDP**
- **TCP-Server** und **TCP-Client**
- **URI** und **URL**



| OSI-Schicht | | Einordnung | DoD-Schicht | Einordnung | Protokollbeispiel | Einheiten |
|---|---|---|---|---|---|---|
| 7 | Anwendungen (Application) | Anwendungs-orientiert | Anwendung | Ende zu Ende (Multihop) | HTTP<br>FTP<br>HTTPS<br>SMTP<br>LDAP<br>NCP | Daten |
| 6 | Darstellung (Presentation) | | | | | |
| 5 | Sitzung (Session) | | | | TCP<br>UDP<br>SCTP<br>SPX | TCP = Segmente<br>UDP = Datagramme |
| 4 | Transport (Transport) | Transport-orientiert | Transport | | | |
| 3 | Vermittlung (Network) | | Vermittlung | Punkt zu Punkt | ICMP<br>IGMP<br>IP<br>IPsec<br>IPX | Pakete |
| 2 | Sicherung (Data Link) | | Netzzugriff | | Ethernet<br>Token Ring<br>FDDI<br>ARCNET | Rahmen (Frames) |
| 1 | Bitübertragung (Physical) | | | | | Bits |