

Überblick

1. Remote Procedure Call
2. Komponenten beim RPC
3. Fehler

Netzprogrammierung 2. 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>



[1] © Robert Tolksdorf, Berlin

Remote Procedure Calls

[3] © Robert Tolksdorf, Berlin

Remote Procedure Call RPC

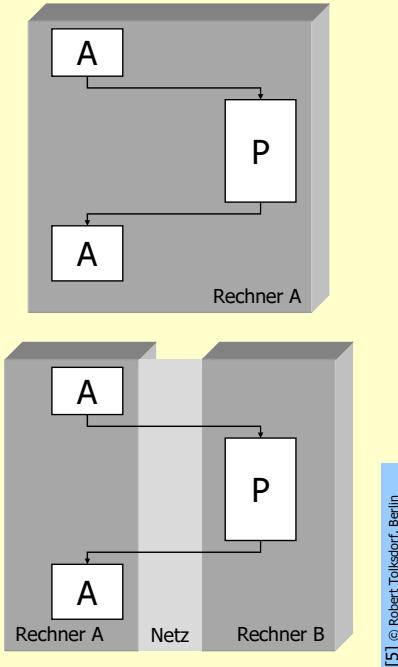
- Entfernter Prozederaufruf, Remote Procedure Call, als grundlegender Mechanismus für verteilte Systeme
 - direkt implementiert (Sun RPC, HP RPC etc.)
 - weiterentwickelt (Java RMI etc.)
 - Konzeptionell enthalten (HTTP etc.)
- [Birrell/Nelson84]:
RPC ist ein synchroner Mechanismus, der Kontrollfluss und Daten als Prozederaufruf zwischen zwei Adressräumen über ein schmalbandiges Netz transferiert

[2] © Robert Tolksdorf, Berlin

[4] © Robert Tolksdorf, Berlin

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 abliefer
- Prozeduraufruf:
Signatur der Prozedur definiert zu übertragende Daten
- Unterschiedlicher Adressraum:
Speicheradressen (Zeiger) sind nicht semantikerverlustfrei übertragbar
- Schmalbandig:
Bandbreite des Netzes ist um Dimensionen geringer als die der Kommunikationspfade innerhalb eines Rechners

[6] © Robert Tollendorf, Berlin

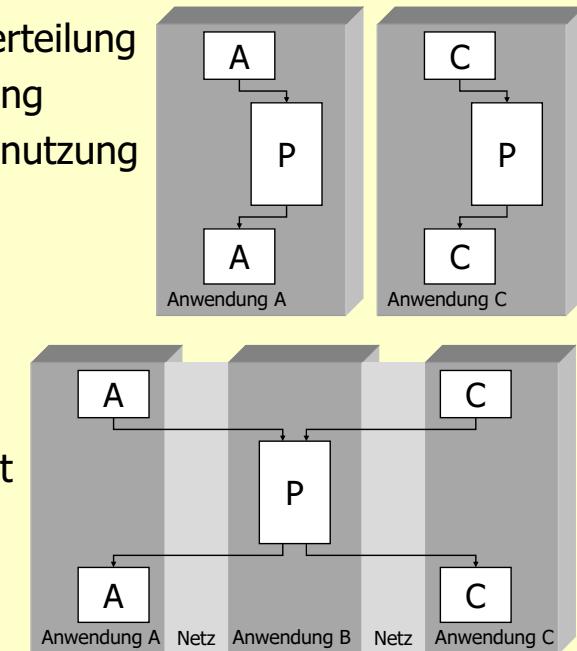
Unterschiede

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

[7] © Robert Tollendorf, Berlin

Vorteile der Netzbasierte Anwendung

- Bessere Aufgabenverteilung
- Bessere Lastverteilung
- Bessere Ressourcennutzung
- Bessere Modularität
- Bessere Wiederverwendbarkeit
- Größere Offenheit
- Besser Integrationsfähigkeit
- ...



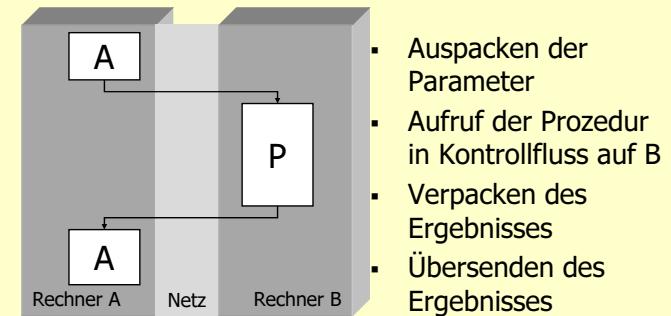
Vorteile der Prozedurabstraktion

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

[9] © Robert Tolksdorf, Berlin

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

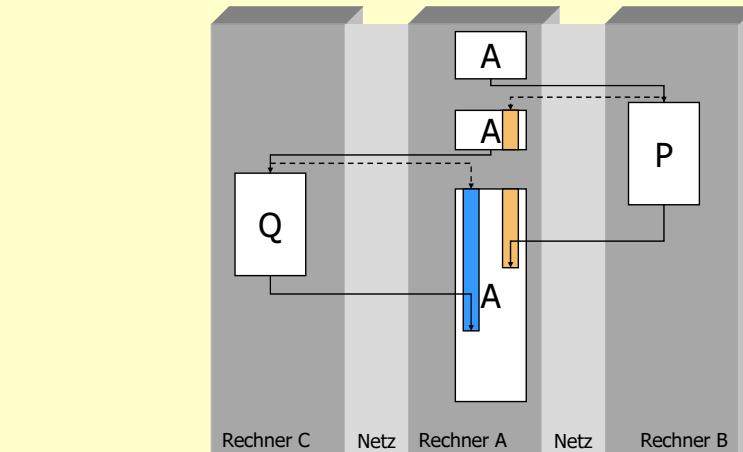
[10] © Robert Tolksdorf, Berlin

Asynchroner RPC

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

[11] © Robert Tolksdorf, Berlin

Asynchroner RPC



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

[12] © Robert Tolksdorf, Berlin

Herausforderungen

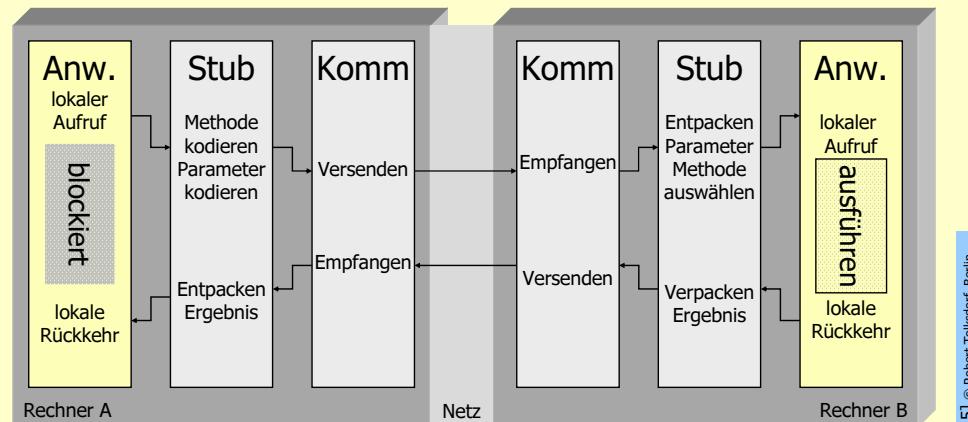
- Semantik im Fehlerfall
- Semantik von Zeigern
- Einbettung in Programmiersprachen
- Auffinden und Binden an die entfernten Prozeduren
- Protokoll des Datenaustauschs
- Eigenschaften der Kommunikation

[13] © Robert Tollendorf, Berlin

Komponenten beim RPC

Komponenten beim RPC

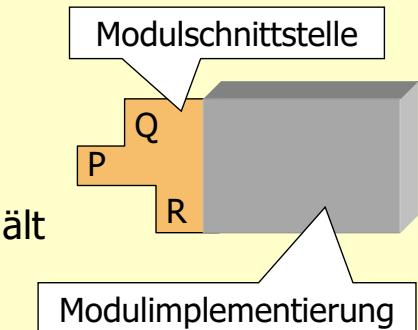
- Anwendungsprozeduren: Eigentliche Arbeit
- Stubs: Ver- und Entpacken von Daten zum Transport
- Kommunikation: Transport von Daten
- Ablauf schematisch:



[15] © Robert Tollendorf, Berlin

Interaktionspunkt: Schnittstellen (lokal)

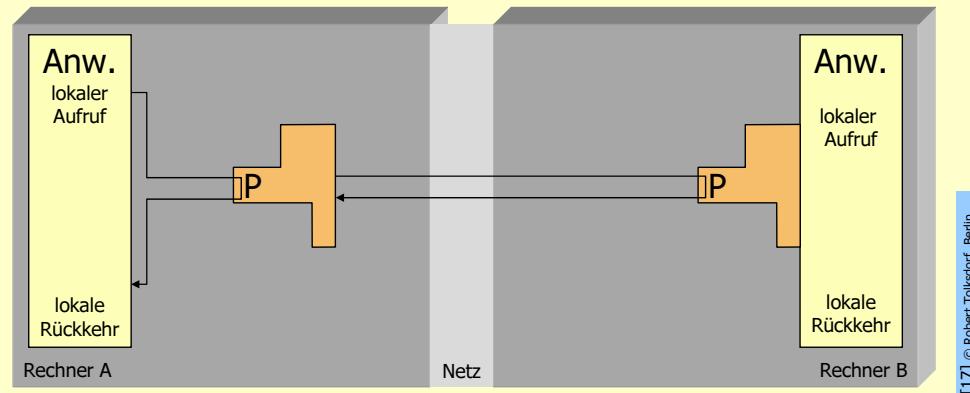
- Ein Modul bietet Prozeduren zum Aufruf an:
Das Modul *exportiert* eine Schnittstelle (Interface)
- Im Beispiel: Schnittstelle enthält Prozeduren P, Q und R
- Definiert durch Signaturen
- Ein anderes Modul ruft diese Prozeduren an:
Das Modul *importiert* eine Schnittstelle



[16] © Robert Tollendorf, Berlin

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 Modulseitegenerieren
- Die Kommunikationskomponente ist generisch und muss lediglich
 - senden
 - empfangenkönnen

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);  
}
```

[17] © Robert Tollendorf, Berlin

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

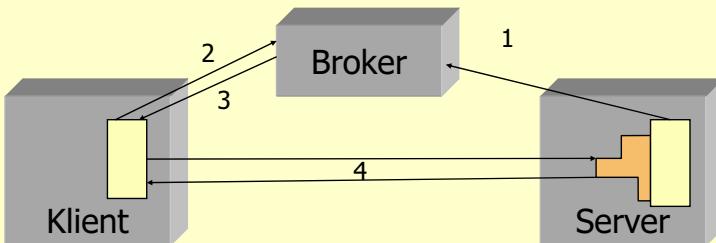
[18] © Robert Tollendorf, Berlin

[19] © Robert Tollendorf, Berlin

[20] © Robert Tollendorf, Berlin

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)



[21] © Robert Tollendorf, Berlin

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
 - ...

[22] © Robert Tollendorf, Berlin

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

[23] © Robert Tollendorf, Berlin

[24] © Robert Tollendorf, Berlin

Fehlerquellen / Ausfälle

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

[25] © Robert Tolksdorf, Berlin

Fehlerquellen

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

[26] © Robert Tolksdorf, Berlin

Semantische Eigenschaften

- Mix aus Fehlerquellen und Reaktion darauf ergibt unterschiedliche semantische Eigenschaften:
 - may-be: RPC wird nicht mehr als 1 Mal ausgeführt (oder 0 Mal oder teilweise)
 - Passend, wenn Klient kein Ergebnis braucht
 - at-least-once: Mindestens 1 Mal ausgeführt
 - at-most-once: 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: Genau 1 Mal
- Meistens: at-least-once oder at-most-once

[27] © Robert Tolksdorf, Berlin

Zusammenfassung

[28] © Robert Tolksdorf, Berlin

Zusammenfassung

1. Remote Procedure Call
 1. Vorteile Verteilung
 2. Unterschiede zum lokalen Prozeduraufruf
2. Komponenten beim RPC
 1. Schnittstellen
 2. Stubs
 3. Bindungen/Broker
3. Fehler
 1. Ausfälle
 2. Reaktionen auf Fehler

[29] © Robert Tolkendorf, Berlin

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.

[30] © Robert Tolkendorf, Berlin