Verteilte Systeme – Übung

Aufgabe 3

Sommersemester 2023

Laura Lawniczak, Harald Böhm, Tobias Distler

Friedrich-Alexander-Universität Erlangen-Nürnberg
Lehrstuhl Informatik 4 (Verteilte Systeme und Betriebssysteme)
Lehrstuhl Informatik 16 (Systemsoftware)

https://sys.cs.fau.de

Lehrstuhl für Verteilte Systeme
und Betriebssysteme

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

TECHNISCHE FAKULTÄT

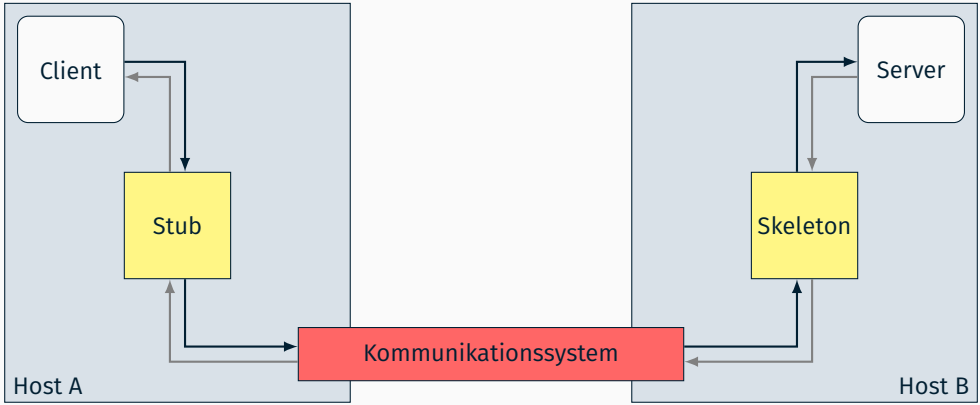
Überblick

Übungsaufgabe 3

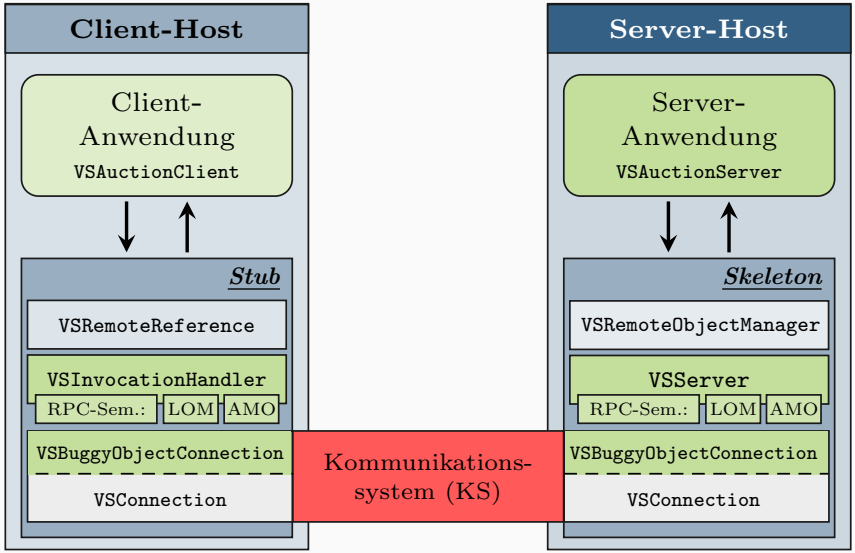
Übungsaufgabe 3

Übungsaufgabe 3

- Bereitstellung von Fehlertoleranzmechanismen
- Simulation von Kommunikationsfehlern



Übungsaufgabe 3



Implementierung der Fernaufrufsemantiken

- *Last-of-Many*
  - Fernaufruf-IDs
  - Sequenznummern
  - Timeouts
- *At-Most-Once*
  - Einmalige Ausführung
  - Speicherung der Ergebnisse
  - Garbage-Collection für Ergebnisse
- Auswahl der Fernaufrufsemantik
  - Methodenspezifische Festlegung
  - Annotierung der Anwendungsschnittstelle bei der Entwicklung
    - `@VSRPCSemantic(VSRPCSemanticType.LAST_OF_MANY)` bzw.
    - `@VSRPCSemantic(VSRPCSemanticType.AT_MOST_ONCE)`
  - Analyse der Annotation durch das Fernaufrufsystem zur Laufzeit

- Annotationen: Bereitstellung von Metadaten im Quelltext
- Beispiel: Kennzeichnung von schreibenden bzw. lesenden Methoden
  - Hilfs-`enum` zur Typunterscheidung

```
public enum VSMethodType {
    READ_ACCESS, WRITE_ACCESS
}
```

  - Definition der Annotation mittels `@interface` in `VSAnnotation.java`

```
@Retention(RetentionPolicy.RUNTIME)
public @interface VSAnnotation {
    VSMethodType value();
}
```

    - `@Retention`-Annotation: Sichtbarkeit von `VSAnnotation` zur Laufzeit
    - Spezifizierung des Rückgabetyps der Standardmethode `value()`

  - Einsatz der Annotation

```
@VSAnnotation(VSMethodType.WRITE_ACCESS)
```

[Hinweis: Sollte der Methodenname von „`value()`“ abweichen, muss beim Einsatz der Annotation der Methodenname explizit angegeben werden.
Beispiel: `foo()` → `@VSAnnotation(foo = VSMethodType.WRITE_ACCESS)`]

- Beispiel: Schnittstelle eines Speichers für Schlüssel-Wert-Paare

```
public interface VSKeyValueStore {
    @VSAnnotation(VSMethodType.WRITE_ACCESS)
    public void put(String key, String value);

    @VSAnnotation(VSMethodType.READ_ACCESS)
    public String get(String key);
}
```

- Analyse der Schnittstelle `VSKeyValueStore`
  - Zugriff auf Annotation mittels `Method.getAnnotation()`

```
for(Method method: VSKeyValueStore.class.getMethods()) {
    VSAnnotation annotation = method.getAnnotation(VSAnnotation.class);
    VSMethodType type = annotation.value();
    System.out.println(method.getName() + ": " + type);
}
```

  - Ausgabe

```
get: READ_ACCESS
put: WRITE_ACCESS
```

Sabotage des Kommunikationssystems

- Simulation von Kommunikationsfehlern
  - Nachrichtenverlust durch Verbindungsabbruch
  - Verzögerung einzelner Nachrichten
  - Nicht betrachtet
    - Korrumpierung von Nachrichten
    - Verlust von Teilnachrichten
- Tests
  - Variation der Fehlerintensität
  - Kombination verschiedener Fehlerarten
- Implementierungsvorschlag
  - Fehlerhafte `VSObjectConnection` → `VSBuggyObjectConnection`
  - Überschreiben von
    - `sendObject()` oder
    - `receiveObject()`
  - „Verbindungsabbruch“ durch Schließen der Verbindung per `close()`

Socket-Timeouts

- Setzen von Socket-Timeouts mittels `setSoTimeout()`
  - Konfigurierung der Maximaldauer, die ein Leseaufruf am Socket blockiert
  - Leseaufruf kehrt bei Timeout-Ablauf mit `SocketTimeoutException` zurück
- Beispiel

```
// Socket-Timeout setzen
Socket socket = [...];
try {
    socket.setSoTimeout(5000);
} catch(IOException ioe) {
    // Fehlerbehandlung
}

// Leseaufruf starten
try {
    socket.getInputStream().read();
} catch(SocketTimeoutException ste) {    // -> "Timeout: Read timed out"
    System.err.println("Timeout: " + ste.getMessage());
} catch(IOException ioe) {
    System.err.println("I/O error: " + ioe);
}
```