Mobile Code Paradigms

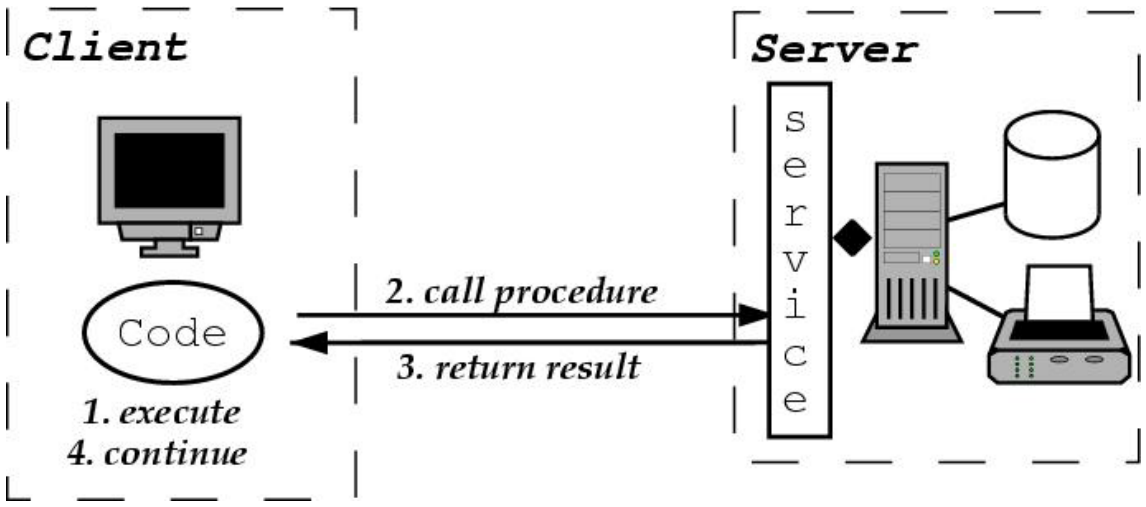
Sebastian Fischmeister

Fischmeister@SoftwareResearch.net

Key Aspect & Elements

- **Mobile code is about „do you move the data or do you move the code“.**
- **Elements**
  - Data (stored result sets)
  - Code (commands)
  - Program stack (current status of the program)

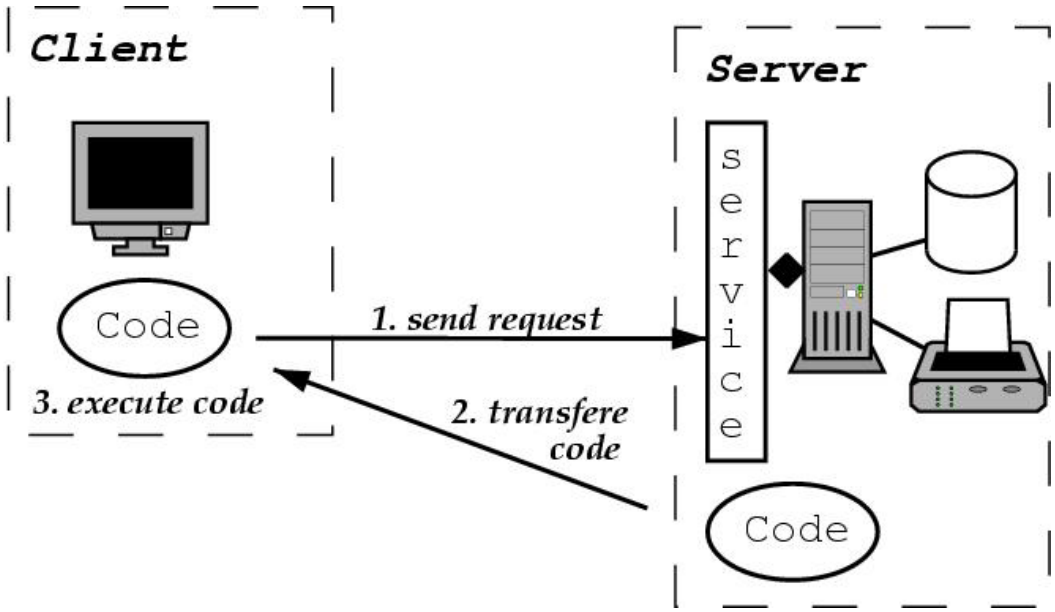
Client/Server



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“

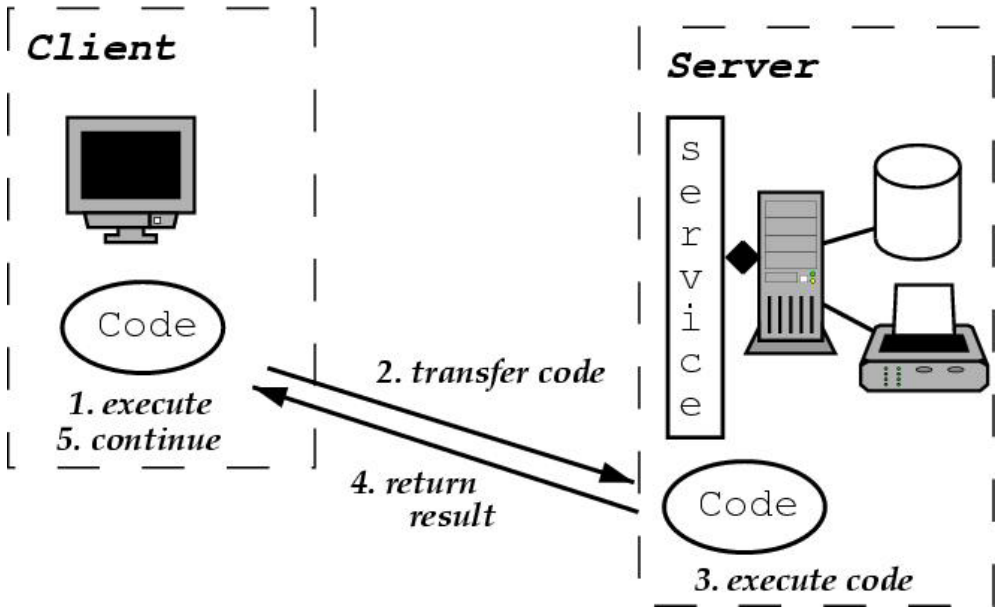
Code on Demand



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

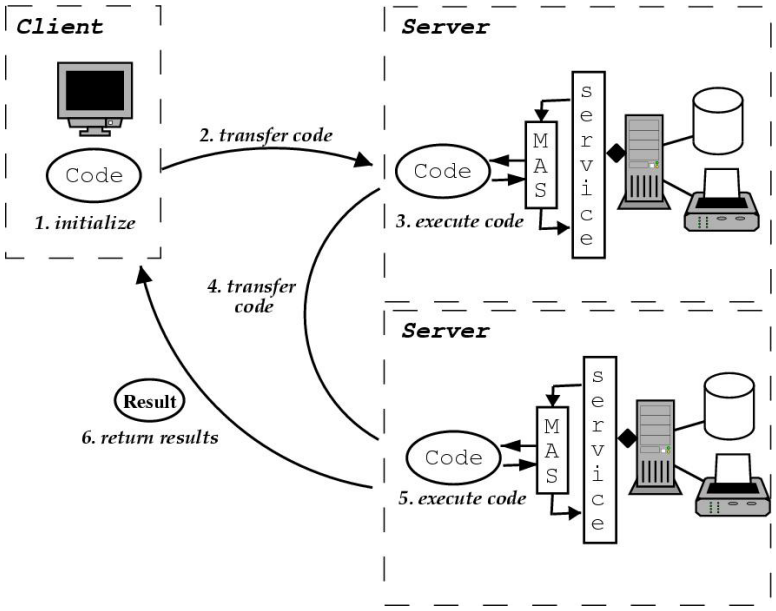
Remote Evaluation



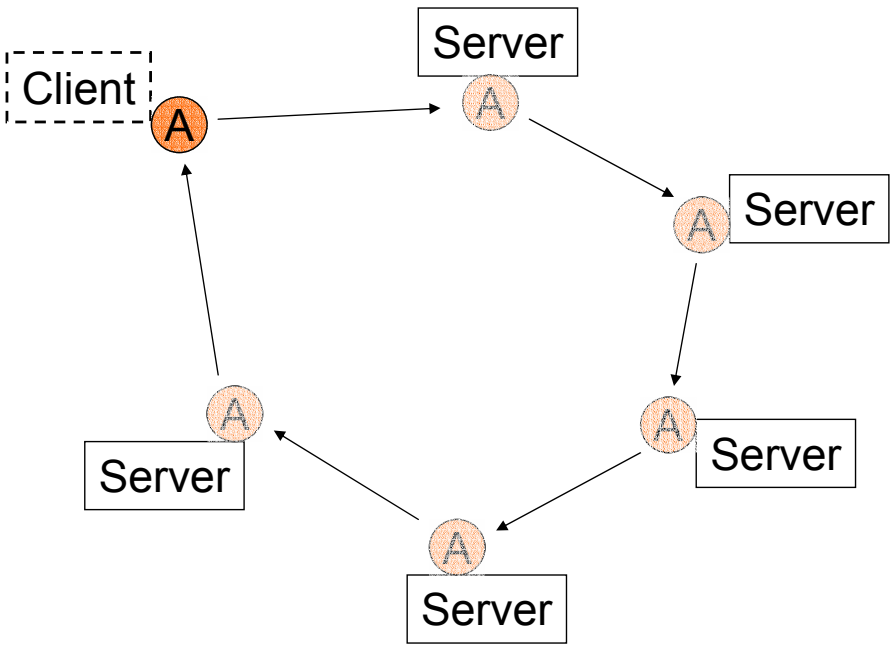
Remote Evaluation

- **A prominent example is SQL (to a certain extent).**
- **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

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

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

Mobile Agent Disadvantages

- Complex to setup
- More complex than client/server
- Everything can also be done with client/server
- Security problems

Mobile Agent Security Problems (I)

- **Masquerading**
  - Agent poses as another agent to gain access to services or data at a host
  - Host assumes false identity in order to lure agents
- **Denial of Service**
  - Agents may attempt to consume or corrupt a hosts resources to preclude other agents from accessing the host's services
  - Hosts can ignore an agent's request for services or access to resources
- **Unauthorized Access**
  - Agents can obtain access to sensitive data by exploiting security weaknesses
  - Agent interferes with another agent to gain access to data

Mobile Agents Security Problems (II)

- **Eavesdropping**
  - With agents that are interpreted, the host can inspect their internal algorithms and data, such as the maximum price the agent's owner is willing to pay for item X
- **Alteration**
  - Hosts can change an agent's internal data or results from previous processing to influence the agent
- **Repudiation**
  - After agreeing to some contract, an agent can subsequently deny that any agreement ever existed or modify the conditions of the contract

Mobile Agent Terms (I)

- **From the OMG MASIF specification**
- Agent
  - An agent is a computer program that **acts autonomously on behalf of a person** or organization. Currently, most agents are programmed in an interpreted language (for example, Tcl and Java) for portability. Each agent has its **own thread of execution** so tasks can be performed on its own initiative.
- Stationary Agent
  - A stationary agent executes only on the system where it begins execution. If the agent needs information that is not on that system, or needs to interact with an agent on a different system, the agent typically uses a communications transport mechanism such as Remote Procedure Calling (RPC). The communication needs of stationary agents are met by current distributed object systems such as CORBA, DCOM, and RMI.

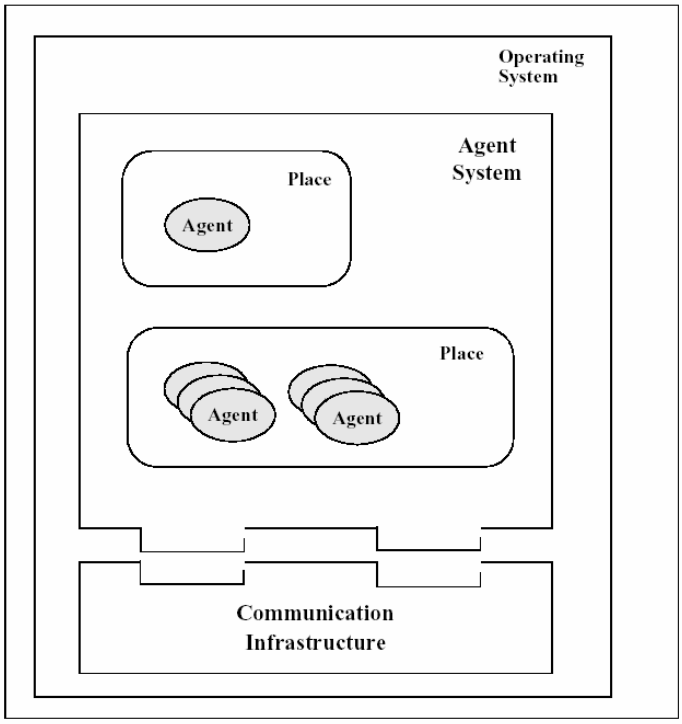
Mobile Agent Terms (II)

- Mobile Agent
  - A mobile agent is not bound to the system where it begins execution. It has the **unique ability to transport itself** from one system in a network to another. This submission is primarily concerned with mobile agents. The ability to travel permits a mobile agent to move to a destination agent system that contains an object with which the agent wants to interact. Moreover, the agent **may utilize the object services** of the destination agent system.
- Agent State
  - When an agent travels, its **state and code are transported with it.** In this context, the agent state can be either it's execution state, or the agent attribute values that determine what to do when execution is resumed at the destination agent system. The **agent attribute values include the agent system state** associated with the agent (e.g. time to live).
- Agent Execution State
  - An agent's execution state is its runtime state, including program counter and frame stacks.

Mobile Agent Terms (III)

- Agent Location
  - The location of an agent is the address of a place. A place resides within an agent system. Therefore, an agent location should contain the name of the agent system where the agent resides and a place name. Note that if the location does not contain a place name, the destination agent system chooses a default place.
- Agent System
  - An agent system is a platform that can **create, interpret, execute, transfer and terminate agents.** Like an agent, an agent system is associated with an authority that identifies the person or organization for whom the agent system acts. For example, an agent system with authority Bob implements Bob's security policies in protecting Bob's resources. An agent system is uniquely identified by its name and address. A host can contain **one or more agent systems.**

Mobile Agent System



Mobile Agent Standardization

- **Object Management Group (OMG) Agents Working Group**
  - Recommends standards for agent technology
  - Mobile Agent System Interoperability Facilities (MASIF) – draft specification
  - www.omg.org
- **FIPA - Foundation For Intelligent Physical Agents**
  - Non-profit organization which promotes the development of specifications of generic agent technologies that maximize interoperability within and across agent-based applications
  - FIPA 98 - seven part specification
  - www.fipa.org
- **New bottom up approaches**

Mobile Agent by Example

```
public class BoomerangAgent extends
de.ikv.grasshopper.agent.MobileAgent {
    // A little data state.
    int state;

    public void live() {
        String location;

        switch(state) {

              // code the states here

        }
         log("Terminating my life.");
    }
}
```


BoomerangAgent (State 0)

```
    case 0:
        log("Waiting for new location...");
        location = JOptionPane.showInputDialog(null,
                                                "Where shall I go?");
        if (location != null) {
             state = 1;
             log("Trying to move...");
             try {
                 // Go away!
                 move(new GrasshopperAddress(location));
             }
             catch (Exception e) {
                 log("Migration failed: ", e);
             }
             // The next statement is only reached
             // if the migration failed!!!
             state = 0;
        }
        break;
```

BoomerangAgent (State 1)

```
case 1:
            log("Arrived at destination!");
            JOptionPane.showMessageDialog(null, "Let me go home!");
            state = 0;
            log("Trying to move...");
            try {
                 // Come home!
                 move(getInfo().getHome());
            }
            catch (Exception e) {
                 log("Return trip failed: ", e);
            }
            // The next statement is only reached
            //if the migration failed!!!
            break;
```


References

- **Grasshopper**
  - http://www.ikv.de
  - http://www.grasshopper.de
- **Aglets**
  - http://aglets.sourceforge.net