Netzwerkprogrammierung & Threads

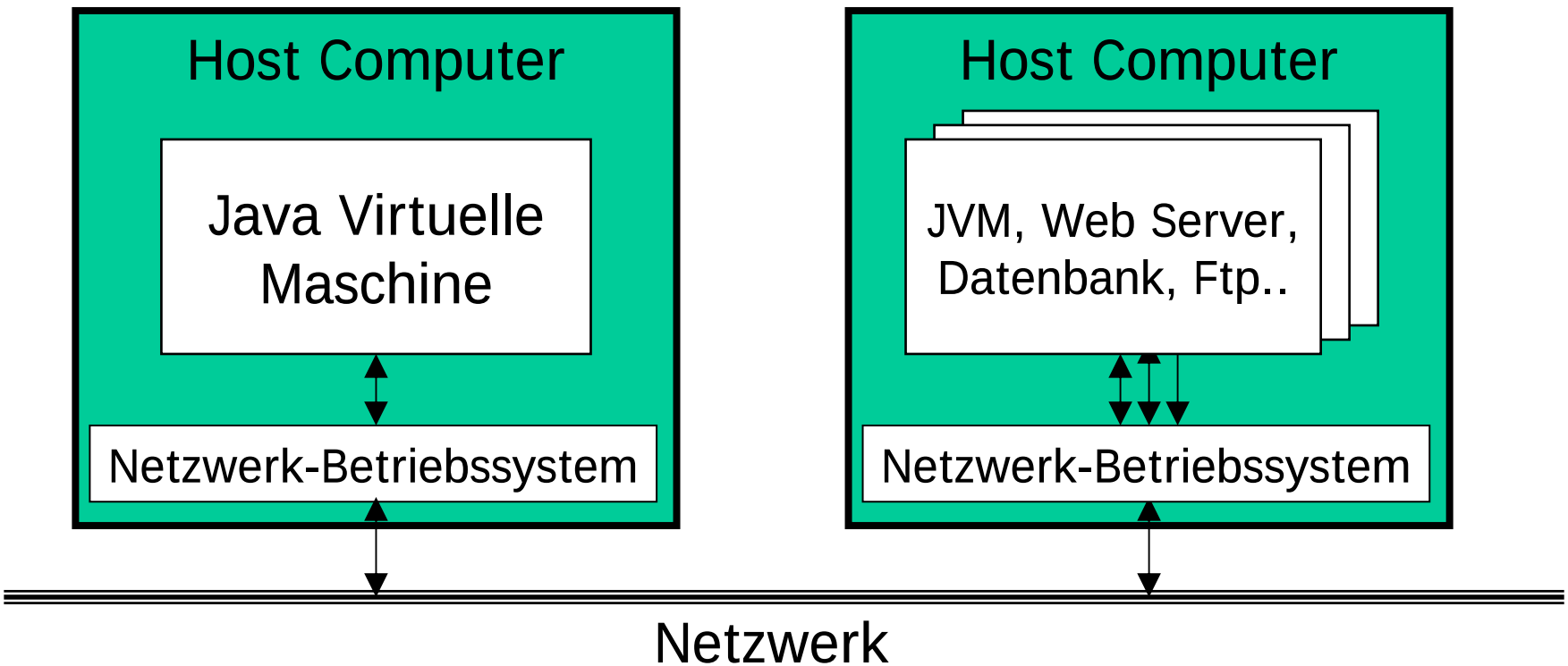
Praktikum aus Softwareentwicklung 2

Netzwerkprogrammierung & Threads

- Netzwerkprogrammierung
  - URL, URLConnection, UDP, TCP
- Threads
  - Parallele Programme, Synchronisation, ...

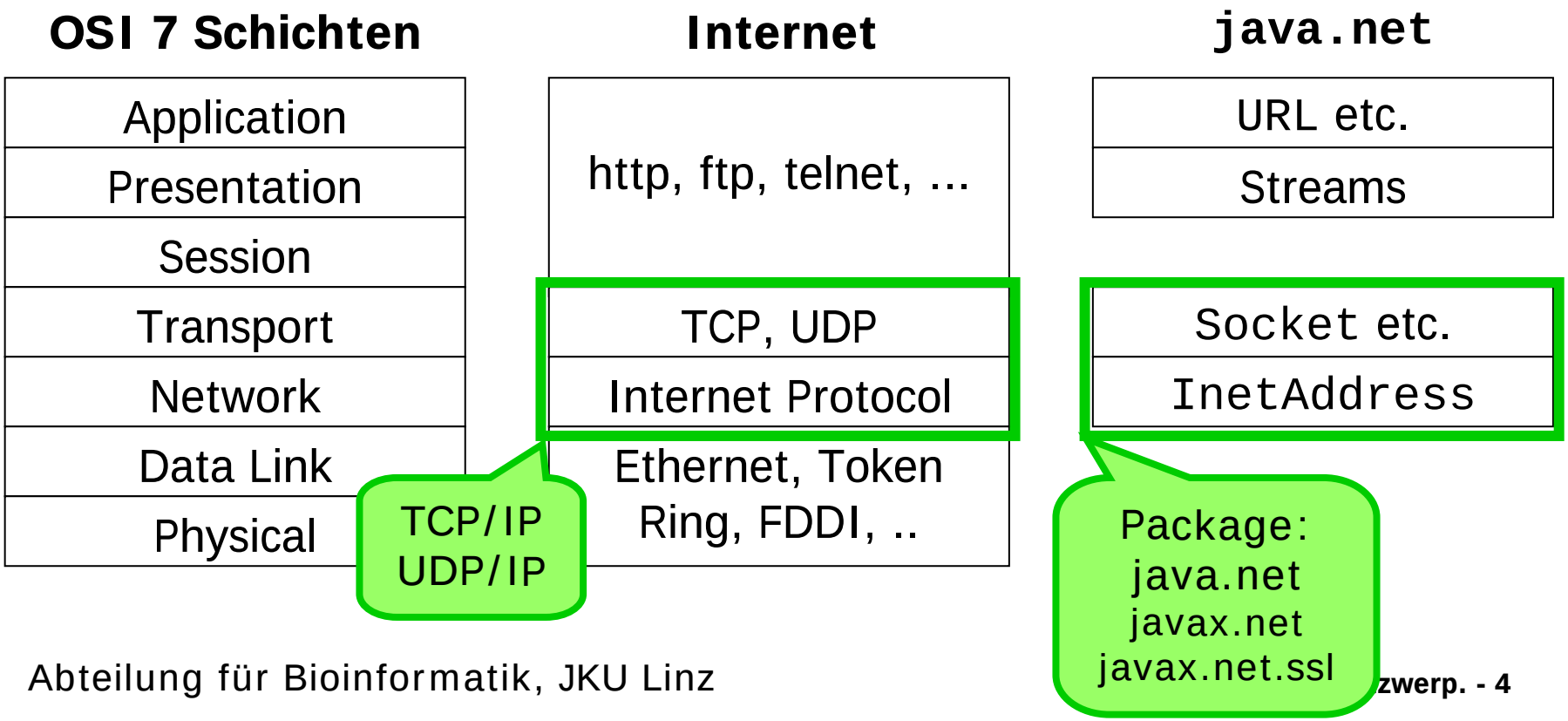
Netzwerkprogrammierung
Grundlagen (1/2)

- Kommunikation zwischen verteilten Programmen



Netzwerkprogrammierung
Grundlagen (2/2)

- ISO OSI Referenzmodell, Internet-Protokolle und Java



Netzwerkprogrammierung
HTTP und URLs

- Hypertext Transfer Protocol (HTTP/1.1)
- Uniform Resource Locator (URL)
  `protocol :// host [:port] /filepath [ref]`
  - Spezifikation siehe w3.org:
    http://www.w3.org/Addressing/URL/url-spec.html
- Beispiele:
  - `http://www.somewhere.com:80/remoteSurvey.html`
  - `https://secure.tiscover.com:443`
  - `ftp://www.nowhere.net/`
  - `file://D:/tmp/bsp.html`
  - `mailto://alexander.sommer@tiscover.com`

Netzwerkprogrammierung
Verwendung von URLs

- Ein Objekt der Klasse URL repräsentiert eine Ressource im Internet (z.B. HTML Dokumente, CGI Programme, ...)

```
try {
  URL u = new URL("http://java.sun.com/index.html");
} catch(MalformedURLException e) { … }
// URL(String protocol, String host, int port, String file)
```

- Parsen:

```
u.getProtocol();
u.getHost(); // ... (File, Port, Ref)
```

- Direktes Lesen von einer URL:

```
InputStream in = u.openStream();
```

Netzwerkprogrammierung

URLConnection

- Verbindung wird durch ein URLConnection Objekt repräsentiert und bei dessen Instanziierung aufgebaut. Generell ist das Erzeugen einer Connection zu einem Url ein mehrstufiger Prozess.

```
URLConnection uc = u.openConnection();
u.connect();
```

- Datenaustausch erfolgt über Streams der URLConnection

```
InputStream ui = uc.getInputStream();
OutputStream uo = uc.getOutputStream();
```

- **HttpURLConnection**

```
HttpURLConnection http_uc = (HttpURLConnection)uc;
uc.getRequestMethod();
uc.getResponseMessage();
```

tiscover
THE TRAVEL NETWORK

Netzwerkprogrammierung
Beispiel: "mailto:"

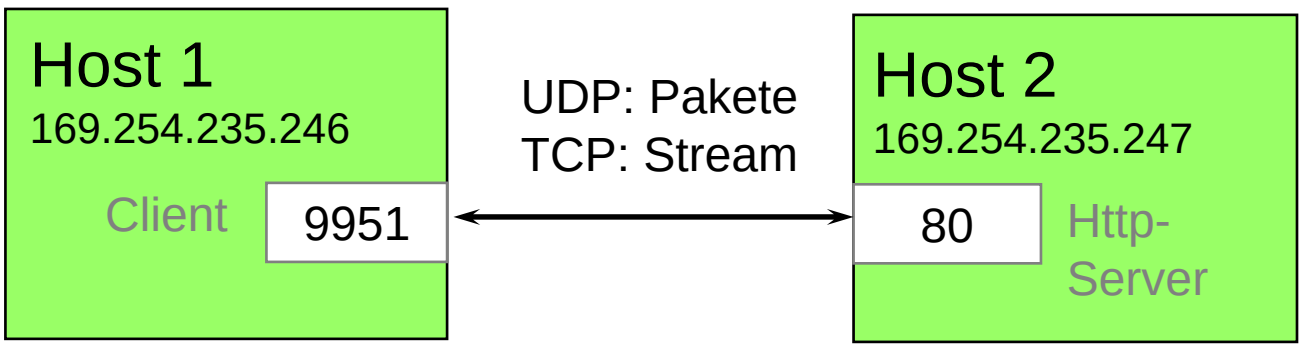
```
import java.io.*;
import java.net.*;

  ...
  System.setProperty("mail.host", „smtp.tiscover.com");
  URL u = new URL("mailto:alexander.sommer@tiscover.com");
  URLConnection conn = u.openConnection();
  PrintWriter out = new PrintWriter(
      new OutputStreamWriter(conn.getOutputStream()));
  out.println("Subject: Test");
  out.println();
  out.println("Hello World!");
  out.close();
  ...
```

Netzwerkprogrammierung
TCP/IP, UDP

- Internet Protocol (IP)
  - Routing der IP-Pakete (versehen mit Empfänger, Sender)
- Transport Control Protocol (TCP)
  - Verbindung über gemeinsamen Kommunikationskanal
  - Stückelt Nachricht in mehrere durchnummerierte Pakete
  - Überprüft ob Paket Ziel erreicht hat, ansonsten Wiederholung
  - Empfänger setzt Pakete in richtiger Reihenfolge wieder zusammen
- Uniform Datagram Protocol (UDP)
  - Keine ständige Verbindung - zustandslos
  - Jedes Paket wird einzeln gesendet
  - Paket kann verloren gehen, keine Kontrolle
  - *Multicast* ist möglich

Netzwerkprogrammierung
Sockets

- Sockets sind die Schnittstelle zwischen Anwendung und Netzwerkverbindung
- `Host:Port`: Zur Identifizierung der Anwendungen



- Die "Server"-Anwendung wartet auf einem vereinbarten Port auf Verbindungsaufnahmen durch Client-Anwendungen

Netzwerkprogrammierung
UDP – paketorientiert

- Datenaustausch basiert auf Paketen (*Datagrammen*)
- `DatagramPacket` repräsentiert Datenpaket
  - Paket zum Senden:
    `DatagramPacket(byte[] buf, int length,`
    `<Host>, <Port>)`
  - Paket zum Empfangen:
    `DatagramPacket(byte[] b, int len)`
- `DatagramSocket` zum Senden und Empfangen

```
public void send(DatagramPacket d);
public void receive(DatagramPacket d);
```

Netzwerkprogrammierung
Beispiel: UDP Time *Client* (1/3)

- Schritt 1: Initialisierung

```
import java.net.*;
import java.io.*;

public class TimeClientUDP {
  public static void main (String[] args)
    throws IOException {
    if (args.length != 1) {
      System.out.println("Usage: java TimeClUDP hostname");
      System.exit(1);
    }
    // get a datagram socket
    DatagramSocket socket = new DatagramSocket();
```

Netzwerkprogrammierung
Beispiel: UDP Time *Client* (2/3)

- Schritt 2: Request schicken

```
// send request
// args[0] is the host name
byte[] buf = new byte[256];
InetAddress address = InetAddress.getByName(args[0]);

DatagramPacket packet =
  new DatagramPacket(buf, buf.length, address, 8889);

socket.send(packet);
```

Netzwerkprogrammierung
Beispiel: UDP Time *Client* (3/3)

- Schritt 3: Response empfangen

```
// get response
packet = new DatagramPacket(buf, buf.length);
socket.receive(packet);

// display response
String received = new String(packet.getData());
System.out.println("Time received: " + received);

socket.close();
}}
```

Netzwerkprogrammierung
Beispiel: UDP Time *Server* (1/2)

- Schritt 1: UDP Socket erzeugen und auf Request warten

```
// imports, public class TimeServerUDP …
// in main()
DatagramSocket socket = new DatagramSocket(8889);
try {
  byte[] buf = new byte[80];
  // receive request
  DatagramPacket packet = new DatagramPacket(buf,
    buf.length);
  socket.receive(packet);
  System.out.println("Time request received.");
```

Netzwerkprogrammierung

Beispiel: UDP Time *Server* (2/2)

Schritt 2: Response zurückschicken

```
// construct date of time string
String timeofday = new Date().toString();
buf = timeofday.getBytes();

// send the response back
packet.setData(buf);
socket.send(packet);
 /* InetAddress address = packet.getAddress();
  int port = packet.getPort();
  packet = new DatagramPacket(buf,buf.length,address,port);
  socket.send(packet); */

System.out.println("Time response sent.");
} catch (IOException e) { … // exception handling }
socket.close();
}}
```

Packet kennt den Absender

Netzwerkprogrammierung

TCP – verbindungsorientiert

- Kommunikation erfolgt bidirektional über **Sockets**
- Socket := (*Host*, *Portnummer*)
- Implementierung des Datenaustausches mit **Streams**
- Client stellt Verbindung her durch Instanziierung eines `Socket` Objektes
- Server ist implementiert durch `ServerSocket` und wartet auf einer bestimmten Portnummer auf Anfragen:

```
public Socket accept();
```

- Verbindungsabbruch durch Schließen des Sockets

Netzwerkprogrammierung
Verbindungsaufbau

- Client

```
Socket s = new Socket(<Host>,<Port>);
InputStream si = s.getInputStream();
OutputStream so = s.getOutputStream();
```

- Server (gestartet auf <Host>)

```
ServerSocket ss = new ServerSocket(<Port>);
Socket s = ss.accept();  // blockierend
InputStream from_so = s.getInputStream();
OutputStream to_si = s.getOutputStream();
```

- Das Socket Objekt repräsentiert die Verbindung

Netzwerkprogrammierung

Beispiel: TCP Time Client

```
import java.net.*;
import java.io.*;

public class TimeClient {
  public static void main (String args[]) throws IOException {
    Socket server = new Socket("localhost",1234);
    System.out.println("Connected to " +
      server.getInetAddress());
    DataInputStream in = new DataInputStream(
      new BufferedInputStream(server.getInputStream()));
    System.out.println("Server said: "+ in.readUTF());
    server.close();
  }
}
```

Netzwerkprogrammierung

Beispiel: TCP Time Server

```
// in main()
ServerSocket server = new ServerSocket(1234);
while (true) {
  System.out.println("Waiting for client...");
  Socket client = server.accept();
  System.out.println("Client from “ +
    client.getInetAddress() + " connected.");
  DataOutputStream out = new DataOutputStream(
    new BufferedOutputStream( client.getOutputStream()));
  Date date = new Date();
  System.out.println(date.toString());
  out.writeUTF(date.toString());
  out.flush();
  client.close();
}
```

Netzwerkprogrammierung
Datenaustausch

- Lesen vom InputStream, schreiben auf den OutputStream
- **Protokoll** muss vereinbart werden
  - Kodierung der Daten
  - Koordination der Aktivitäten (Kontrollfluss)
- Beispiel: **Simple Mail Transfer Protocol** (RFC-821)
  - Zeilenweiser Austausch von Kommandos und Statusantwort
  - Client sendet Kommando an den SMTP-Server
    (z.B. "MAIL FROM:<somebody@somewhere.com>\r\n")
  - Server antwortet mit Status-Code
    (z.B. "250 OK\r\n")

Netzwerkprogrammierung
Ressourcen

Networking Tutorial:
http://java.sun.com/docs/books/tutorial/networking/

Java Newsletters:
http://java.sun.com/community/newsletters/index.html

Java Network Programming FAQ:
http://www.davidreilly.com/java/java_network_programming/

Networking with Java:
http://www.javacoffeebreak.com/java109/java109.html

Java Buch-DE: http://www.javabuch.de

Ende der 3. Übung