Verteilte Systeme – Übung

Java RMI

Sommersemester 2021

Michael Eischer, Laura Lawniczak, Tobias Distler

Friedrich-Alexander-Universität Erlangen-Nürnberg
Lehrstuhl Informatik 4 (Verteilte Systeme und Betriebssysteme)
www4.cs.fau.de

Lehrstuhl für Verteilte Systeme
und Betriebssysteme

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

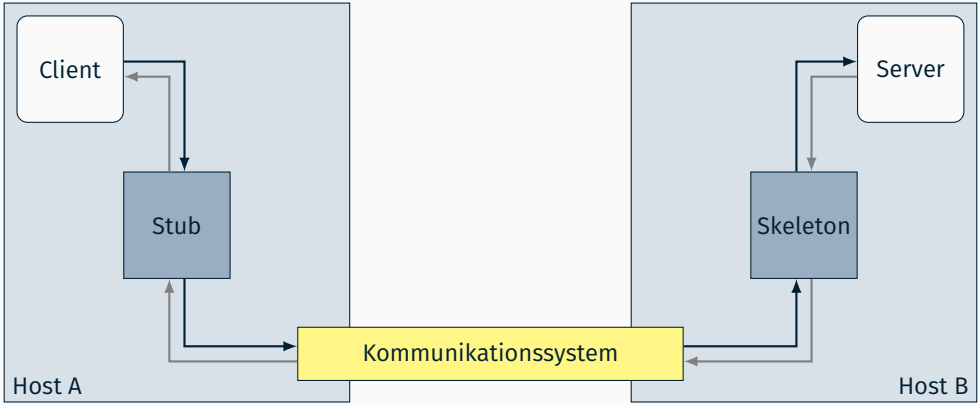
TECHNISCHE FAKULTÄT

Überblick

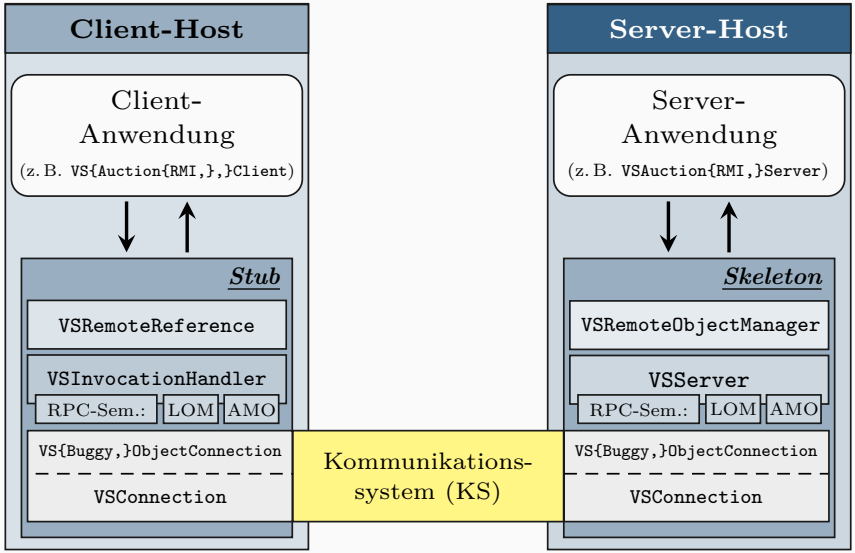
Aufgabe 1: Java RMI

Verteilte Systeme: Übungsaufgaben 1-3

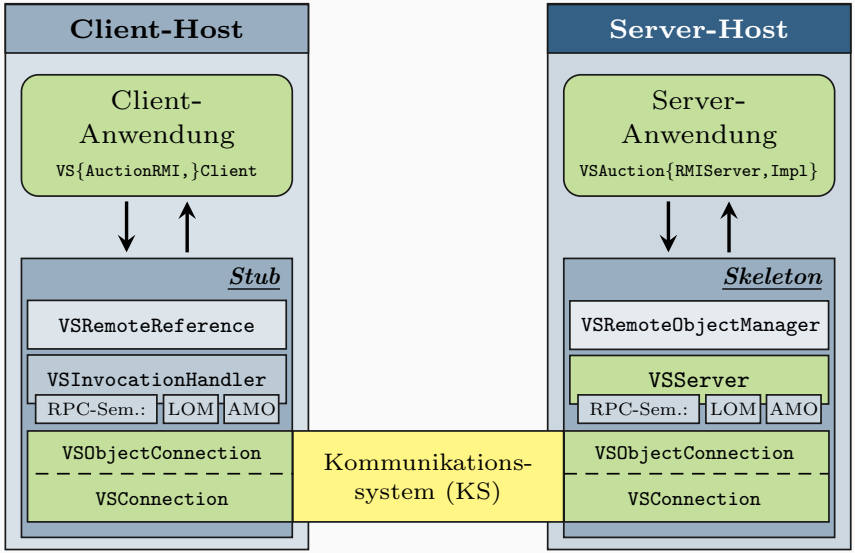
- Entwicklung eines eigenen Fernaufrufsystems
- Orientierung an Java RMI



Gesamtüberblick (Übungsaufgaben 1 bis 3)



Übungsaufgabe 1



Programmieren mit Java RMI

- Beispielanwendung: Auktionsdienst

```
public interface VSAuctionService {
  public void registerAuction(VSAuction auction, int duration,
          VSAuctionEventHandler handler) throws VSAuctionException;
  public VSAuction[] getAuctions();
  public boolean placeBid(String userName, String auctionName, int price,
          VSAuctionEventHandler handler) throws VSAuctionException;
}
```

```
public interface VSAuctionEventHandler {
  public void handleEvent(VSAuctionEventType event, VSAuction auction);
}
```

  - `registerAuction()` Registrieren einer neuen Auktion
  - `getAuctions()` Abfragen aller laufenden Auktionen
  - `placeBid()` Neues Gebot für eine laufende Auktion abgeben

- Verteilung mittels Java RMI
  - Server
    - Bereitstellung der Anwendung als Remote-Objekt
    - Bekanntmachen des Diensts bei einer Registry
  - Client
    - Zugriff auf den Dienst über Fernaufrufe
    - Interaktion mit dem Nutzer per Kommandozeile

Implementierung der Kommunikationsschicht

- Übertragung von Datenpaketen

```
public class VSConnection {
    public void sendChunk(byte[] chunk);
    public byte[] receiveChunk();
}
```

  - Senden und Empfangen von Byte-Arrays beliebiger Länge
  - Übermittlung von Daten über eine TCP-Verbindung

- Übertragung von Objekten

```
public class VSObjectConnection {
    public void sendObject(Serializable object);
    public Serializable receiveObject();
}
```

  - Senden und Empfangen von beliebigen Objekten
  - Marshalling und Unmarshalling

- Ziel

  Minimierung der über das Netzwerk zu übertragenden Daten

- Ausgangspunkt
  - Analyse der vom `ObjectOutputStream` erzeugten Daten
  - Beispielklasse

```
public class VSTestMessage implements Serializable {
    private int integer;
    private String string;
    private Object[] objects;
}
```

- Reduzierung der benötigten Datenmenge
  - Anwendung der Schnittstelle `Externalizable`
  - Manuelle Implementierung der {S,Des}erialisierungmethoden
- Hinweis: Ausgabe eines Byte-Array als Zeichenkette in Eclipse

```
byte[] chunk = [...];
System.out.println(new String(chunk).replace("\0", "\ufffe"));
```