Internetprogrammierung mit `Java` und `C/C++`

Volker Simonis
`simonis@informatik.uni-tuebingen.de`

1. Juli 1997

Zusammenfassung
Dieser Vortrag beschreibt, wie mit standardisierten Sprachen wie `Java` und `C/C++` über Standardprotokolle wie `TCP/IP` und `http` die Möglichkeiten des Internet für eigene Anwendungen genutzt werden können.

Übersicht

1. Standards, Kompatibilität und die normative Kraft des Faktischen
2. `TCP/IP`, `http`, `CGI` und andere Protokolle
3. Das Internet
4. `Java` - die Internetsprache
5. Der Server
   - 5.1 Anforderungen
   - 5.2 Realisierung
6. Der Client
   - 6.1 Anforderungen
   - 6.2 Realisierung

Standards, Kompatibilität und die normative Kraft des Faktischen

Es gibt :

- Kompatibilität auf Maschinenebene (Binärkompatibilität)
- Kompatibilität auf Quellcodeebene
- Kompatibilität auf Protokollebene

und Gremien die Standards für jede Ebene festlegen (z.Bsp. `ANSI`, `ISO`, `IEEE`, `X/Open`, usw ..)

Heutzutage ist die Entwicklung neuer Softwaretechnologien so schnell, daß neu Entwicklungen als sogenannte RFC's (Request For Comments) im Internet publiziert und als neue Standards akzeptiert werden.

TCP/IP, http, CGI **und andere Protokolle**

Schicht	Name	Beispiel
7	Application (Anwendung)	NFS
6	Presentation (Darstellung)	XDR
5	Session (Kommunikation)	RPC
4	Transport	TCP, UDP
3	Network (Vermittlung)	IP, X.25
2	Data link (Sicherung)	Ethernet
1	Physical (Datenleitung)	RS-232

Tabelle 1: Das OSI-Schichtenmodell

- `http`, `nntp` und `smtp` sind auch in Schicht 5 angesiedelt
- Wir entwickeln für unsere Anwendungen eigene Kommunikationsprotokolle in Schicht 5

Das Internet

Globale Vernetzung und Interoperabilität durch Einigung auf Protokollebene (Internetprotokoll `TCP/IP` und `UDP/IP`)

Möglichkeiten für Anwendungsentwickler :

- Weltweite Verfügbarkeit von Ressourcen
- Verteilung von Aufgaben im Netz
- Transparente Kommunikation der Anwendungen

Man kann :

- Eigene Dienste anbieten
- Externe Dienste nutzen

⇒ klassisches Client-Server Modell

Java - **die Internetsprache**

Was macht Java zur Internetsprache :

- "Binärkompatibilität" auf Softwareebene (Bytecode)
- bietet Schnittstelle zu TCP, UDP und http an
- Java-Programme (applets) und Bibliotheken (packages) lassen sich über das Netz laden

Der Server

Anforderungen :

- hohe Verfügbarkeit, Stabilität und Schnelligkeit
- bestmögliche Ausnutzung der zugrundeliegenden Hard- und Sofware

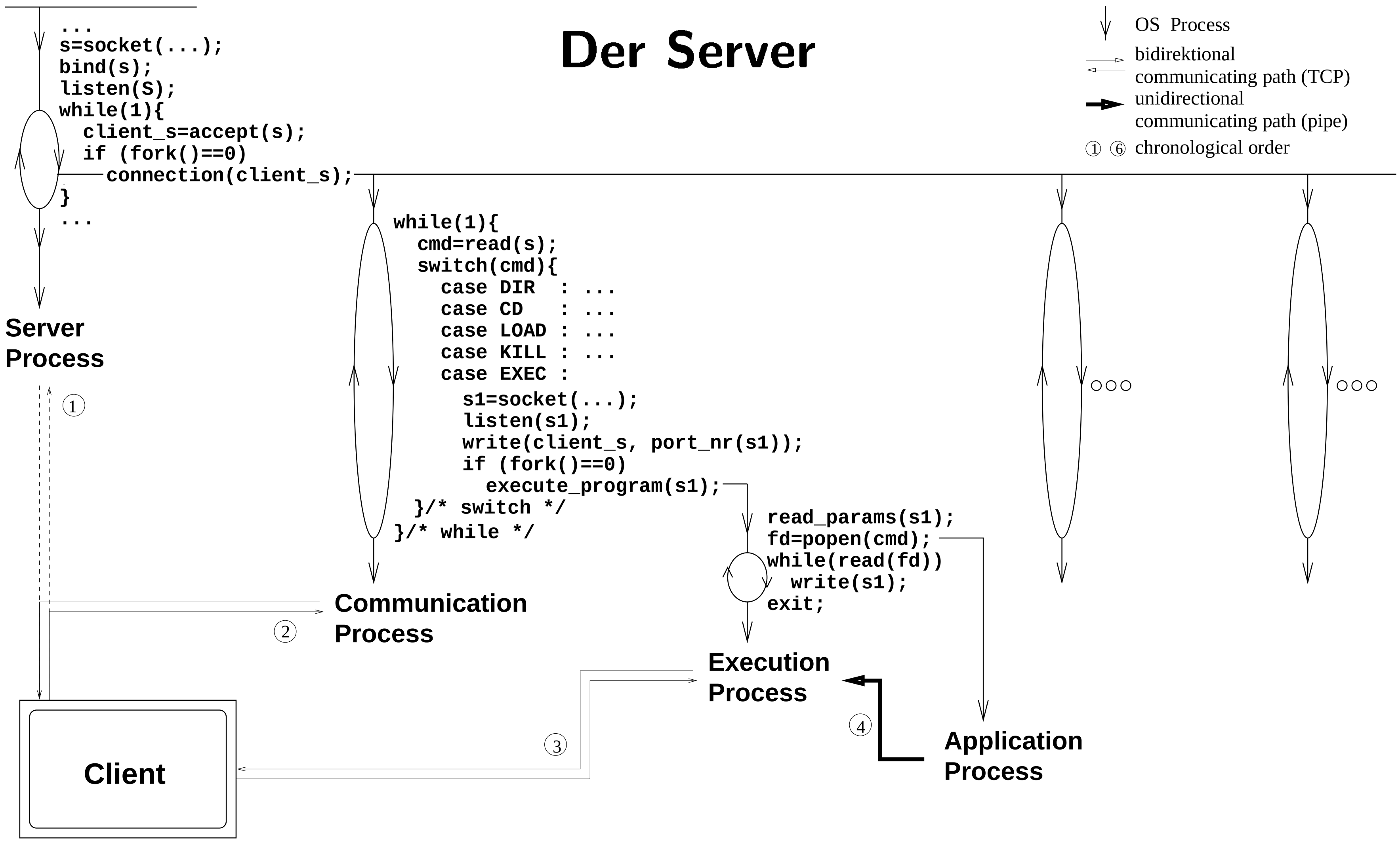
Im Client-Server Modell gibt es :

- einige wenige Server und viele Clients
- Server laufen auf ausgezeichneter Hardware

⇒ Es lohnt sich den Server nativ, z.Bsp. in C++ zu implementieren, denn :

- Geschwindigkeit → optimal
- Integration ins System → optimal
- Portierungsaufwand → vertretbar (normalerweise Quellcodekompatibilität)

Der Server


OS Process
bidirektional
communicating path (TCP)
unidirectional
communicating path (pipe)
① ⑥ chronological order
...
s=socket(...);
bind(s);
listen(S);
while(1){
client_s=accept(s);
if (fork()==0)
connection(client_s);
}
...
Server
Process
while(1){
cmd=read(s);
switch(cmd){
case DIR : ...
case CD : ...
case LOAD : ...
case KILL : ...
case EXEC :
s1=socket(...);
listen(s1);
write(client_s, port_nr(s1));
if (fork()==0)
execute_program(s1);
}/* switch */
}/* while */
Communication
Process
read_params(s1);
fd=popen(cmd);
while(read(fd))
write(s1);
exit;
Execution
Process
Application
Process
Client
1
2
3
4
○○○
○○○

Der Client

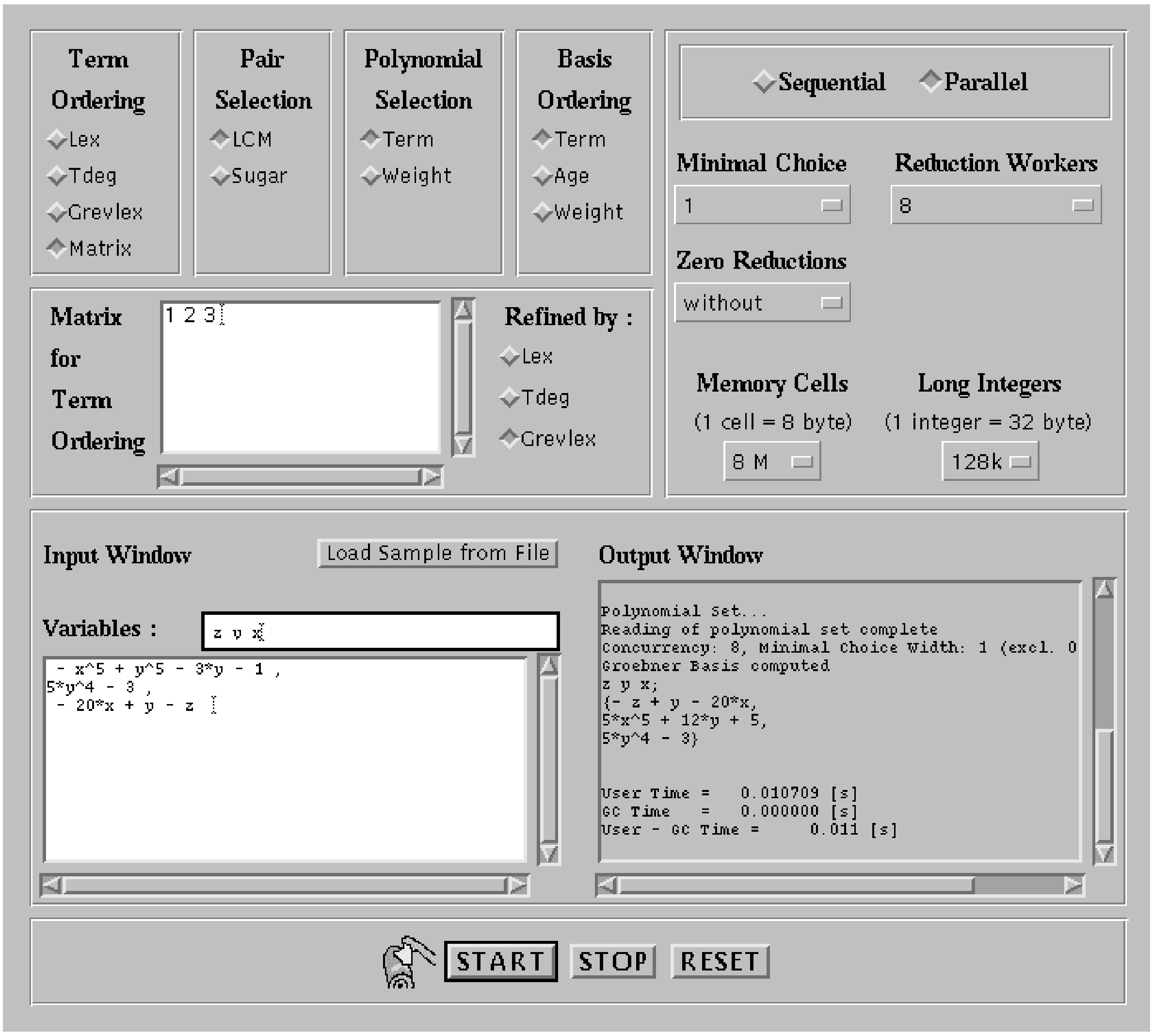
Anforderungen :

- Lauffähigkeit auf allen Plattformen
- gute Wartbarkeit
- einheitliches “Look and Feel”

⇒ Es lohnt sich die Clients in Java zu implementieren, denn :

- Portierungsaufwand → keiner
  (“Binärkompatibilität”)
- Wartungsaufwand → optimal
  (transparente Wartung über das Netz)
- Geschwindigkeitsverlust → vertretbar

Der Client


Term Ordering
Lex
Tdeg
Grevlex
Matrix
Pair Selection
LCM
Sugar
Polynomial Selection
Term
Weight
Basis Ordering
Term
Age
Weight
Sequential
Parallel
Minimal Choice
1
Reduction Workers
8
Zero Reductions
without
Matrix for Term Ordering
1 2 3
Refined by :
Lex
Tdeg
Grevlex
Memory Cells
(1 cell = 8 byte)
8 M
Long Integers
(1 integer = 32 byte)
128k
Input Window
Load Sample from File
Variables :
z y x
- x^5 + y^5 - 3*y - 1 ,
5*y^4 - 3 ,
- 20*x + y - z
Output Window
Polynomial set...
Reading of polynomial set complete
Concurrency: 8, Minimal choice width: 1 (excl. 0
Groebner Basis computed
z y x;
{- z + y - 20*x,
5*x^5 + 12*y + 5,
5*y^4 - 3}
User Time = 0.010709 [s]
GC Time = 0.000000 [s]
User - GC Time = 0.011 [s]
START
STOP
RESET

Der Client

Java Thread

bidirectional communication path (TCP)

① ② chronological order

class GbGui

```
. . .
try{
  sock=new Socket(appletHost,sreverPort);
  in=new DataInputStream(sock);
  out=new PrintStream(sock);
}catch(IOException){
  url=new URL("http://"+appletBase+"/start_server");
  url.println("Start server");
  if (url.getLine()=="OK"){
    sock=new Socket(appletHost,sreverPort);
    in=new DataInputStream(sock);
    out=new PrintStream(sock);
  } else error();
}
animationThread = new AnimationThread();
. . .
handleEvent(..) {
  CD ..
  DIR ..
  LOAD ..
  KILL ..
  EXEC : new ExecThread(cmd, options)
}
. . .
```

main client Thread

①

Server

②

class ExecThread

```
animationThread.resume();
sock=new Socket(appletHost,sreverPort);
in=new DataInputStream(sock);
out=new PrintStream(sock);
out.println(cmd + options);
while(in.getLine() != EOF)
  showResult(...);
animationThread.stop()
```

execution Thread

class AnimThrad

```
animationThread.run();
```

animation Thread