Script **generated by TTT**

Title: Grundlagen_Betriebssysteme (18.01.2012)

Date: Wed Jan 18 13:18:32 CET 2012

Duration: 42:16 min

Pages: 12

Prozesskommunikation

Disjunkte Prozesse, d.h. Prozesse, die völlig isoliert voneinander ablaufen, stellen eher die Ausnahme dar. Häufig finden Wechselwirkungen zwischen den Prozessen statt ⇒ Prozesse interagieren. Die Unterstützung der Prozessinteraktion stellt einen unverzichtbaren Dienst dar.

Fragestellungen

Dieser Abschnitt beschäftigt sich mit den Mechanismen von Rechensystemen zum Austausch von Informationen zwischen Prozessen.

Kommunikationsarten.

nachrichtenbasierte Kommunikation, insbesondere Client-Server-Modell.

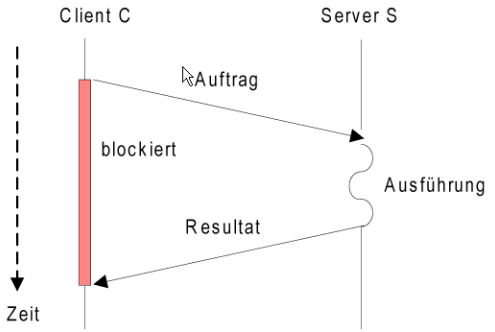
Netzwerkprogrammierung auf der Basis von Ports und Sockets.

Client-Server Modell basiert i.a. auf der Kommunikationsklasse der synchronen Aufträge. ***Server*** stellen Dienste zur Verfügung, die von vorher unbekannten ***Clients*** in Anspruch genommen werden können.

Beispiele für Dienste (Services), die mittels Server realisiert werden: Dateidienst, Zeitdienst, Namensdienst.

ein System kann sowohl Client als auch Server sein.

Client-Server Architektur



Definition: Client

Ein Client ist eine Anwendung, die auf einer Clientmaschine läuft und i.a. einen Auftrag initiiert, und die den geforderten Dienst von einem Server erhält.

Clients sind meist a-priori nicht bekannt.

Definition: Server

Ein Server ist ein Subsystem, das auf einer Servermaschine läuft und einen bestimmten Dienst für a-priori unbekannte Clients zur Verfügung stellt.

Server sind dedizierte Prozesse, die kontinuierlich folgende Schleife abarbeiten.

```
while (true) {
   receive (empfangsport, auftrag);
   führe Auftrag aus und erzeuge Antwortnachricht;
   bestimme sendeport für Antwortnachricht;
   send (sendeport, resultat);
}
```

Client und Server kommunizieren über Nachrichten, wobei beide Systeme auf unterschiedlichen Rechnern ablaufen können und über ein Netz miteinander verbunden sind.

Client-Server-Modell

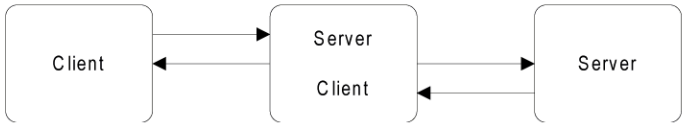
Client-Server Modell basiert i.a. auf der Kommunikationsklasse der synchronen Aufträge. ***Server*** stellen Dienste zur Verfügung, die von vorher unbekannten ***Clients*** in Anspruch genommen werden können.

Client-Server Architektur

Definitionen

Beispiele für Dienste (Services), die mittels Server realisiert werden: Dateidienst, Zeitdienst, Namensdienst.

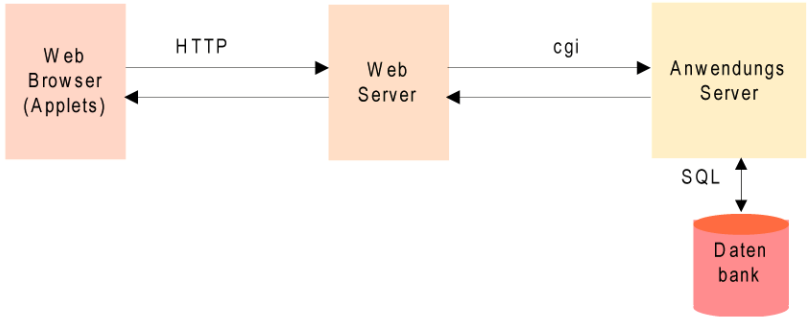
ein System kann sowohl Client als auch Server sein.



Beispiel Multi-Tier

Peer-to-Peer Computing

Beispiel Multi-Tier



Es findet keine Unterscheidung zwischen Client und Server statt

⇒ Systeme übernehmen beide Rollen, d.h. sie sind sowohl Client als auch Server (Peers).

Damit soll der Server-Flaschenhals elimiert werden.

Teilnahme an einem P2P-System erfordert explizites Beitreten.

Feststellung, welche Dienste im P2P-Systeme verfügbar sind

ein Knoten registriert seine Dienste bei einem zentrale Directory-Service.

es existiert ein Discovery-Protokoll, mit Hilfe dessen ein Client-Knoten eine Liste möglicher Dienste im P2P-System erstellen kann.

Beispielsysteme sind Napster und Gnutella.

P2P-Systeme wurde in der Vergangenheit häufig im Zusammenhang mit Musik-Tauschbörsen eingesetzt ⇒ rechtliche Probleme.

Prozesskommunikation

Disjunkte Prozesse, d.h. Prozesse, die völlig isoliert voneinander ablaufen, stellen eher die Ausnahme dar. Häufig finden Wechselwirkungen zwischen den Prozessen statt ⇒ Prozesse interagieren. Die Unterstützung der Prozessinteraktion stellt einen unverzichtbaren Dienst dar.

Fragestellungen

Dieser Abschnitt beschäftigt sich mit den Mechanismen von Rechensystemen zum Austausch von Informationen zwischen Prozessen.

Kommunikationsarten.

nachrichtenbasierte Kommunikation, insbesondere Client-Server-Modell.

Netzwerkprogrammierung auf der Basis von Ports und Sockets.

Netzwerkprogrammierung

Eine ***verteilte Anwendung*** ist eine Anwendung A, dessen Funktionalität in eine Menge von kooperierenden Teilkomponenten A_1 , .., A_n , $n \in IN$, $n > 1$ zerlegt ist;

Jede Teilkomponente umfasst Daten (interner Zustand) und Operationen, die auf den internen Zustand angewendet werden.

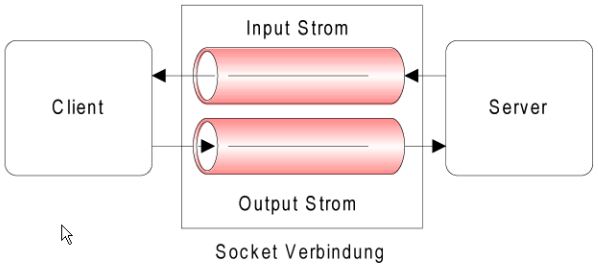
Teilkomponenten A_i sind autonome Prozesse, die auf verschiedenen Rechensystemen ausgeführt werden können. Mehrere Teilkomponenten können demselben Rechensystem zugeordnet werden.

Teilkomponenten A_i tauschen über das Netz untereinander Informationen aus.

Die Teilkomponenten können z.B. auf der Basis des Client-Server Modells realisiert werden.

Einführung

Ein ***Socket*** definiert einen einfachen, bidirektionalen Kommunikationskanal zwischen 2 Rechensystemen, mit Hilfe dessen 2 Prozesse über ein Netz miteinander kommunizieren können.



Sockets abstrahieren von den technischen Details eines Netzes, z.B. Übertragungsmedium, Paketgröße, Paketwiederholung bei Fehlern, Netzadressen.

Ein Socket kombiniert 2 Ströme, einen Input- und einen Output-Strom.

Ein Socket unterstützt die folgenden Basisoperationen:

richte Verbindung zu entferntem Rechner ein ("connect").

sende Daten.

empfange Daten.

schließe Verbindung.

assoziiere Socket mit einem Port.

warte auf eintreffende Daten ("listen").

akzeptiere Verbindungswünsche von entfernten Rechnern (bzgl. assoziiertem Port).

Ein Server kommuniziert mit einer Menge von Clients, die a priori nicht bekannt sind. Ein Server benötigt eine Komponente (z.B. ein Verteiler-Thread), die auf eintreffende Verbindungswünsche reagiert.

Informeller Ablauf aus Serversicht

1. Erzeugen eines SocketServer und Binden an einen bestimmten Port.
2. Warten auf Verbindungswünsche von Clients.
3. Austausch von Daten zwischen Client und Server entsprechend einem wohldefinierten Protokoll (z.B. HTTP).
4. Schließen einer Verbindung (durch Server, durch Client oder durch beide); weiter bei Schritt 2.

Programmstück

```
Socket socket; //reference to socket
ServerSocket port; //the port the server listens to
try {
    port = new ServerSocket(10001, ...);
    socket = port.accept(); //wait for client call
    // communicate with client
    socket.close()
}
catch (IOException e) {e.printStackTrace();}
```

Für das Abhören des Ports kann ein eigener Verteiler-Thread spezifiziert werden; die Bearbeitung übernehmen sogenannte Worker-Threads.