

Verteilte Systeme – Übung

Fernaufwurfsemantiken

Sommersemester 2025

Harald Böhm, Christian Berger, Tobias Distler

Friedrich-Alexander-Universität Erlangen-Nürnberg
Lehrstuhl für Informatik 4 (Systemsoftware)

<https://sys.cs.fau.de>



Lehrstuhl für Informatik 4
Systemsoftware



Friedrich-Alexander-Universität
Technische Fakultät

Fernaufsemantiken

Fehler bei Fernaufrufen

Fehlertolerante Fernaufrufe

Fernaufwurfsemantiken

Fehler bei Fernaufrufen

■ In der Anwendung begründete Fehler

- Fehlersituationen treten bei lokalem Methodenaufruf ebenfalls auf
- Beispiele
 - Falsche Eingaben [Vergleiche: `VSAuctionException` bei `VSAuctionService.registerAuction()`]
 - Programmierfehler in der Anwendung
- Reaktion des Fernaufrufsystems
 - Aus Sicht des Fernaufrufsystems: Reguläres Verhalten
 - Keine Fehlerbehandlung im Fernaufrufsystem → Transparente Signalisierung

■ Im Fernaufruf begründete Fehler

- Fehlersituationen sind bei lokalem Methodenaufruf nicht relevant
- Beispiele
 - Rechner: Prozess-, Programm-, Rechnerabsturz, Verzögerungen (Überlast)
 - Nachrichten: Reihenfolgeänderung, Korruption, Verlust
 - Verbindung: Verlangsamung, Abbruch
- Reaktion des Fernaufrufsystems
 - Fehlerbehandlung im Fernaufrufsystem
 - Signalisierung nur bei Scheitern der Fehlerbehandlung

■ Rechnerfehler

- Lokaler Methodenaufruf
 - Aufrufer und Aufgerufener in gleichem Maße betroffen
 - Im Fehlerfall sind beide abgestürzt bzw. langsam
- Fernaufruf
 - Aufrufer und Aufgerufener können unabhängig ausfallen
 - Im Fehlerfall ist eventuell nur einer betroffen

■ Kommunikationsfehler

- Lokaler Methodenaufruf
 - Keine Netzwerkkommunikation
 - Fehlerart nicht relevant
- Fernaufruf
 - Temporäre oder sogar dauerhafte Fehler möglich
 - Nicht alle Fehler lassen sich im Fernaufrufsystem tolerieren

⇒ **Komplexeres Fehlermodell macht vollständig transparente Fernaufrufe unmöglich!**

■ Fehlertolerierung

- Einsatz von Fernaufrufsemantiken
- Problem: Nicht alle Fehler lassen sich tolerieren

■ Fehlersignalisierung

- **Verletzung der Transparenzeigenschaften**
 - Benachrichtigung an den Benutzer des Fernaufrufsystems
 - Benutzer des Fernaufrufsystems muss darauf vorbereitet sein
- Umsetzung in Java RMI mittels `java.rmi.RemoteException`
 - Muss von jeder Methode einer Remote-Schnittstelle geworfen werden
 - Unterklassen von `RemoteException` (Beispiele)

Exception	Beschreibung
<code>ConnectException</code>	Verbindungsaufbau fehlgeschlagen
<code>ServerError</code>	Auspacken der Anfrage, Ausführung der Methode oder Einpacken der Antwort fehlgeschlagen
<code>NoSuchObjectException</code>	Remote-Objekt nicht (mehr) verfügbar
<code>UnknownHostException</code>	Remote-Host nicht bekannt

- Probleme
 - Keine definitive Fehlererkennung (Liegt überhaupt ein Fehler vor?)
 - Keine exakte Fehlerlokalisierung (Wo liegt der Fehler?)
 - Beispielszenario: Ein Client erhält keine Antwort auf seine Anfrage
 - Mögliche Gründe
 - Anfrage ging verloren
 - Antwort ging verloren
 - Server ausgefallen
 - Server überlastet
 - Netzwerk überlastet
 - ...
 - Konsequenz: Mindestens einer der beiden Fernaufruf-Teilnehmer kann nicht erkennen, ob (und wenn ja, wo) ein Fehler vorliegt
- ⇒ **Eine präzise Fehlererkennung ist in verteilten Systemen im Allgemeinen nicht möglich!**

Fernaufrufsemantiken

Fehlertolerante Fernaufrufe

- **Ansatzpunkt**
 - Tolerierung von Kommunikationsfehlern
 - Wiederanlaufen nach Rechnerausfällen erfordert zusätzliche Mechanismen
- **Semantiken**
 - Maybe
 - At-Least-Once
 - At-Most-Once
 - Last-of-Many
- **Unterschiede**
 - Mehrmaliges Senden von Anfragen
 - Aktualität der Antworten
 - Anzahl der Ausführungen
 - Idempotente Operationen?
 - Duplikaterkennung?
 - Antwortspeicherung → Wie lange wird eine Antwort aufgehoben?

■ At-Least-Once

- Funktionsweise
 - Client wiederholt Anfrage, falls Antwort ausbleibt
 - Client akzeptiert die erste Antwort, die ihn erreicht
- Eigenschaften
 - Client verwendet eventuell veraltete Antwort
 - Anfragen werden eventuell mehrfach ausgeführt

■ At-Most-Once

- Funktionsweise
 - Client wiederholt Anfrage, falls Antwort ausbleibt
 - Server speichert Antwort
 - Server sendet bei Anfragewiederholungen gespeicherte Antwort
- Eigenschaften
 - Anfragen werden höchstens einmal ausgeführt
 - Speichern von Antworten erforderlich

■ Last-of-Many

■ Funktionsweise

- Client wiederholt Anfrage, falls Antwort ausbleibt
- Client akzeptiert nur Antwort auf seine aktuellste Anfrage

■ Eigenschaften

- Keine Antwortspeicherung nötig
- Anfragen werden eventuell mehrfach ausgeführt

■ Implementierung der Semantiken

■ Allgemein: Fernaufruf muss eindeutig identifizierbar sein

- Client
- Remote-Objekt
- Remote-Methode
- Aufrufzähler

■ Zusätzlich bei LOM: Eindeutige Identifizierung jeder Fernaufrufnachricht

- Anfragezähler
- Zuordnung: Antwort zu Anfrage

■ Idempotente Funktionen (Mathematik)

■ Definition

$$f(x) = f(f(x))$$

■ Beispiele: Operationen auf Mengen

- Konstante Funktion
- Hinzufügen eines bestimmten Elements
- Entfernen eines bestimmten Elements

$$f(S) = \{c\}$$

$$f(S) = S \cup \{c\}$$

$$f(S) = S \setminus \{c\}$$

■ Idempotente Operationen (Informatik)

■ Charakteristika mehrfacher Ausführungen

- Identische Anwendungszustände
- (*Identische Rückgabewerte*)

■ Beispiele

- Leseoperationen
- Zustandsmodifikation durch Setzen neuer Daten

■ Triviale Kombination idempotenter Operationen nicht immer idempotent

- Problem
 - Server stellt eigene Ressourcen für Fernaufrufe bereit (→ Antwort-Cache)
 - Mit jedem neuen Fernaufruf werden zusätzliche Ressourcen belegt
 - Wann können die gespeicherten Antworten verworfen werden?
- Lösungsansätze (Kombinationen möglich bzw. nötig)
 - Explizit
 - Benachrichtigung durch Client oder Nachfrage vom Server
 - Problem: Nicht alle Clients können oder wollen sich daran halten
 - Implizit
 - Bei neuem Fernaufruf eines Clients wird die alte Antwort gelöscht
 - Problem: Letzter Fernaufruf eines Clients
 - Timeout
 - Antwortlöschung nach Ablauf eines fernaufrufspezifischen Timeout
 - Als Rückfallposition immer nötig
- Herausforderung: Aufrechterhaltung der Semantikgarantien