Netzprogrammierung
4. Interaktionsmuster
Remote Procedure Calls

Prof. Dr.-Ing. Robert Tolksdorf
Freie Universität Berlin
Institut für Informatik
Netzbasierte Informationssysteme
mailto: tolk@inf.fu-berlin.de
http://www.robert-tolksdorf.de

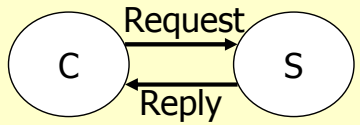
Überblick

1. Muster mit mehreren Servern
2. Remote Procedure Call
3. Komponenten beim RPC
4. Fehler

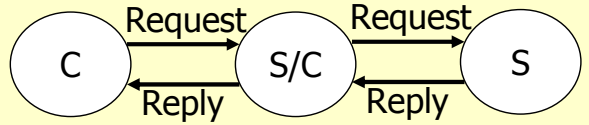
Muster mit mehreren Servern

Client-Server-Server Systeme

- Client-Server tritt selten pur auf



- Beispiel: 3-Tier/3-Schichten:

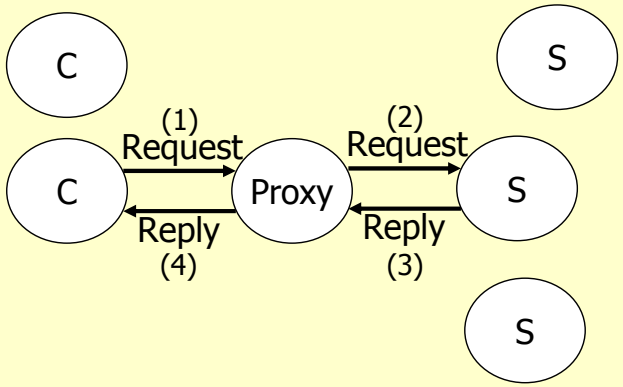


Web-Browser Web-Server Datenbank

- Es gibt verschiedene wiederkehrende Muster solcher Client-Server-Server Systeme

Proxy

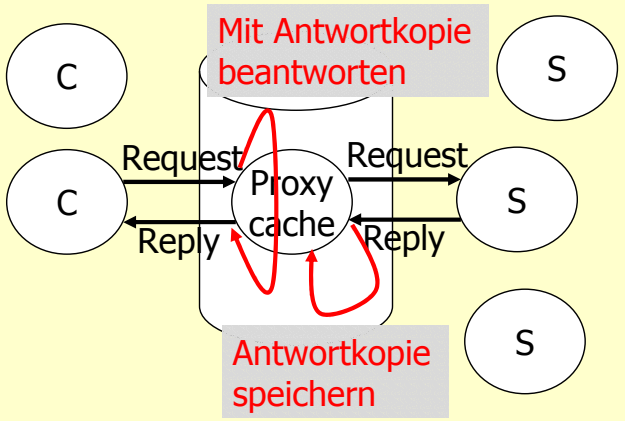
- Proxy-Objekt vertritt
  - Mehrere Clients gegenüber Servern
  - Mehrere Server gegenüber Clients



- Beispiel: Web-Proxy

Proxy mit Zwischenspeicher

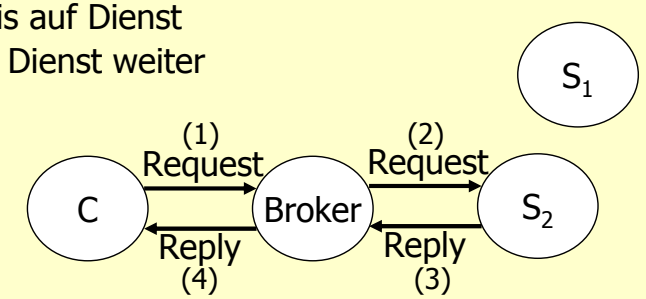
- Proxy Cache
  - Speichert erhaltene Antworten ab
  - Beantwortet Anfragen möglichst aus Zwischenspeicher

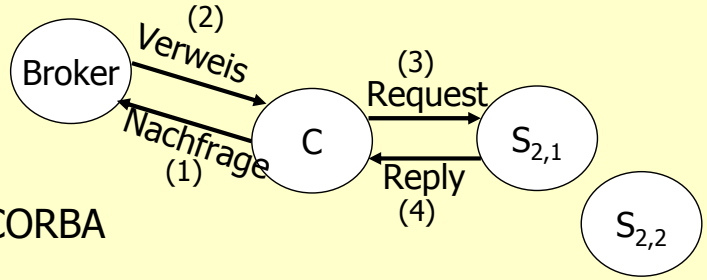


- Beispiel: Caching Web-Proxy

Broker, auch Trader

- Broker
  - vermittelt Verweis auf Dienst
  - leitet Anfrage an Dienst weiter
- Weiterleitend
- Verweisend

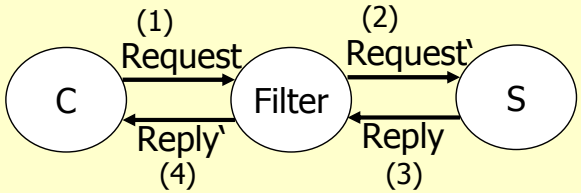




- Beispiel: CORBA

Filter

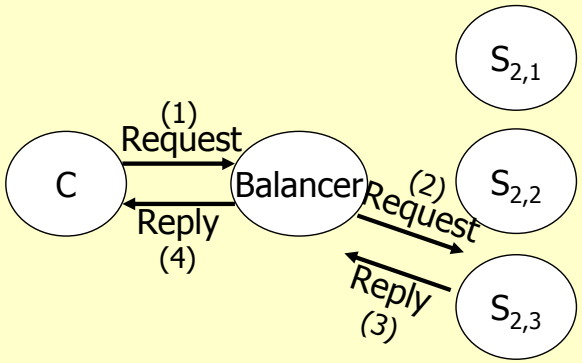
- Filter
  - leitet modifizierte Anfrage weiter
  - leitet modifizierte Antwort weiter



- Beispiel: Web-Anonymisierer

Balancer

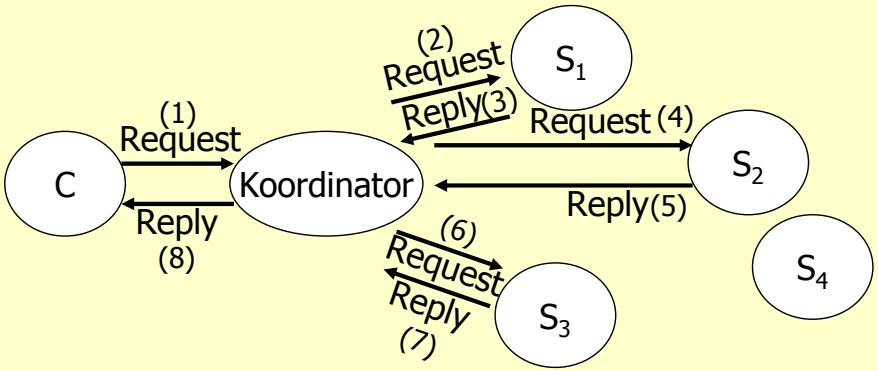
- Balancer
  - gleicht Lastverteilung durch ausgeglichene Anfragenweiterleitung aus



- Beispiel: Web-Server aus mehreren Maschinen

Koordinator

- Koordinator
  - realisiert zusammengesetzten Dienst durch entsprechende Aufrufe an



- Beispiel: Onlinetransaktion (Flugticketbuchung, Mietwagenbuchung, Kreditkartenabbuchung)

Remote Procedure Calls

Remote Procedure Call RPC

- Entfernter Prozeduraufruf, Remote Procedure Call, als grundlegender Mechanismus für verteilte Systeme
  - direkt implementiert (Sun RPC, HP RPC etc.)
  - weiterentwickelt (Java RMI etc.)
  - Konzeptionell enthalten (HTTP etc.)

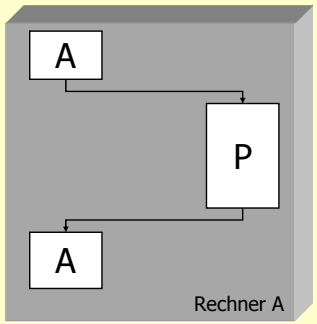
- [Birrell/Nelson84]:
  RPC ist ein synchroner Mechanismus, der Kontrollfluss und Daten als Prozeduraufruf zwischen zwei Adressräumen über ein schmalbandiges Netz transferiert

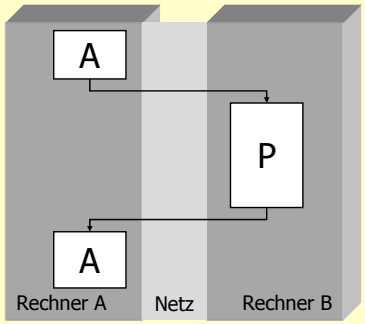
RPC Kontroll- und Datenfluss

- Prozeduraufruf steuert
  - Kontrollfluss und
  - Parameter

  vom Aufrufer in eine Prozedur
- Prozedurbeendigung steuert
  - Kontrollfluss und
  - Ergebnisdaten

  zurück
- Entfernter Prozeduraufruf (Remote Procedure Call, RPC) transferiert Kontrollfluss und Daten über ein Netzwerk zwischen Rechnern





RPC Eigenschaften

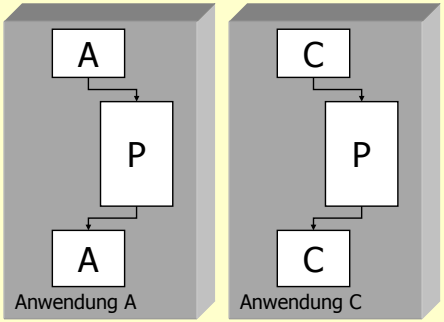
- Synchron:
  Aufrufer blockiert bis Aufgerufener Ergebnis abliefert
- Prozeduraufruf:
  Signatur der Prozedur definiert zu übertragende Daten
- Unterschiedlicher Adressraum:
  Speicheradressen (Zeiger) sind nicht semantikerhaltend übertragbar
- Schmalbandig:
  Bandbreite des Netzes ist um Dimensionen geringer als die der Kommunikationspfade innerhalb eines Rechners

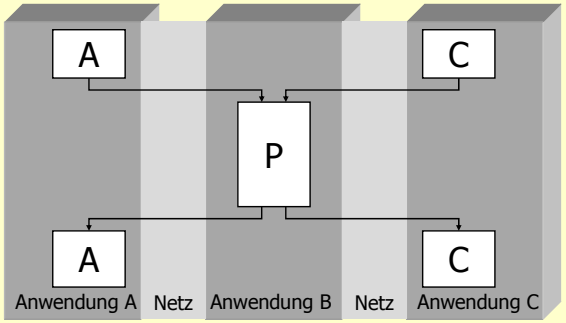
Unterschiede

Lokaler Aufruf	*Entfernter Aufruf*
Aufrufer und Prozedur im selben Prozess ausgeführt	Aufrufer und Prozedur in unterschiedlichen nebenläufigen Prozessen
Aufrufer und Prozedur im selben Adressraum	Aufrufer und Prozedur in unterschiedlichen Adressräumen
Aufrufer und Prozedur in selben Hard- und Software-umgebung	Aufrufer und Prozedur in unterschiedlicher Hard- und Softwareumgebung
Aufrufer und Prozedur haben gleiche Lebensdauer	Aufrufer und Prozedur haben unterschiedliche Lebensdauer
Aufruf ist immer fehlerfrei	Aufruf ist fehlerbehaftet (Netz, Aufgerufener)
Nur Anwendungsfehler berücksichtigt	Zusätzlich Aufruffehler behandeln

Vorteile der Netzbasierung

- Bessere Aufgabenverteilung
- Bessere Lastverteilung
- Bessere Ressourcennutzung
- Bessere Modularität
- Bessere Wieder-verwendbarkeit
- Größere Offenheit
- Besser Integrationsfähigkeit
- ...



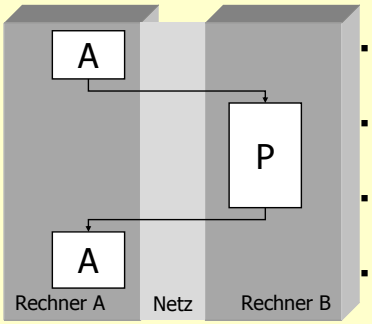


Vorteile der Prozedurabstraktion

- Klare und verständliche Semantik von RPC
- Prozeduraufrufe wohlverstanden
- Prozeduraufrufe geeignetes Mittel zur Kommunikation in Anwendungen
- Einfachheit des Mechanismus: Effiziente Implementierung möglich

Ablauf RPC

- Anhalten des Kontrollfluss auf A
- Verpacken von Parametern
- Übersenden der Parameter
- Auspacken des Ergebnisses
- Fortführen des Kontrollfluss auf A

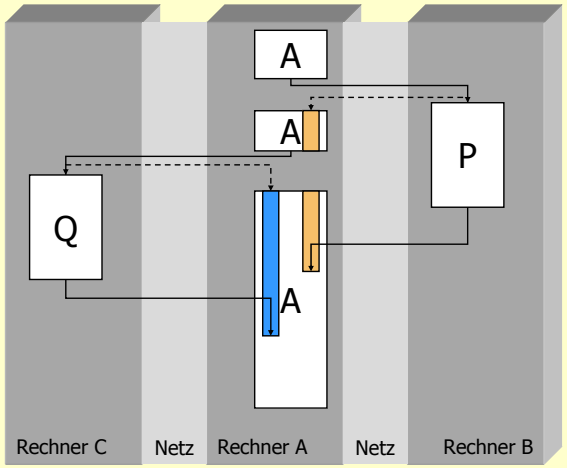


- Auspacken der Parameter
- Aufruf der Prozedur in Kontrollfluss auf B
- Verpacken des Ergebnisses
- Übersenden des Ergebnisses

Asynchroner RPC

- RPC ist eigentlich synchron, Klient ist blockiert
- Asynchroner RPC
  - Klient nicht blockiert
  - Holt Ergebnis später ab
- Future:
  - „Platzhalter" für Ergebnis
  - Lesen des Futures bewirkt Blockierung bis Ergebnis da

Asynchroner RPC



- Ermöglicht größeren Grad der Nebenläufigkeit:
  - Mehrere RPCs können „offen" sein
  - Mehrere Prozesse in Servern arbeiten an Ergebnissen

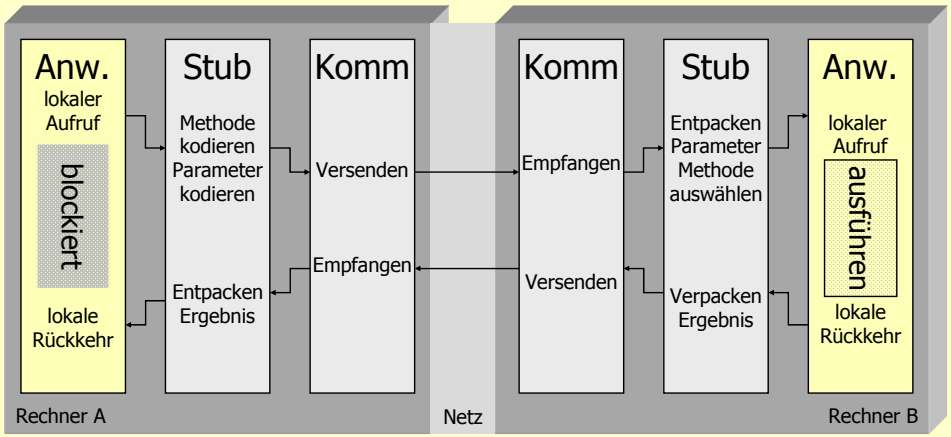
Herausforderungen

- Semantik im Fehlerfall
- Semantik von Zeigern
- Einbettung in Programmiersprachen
- Auffinden und Binden an die entfernten Prozeduren
- Protokoll des Datenaustauschs
- Eigenschaften der Kommunikation

Komponenten beim RPC

Komponenten beim RPC

- Anwendungsprozeduren: Eigentliche Arbeit
- Stubs: Ver- und Entpacken von Daten zum Transport
- Kommunikation: Transport von Daten
- Ablauf schematisch:

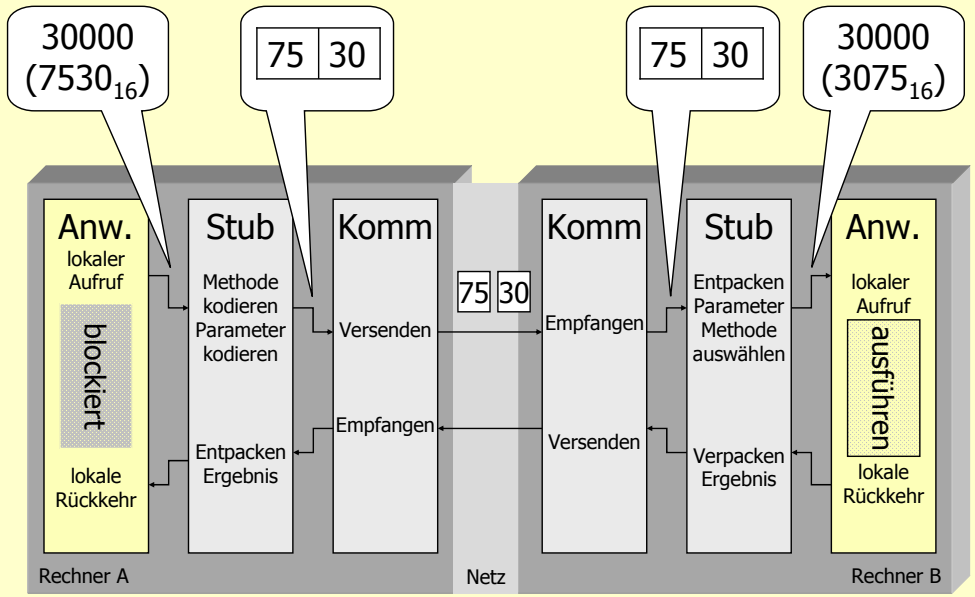


Marshalling

- Rechner können unterschiedliche Darstellung von Daten haben
  - Big-Endian („network order"): Absteigende Wertigkeit
    30000_{10} 7530_{16} 0111010100110000_2
  - Little-Endian: Aufsteigende Wertigkeit
    30000_{10} 3075_{16} 0011000001110101_2
- Komplexer bei zusammengesetzten Typen:
  - Beim Datum schon im realen Leben kompliziert:
    - Europa: dd.mm.yy (little-endian)
    - Japan: yy/mm/dd (big-endian)
    - US: mm/dd/yy („middle-endian")
  - Datenstrukturen?
- Notwendig:
  - Externe Datenrepräsentation
  - Kodierung/Dekodierung = Marshalling

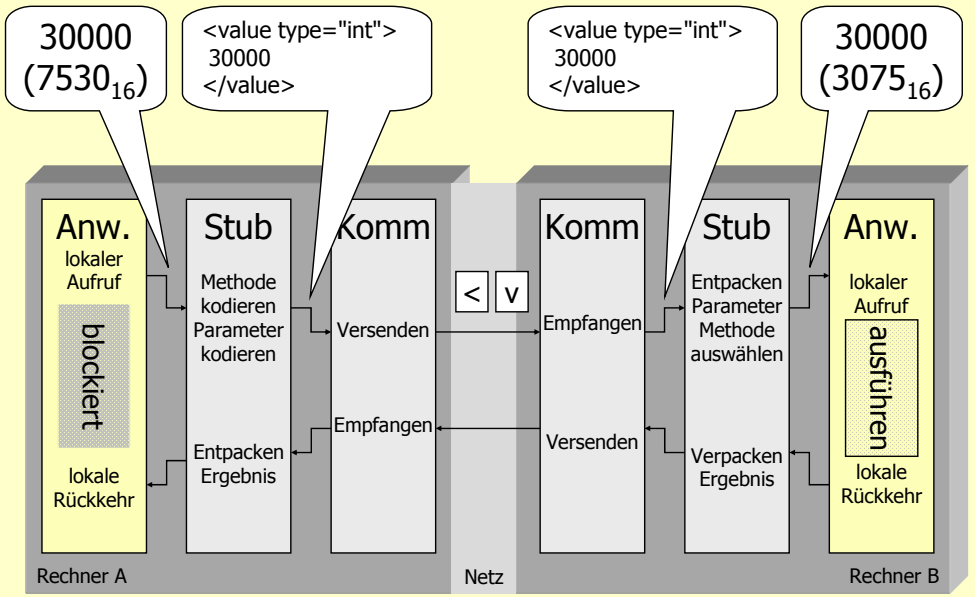
Marshalling

- Binär:



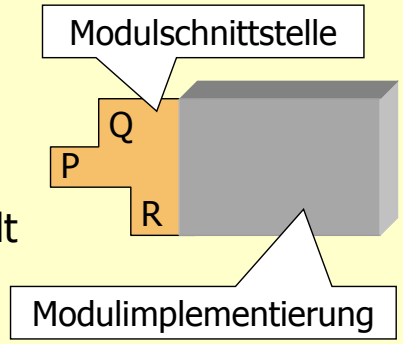
Marshalling

- Anders:

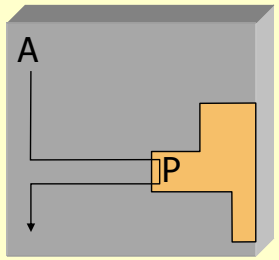


Interaktionspunkt: Schnittstellen (lokal)

- Ein Modul bietet Prozeduren zum Aufruf an:
  Das Modul *exportiert* eine Schnittstelle (Interface)
- Im Beispiel: Schnittstelle enthält Prozeduren P, Q und R
- Definiert durch Signaturen

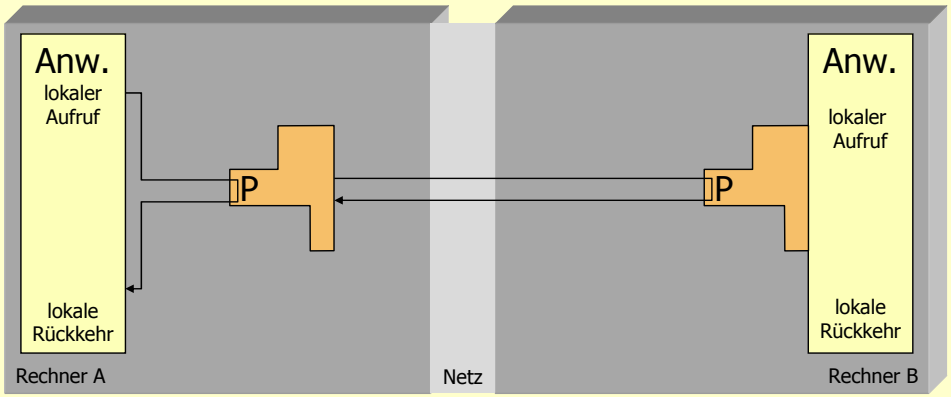


- Ein anderes Modul ruft diese Prozeduren an:
  Das Modul *importiert* eine Schnittstelle



Interaktionspunkt: Schnittstellen (entfernt)

- Anwender-Stub stellt Schnittstelle lokal bereit
- Stubs und Kommunikationskomponente leiten Aufrufe an Schnittstelle des entfernten Moduls weiter
- Stubs und Kommunikationskomponente leiten Ergebnisse des entfernten Moduls zurück



RPC Automatisierung

- Aus der Definition einer Schnittstelle kann man automatisch
  - Stub auf Aufruferseite
  - Stub auf Modulseite

  generieren
- Die Kommunikationskomponente ist generisch und muss lediglich
  - senden
  - empfangen

  können

Stubgenerierung

- Sei die die Schnittstelle mit drei Methoden {P(int a, int b), Q(int a, float f), int R(string s)}
- Stub für Aufrufer:

```
def P(int a, int b) { sende(1); sende(a); sende(b)}
def Q(int a, float f) { sende(2); sende(a); sende(f)}
def R(string s) { sende(3); sende(s); empfange i}
```

- Stub für Modul:

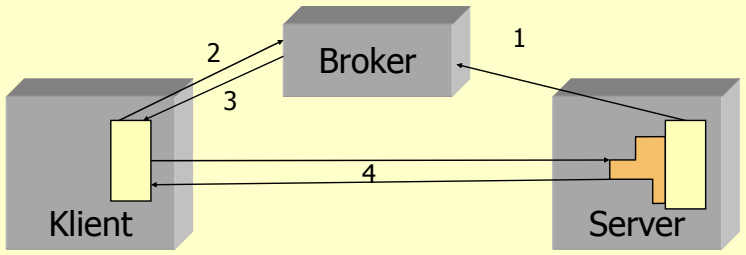
```
empfange(p_id);
switch (p_id) {
 case 1: empfange(a); empfange (b); P(a,b);
 case 2: empfange(a); empfange (f); Q(a,f);
 case 3: empfange(s); r=R(a,b); sende(r);
}
```


Bindung

- Wo schickt das Kommunikationssystem eigentlich den verpackten Prozeduraufruf hin?
- *Bindung* zwischen Aufrufer und Aufgerufenem
- Statische Bindung
  - Zur Übersetzungszeit werden Klienten an feste Serveradressen gebunden
- Halbstatische Bindung
  - Zur Startzeit des Klienten werden Serveradressen konfiguriert
  - Ermittelt aus Datenbank/Tabelle
- Dynamisch
  - Zur Aufrufzeit werden Serveradressen ermittelt
  - Serveradresse kann zwischen zwei RPCs wechseln

Vermittler

- Trader/Broker/Mediator: Komponente, die Server kennt und Referenzen auf sie vermitteln kann
  - Registrierung/Abmeldung von Servern (1)
    - Angabe der exportierten Schnittstelle
    - Referenz auf sich
  - Aufsuchen eines passenden Servers
    - Angabe einer Schnittstellenbeschreibung (2)
    - Ergebnis: Referenz (3) für RPC (4)



Designentscheidungen

- Designentscheidungen
  - Welche Informationen speichert der Broker
  - Nur Schnittstellen, weitere Informationen?
  - Nur statische Informationen, auch dynamische?
  - Welche Informationen gibt der Broker heraus?
  - Ist der Broker nur beim Auffinden von Referenzen beteiligt oder vermittelt er jeden RPC Aufruf?
- Bei allen Technologien finden wir eine Art Broker
  - Java RMI: Registry
  - OMG OMA: CORBA/ORB
  - Web Services: UDDI
  - Internet: DNS
  - …

Fehler

Fehlerquellen / Ausfälle

- Server nicht erreichbar für RPC
  - Maschine ausgefallen
  - Netzwerk ausgefallen
  - Transportschicht maskiert Fehler durch Neuversuche etc.
  - Broker maskiert Fehler durch andere Serverwahl
  - Klient muss Fehler verarbeiten
- Server fällt während RPC Ausführung aus
  - Ausfall vor RPC -> Nicht erreichbar
  - Ausfall während RPC -> Klient wartet, Seiteneffekte realisiert
  - Ausfall nach RPC -> Klient wartet, Seiteneffekte eingetreten

Fehlerquellen / Ausfälle

- Klient fällt während RPC Ausführung aus
  - Server kann Ergebnis nicht abliefern
  - Neugestarteter Klient kann „alte" RPCs absagen
  - Neugestarteter Klient kann „alten" RPC übernehmen
  - Server kann Klient überwachen, bei Timeout und Feststellung von Klient-Ausfall RPC abbrechen
  - Server kann bei Ergebnisablieferung Timeout setzen und Ergebnis bei Ablauf verwerfen

Fehlerquellen

- Server hat Schnittstelle geändert und bietet Dienst nicht mehr an
- RPC Mitteilungen gehen verloren
  - Anforderung, Teile davon
  - Ergebnis

Ablaufsemantik

- Mix aus Fehlerquellen und Reaktion darauf ergibt unterschiedliche semantische Eigenschaften:
  - may-be Semantik:
    - Nachricht wird abgeschickt, Empfang nicht überprüft
    - RPC wird
      - nicht ausgeführt
      - einmal ausgeführt
    - Passend, wenn Klient kein Ergebnis braucht
  - at-least-once:
    - Nachrichtenempfang wird quittiert
    - Nachricht wird abgeschickt, bei keiner Antwort erneut
    - RPC wird
      - Mindestens 1 Mal ausführt
    - Problem: Mehrfache Bearbeitung kann zu inkonsistenten Daten führen

Ablaufsemantik

- at-most-once:
  - Erneute Anforderung wird mit bisherigen Anforderungen abgeglichen
  - RPC wird
    - höchstens 1 Mal ausgeführt, nicht teilweise
  - Ausfall vor Verarbeitung: Nicht ausgeführt
  - Ausfall während Verarbeitung: Neustart, Konsistenz durch Transaktion
  - Ausfall nach Verarbeitung: Ergebniskopie nach Neustart versandt
- exactly-once:
  - Anforderungen dauerhaft speichern
  - RPC wird
    - genau 1 Mal ausgeführt.

Zusammenfassung

Zusammenfassung

1. Muster mit mehreren Servern
   1. Client-Server selten pur auftretend
   2. Muster: Proxy, Proxy cache, Broker, Trader, Filter, Balancer Koordinator,…
2. Remote Procedure Call
   1. Vorteile Verteilung
   2. Unterschiede zum lokalen Prozeduraufruf
3. Komponenten beim RPC
   1. Schnittstellen
   2. Stubs
   3. Bindungen/Broker
4. Fehler
   1. Ausfälle
   2. Reaktionen auf Fehler

Literatur

- Andrew D. Birrell, Bruce Jay Nelson. Implementing remote procedure calls. ACM Transactions on Computer Systems. Volume 2 , Issue 1 (February 1984) pp 39 – 59.