Software Technologies

Mobile Code

Sebastian Fischmeister
fischmeister@softwareresearch.net
University of Salzburg

Course Layout

- consists of two parts: lecture and assignments
- Lecture
  - introduction to mobile code paradigms
  - architecture and components
  - agent systems: grasshoppers & aglets
  - standardization
  - security
- Assignments
  - programming examples in the lab
  - preparing short talks about special topics

Course Layout

- Grading system
  - short talk + literature survey
  - lab assignments
- Material
  - slides of the lecture
  - library introduction
  - D. Lange, M. Oshima. Programming and Deploying Java Mobile Agents with Aglets
  - Grasshoppers, Basics And Concepts 2.2
  - Grasshoppers, Programmers Guide 2.2
  - Grasshoppers, User‘s Guide 2.2
  - website: http://www.softwareresearch.net/site/teaching/SS2004/MC/MC.html

3

Key Aspect

- Mobile code is about „do you move the data or do you move the code“.
- Capability to dynamically change the location of execution

4

Case Study Amoeba

- Timeline:
  - 1970s: Timesharing (1 computer with many users)
  - 1980s: Personal computing (1 computer per user)
  - 1990s: Parallel computing (many computers per user
- Amoeba tried to build a system that reflects the characteristics of an parallel computer or large collection of CPUs shared by a small number of users
  (Tanenbaum, A.S., Renesse, R., Staveren, H., Sharp, G., Mullender S., Jansen, J., Rossum, G.)

5

Case Study Amoeba

- The basic intention is to control a collection of machines based on the pool-of-processors idea.
- Design Goals:
  - Distribution: Connecting together many machines
  - Parallelism: Allowing individual jobs to use multiple CPUs easily
  - Transparency: Having the collection of computers act like a single system
  - Simplicity: Simplicity of the operating system, interprocess communication mechanism
  - Performance: Achieving all of the above in an efficient manner

Utilized process migration !!

6

Case Study Amoeba

- Process Migration
  - Dynamic load balancing (e.g., floating point microcode vs. floating point hardware)
  - Create a port from which a binary image can be fetched.
  - Send descriptor out in the world
  - When new home found, send binary file, delete port and discard the process
  - further reading: process migration survey

7

From OS support to generic

- OS support made assumptions about the environment
  - small-scale networks
  - high bandwidth
  - predictable latency
  - closed system
  - homogenous system
  - connectivity

8

From OS support to generic

- Generic systems need not meet this assumptions, thus
  - heterogenous system
  - connection loss
  - high latency
  - open system => security problems
  - variable bandwidth
- That‘s where mobile agents start.

Seven Good Reasons for Mobile Agents

- Danny Lange’s Seven Good Reasons For Mobile Agents
  - They reduce network load
  - They overcome network latency
  - They encapsulate protocols
  - They execute asynchronously and autonomously
  - They adapt dynamically
  - They are naturally heterogeneous
  - They are robust and fault-tolerant

There is still no killer app for mobile agents!

One more...

- **Wait for events to occur and react!**
  - compex dynamic queries → no more polling
  - ➔ enables proactive applications

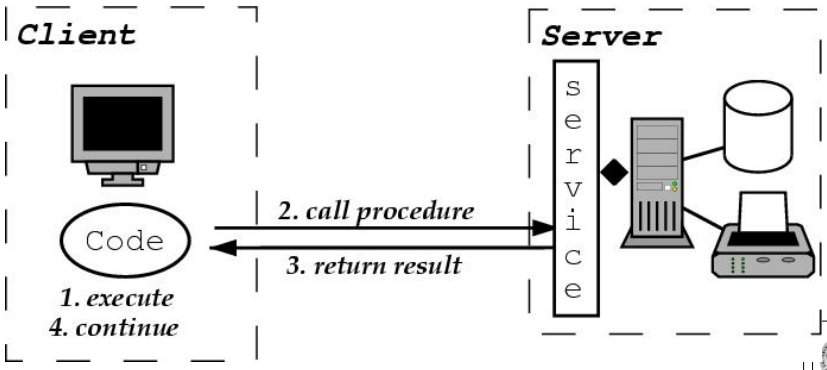
11

Paradigms

- Four basic types:
  - Client/Server
  - Code on Demand
  - Remote Evaluation
  - Mobile Agents
- Elements
  - Data (stored result sets)
  - Code (commands)
  - Program stack (current status of the program)

12

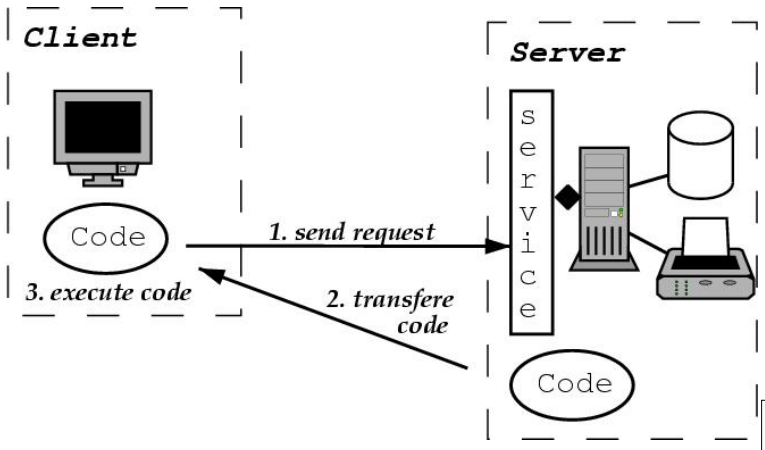
Client/Server



Client/Server Discussion

- Examples: WWW, RPC, Webservices, CORBA, EJBs
- Elements
  - data → mobile
  - code → static
  - program stack → static
- Advantages
  - easy to implement
  - widespread
  - millions of implementations
- Disadvantages
  - there‘s no „one size fits all“

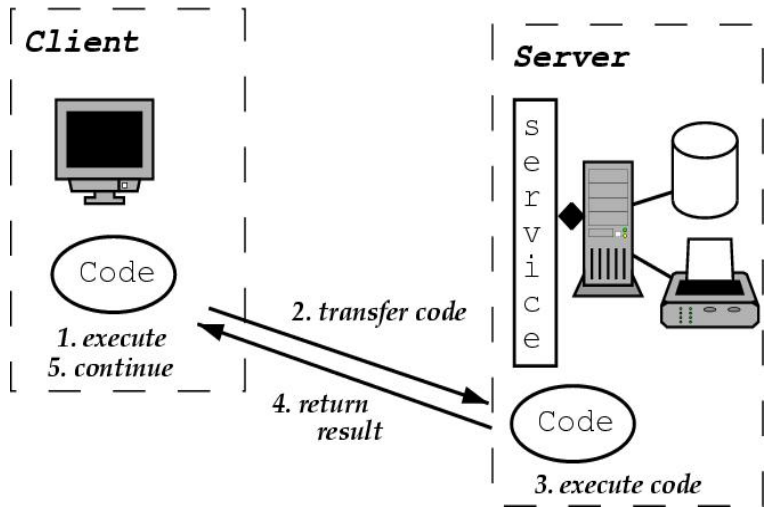
Code on Demand



Code on Demand Discussion

- The idea behind code-on-demand was the thin client or network computer *(created by Larry Allison)*
- Examples: Java Applets
- Elements
  - data → static
  - code → mobile
  - program stack → static
- Advantages
  - centralized codebase
  - simple software update mechanisms
  - dynamic binding ➔ lean software (load help dialog only if activated)
- Disadvantages
  - interoperable code
  - network as single point of failure
  - long delay for start up

Remote Evaluation



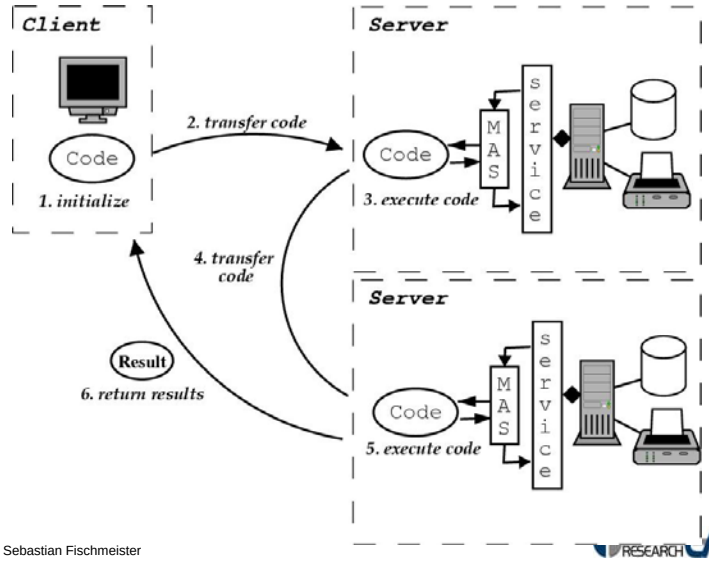
17 © 2004 Sebastian Fischmeister

Remote Evaluation

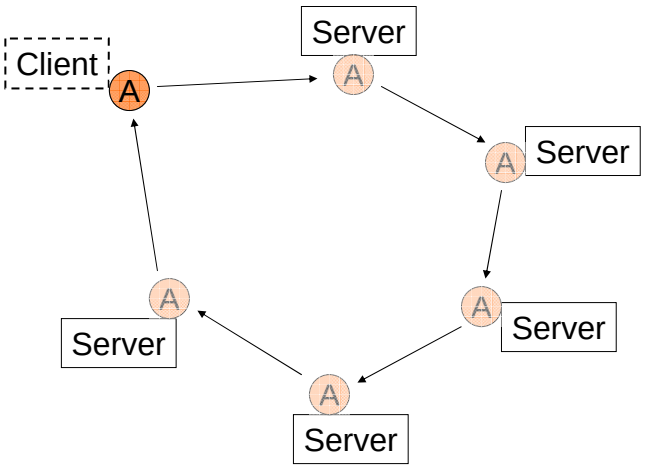
- A prominent example is SQL (to a certain extent), postscript.
- Elements
  - data → static
  - code → mobile
  - program stack → static
- Advantages
  - sometimes better to move the code and not the data (search video database, Postscript)
- Disadvantages
  - difficult to debug
  - security problems
- Case Study: Modern Games

18 © 2004 Sebastian Fischmeister

Mobile Agents



Mobile Agent Example



Mobile Agents Discussion

- Elements
  - data → semi-mobile (necessary data is mobile; semantic compression)
  - code → mobile
  - program stack → mobile

What is a Mobile Agent?

- Program that can migrate from system to system within a network environment
  - Performs some processing at each host
- Agent decides when and where to move next
- How does it move?
  - Save state
  - Transport saved state to next system
  - Resume execution of saved state