Universidade do Minho
Escola de Engenharia
Departamento de Informática

Mestrado Integrado em Engenharia Informática
Mestrado em Engenharia Informática
Sistemas Inteligentes
2020/2021

Filipe Gonçalves, Paulo Novais, César Analide

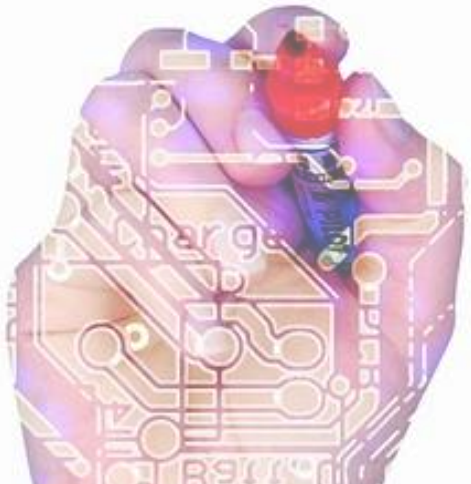
ISLab
Synthetic Intelligence Lab

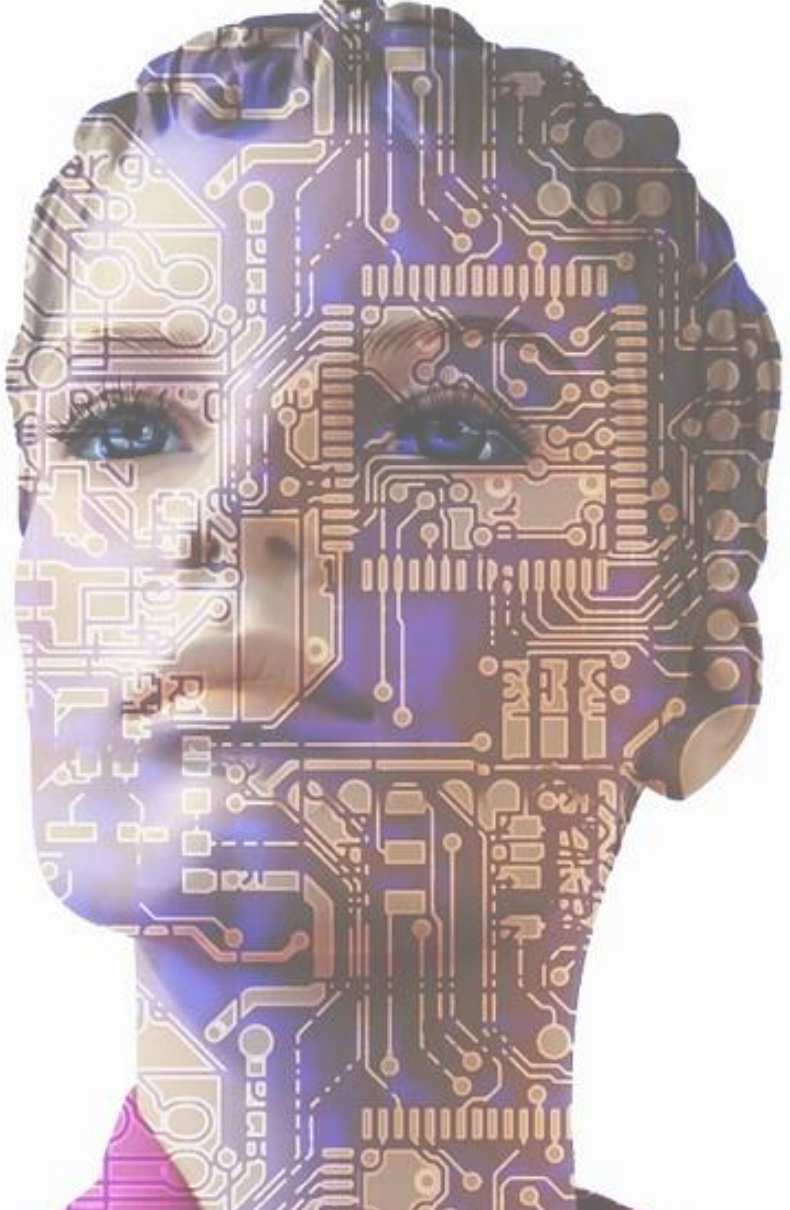
Agentes Inteligentes @ 2020/2021

- Paulo Novais – pjon@di.uminho.pt
- César Analide – analide@di.uminho.pt
- Filipe Gonçalves – fgoncalves@algoritmi.uminho.pt

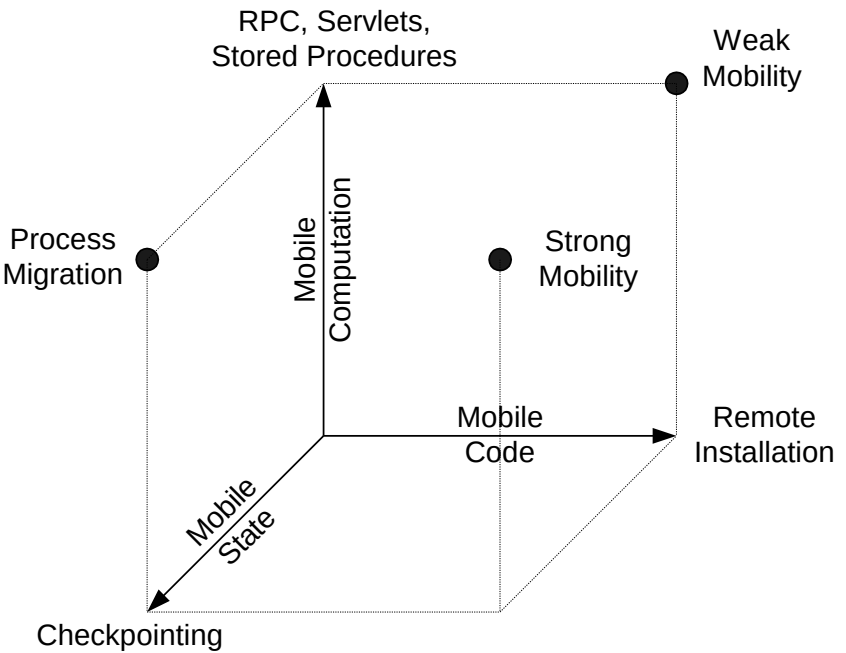
- Departamento de Informática
  Escola de Engenharia
  Universidade do Minho
- ISLab – (Synthetic Intelligence Lab)
- Centro ALGORITMI
  Universidade do Minho

Mobile Agents



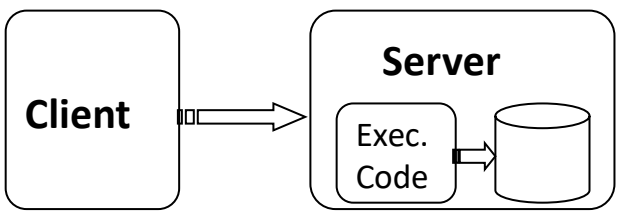


- **Mobile agents** are defined as active objects (or clusters of objects) that have behaviors, state and location.
  - **Mobility:** *Agents* that can travel in network
  - **Autonomy:** Agent itself decides when and where to migrate next
- **Opposite**:
  - **Stationary agent**
  - **Mobile code** (Applets)

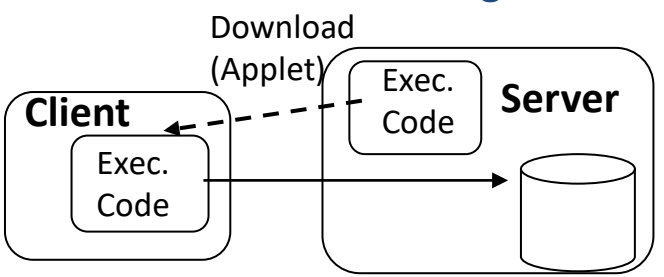


Network Computing Paradigm Comparison

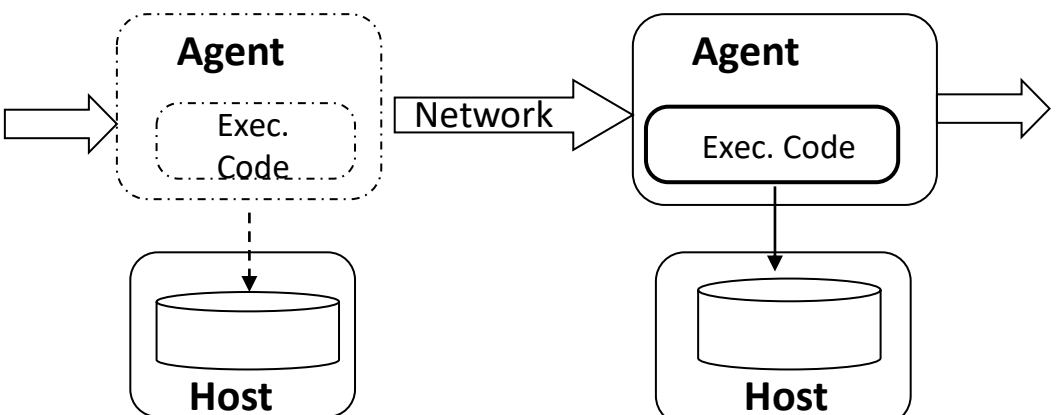
Client-Server Paradigm



Code-on-Demand Paradigm



Mobile Agent Paradigm



- A mobile agent contains the following 3 components:
  - ***Code***: the program that defines the agent's behavior
  - ***State***: the agent's internal variables which enable it to resume its activities after moving to another host
  - ***Attributes***: information describing the agent, its origin and owner, its movement history, resource requirements, authentication keys, etc. Part of this may be accessible to the agent itself, but the agent must not be able to modify the attributes

Mobile Agents Events:

- ***Creation:*** a brand new agent is born and its state is initialized
- ***Dispatch:*** an agent travels to a new host
- ***Cloning:*** a twin agent is born and the current state of the original is duplicated in the clone
- ***Deactivation***: an agent is put to sleep and its state is saved in persistent storage
- ***Activation***: a deactivated agent is brought back to life and its state is restored from persistent storage
- ***Retraction***: an agent is brought back from a remote host along with its state to the home machine
- ***Disposal***: an agent is terminated and its state is lost forever
- ***Communication***: notifies the agent to handle messages incoming from other agents , which is the primary means of inter-agent correspondence

MAS Architecture:

- ***Agent Execution Support***: provided the capability to create mobile agents, taking into account agent-specific requirements regarding the runtime environment
- ***Management Support***: **monitors** and **controls** the agents. It controls aspect comprises among others the temporary interruption of an agent's task execution, its premature termination, or the modification of its task
- ***Security Support***: **manages** the **privacy and integrity** of agents. Comprises the encryption and decryption of agents during migration, the authentication and authorization of agents and agent systems, and access control regarding the resources of an agent system
- ***Mobility Support***: supports remote execution as well as migration
- ***Unique Identification of Agents Support***: handles the unique agent identifiers of each agent
- ***Transaction Support***: support for correct and reliable execution of agents in presence of concurrency and occurrence of failures
- ***Communication Support***: handles the communications mechanisms between agents

Object Mobility:

- **Strong Mobility**
  - Migration of agent code, data and execution state
- **Weak Mobility**
  - Migration of only the agent code and data
- **Strong mobility** is difficult to accomplish!
  - If the agent code is interpreted, access to the execution state is difficult to obtain
  - If the agent code is compiled before execution, the execution state is represented by the stack. Transporting the stack and rebuilding it on a different host, which can be an entirely different architecture, is a nontrivial task

Mobile Agent Support Languages:

- The first language for programming *mobile agents* called TeleScript was introduced in 1994
- Mobile agent could be implement by various languages
- Java language is the most popular since it presents:
  - **Platform independence**
  - **Object-oriented feature**
  - **Security model**
  - Create once, go everywhere

Security Issues:

- **Security and resource control** are crucial when agents executing on a local machine come from foreign hosts and when agents cross organizational boundaries.
- **Security issues** must be dealt with to **encourage** the uptake of mobile agent
- **Host security**:
  - protecting the host against malicious agents
- **Agent security**:
  - protecting the agents against malicious host
- **Security Steps**:
  - **Authentication** of user
  - Determination of whether the user has the **authorization** to execute
  - Determination of agents ability to **pay for the service** provided by the server

Trusted and Untrusted Mobile Agent:

- Some MA platforms treat trusted and untrusted agent differently
- Untrusted agents are prohibited to run dangerous command

- **Example:**
  - An agent roams around the Internet to look for the lowest price of a air ticket; it remembers the lowest price it finds most recently
  - ***Data tampering***: change of execution state of agents by malicious hosts ("brain-flush" the agent of the lowest price it remembers)
  - **Solution**:
    - **Agent tampering detection** (e.g. cryptographic watermarks)
    - **Prevention** (e.g. execution of encrypted functions)

Virus Handling:

- **One Solution:** make the system **Turing-complete** (system in which a program can be written that will find an answer, with no guarantees regarding runtime, permissions or memory)

- Consider a mobile agent only have the following **limited power**:
  - Alter its own internal state variables
  - Make database queries in the current server
  - Move to another server
  - Send text message back to its owner

Mobile Agent Standardization:

- There are various mobile agent system available such as **AgentTcl** , **Aglets**, **MOA**, **Grasshopper** and **Odyssey.** These differ widely in architecture and implementation

- **Standardization effort** include:
  - ***Interoperability***: developing a **common Agent Communication Languages** to improve interoperability between agents, and between agents and mobile agent middleware
  - ***Execution performance***: execution performance is often sacrificed in order to achieve portability via code interpretation. Justin time compilation is one way towards better performance
  - ***Robustness***: Adding robustness involves fault tolerance schemes (e.g., coping with hosts crashing) and is key for trust and acceptance of the technology in companies

Mobile Agent vs RPC:

Remote Procedure Call (RPC):

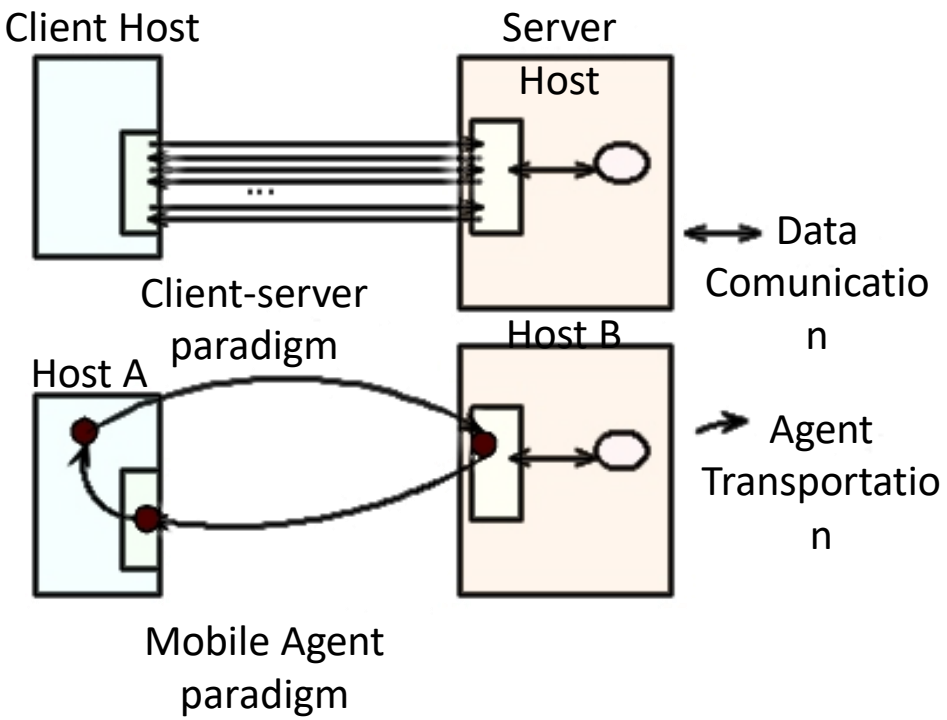
- **Synchronous protocol** - High Efficiency and low latency
- Scaling: Multiple servers

Mobile Agent:

- **Asynchronous Messaging** - Robustness especially in wide area network
- Every node is a server

Competing Technologies:

- Message passing systems
- Remote Method Invocation (RMI)
- CORBA



Application Areas for Mobile Agents:

- Data collection from many places
- Searching and filtering
- Monitoring
- Negotiation
- E-commerce
- Parallel Computing
- Entertainment
- Targeted information dissemination

JADE Mobility Support

- void doMove(Location destination)
  - Location can be the identifier of ContainerID or PlatformID;
- void beforeMove()
  - Last instruction to be executed before agent moves;
- void afterMove()
  - First instruction to be executed after the agent move is completed;
- void doClone(Location destination, String name)
  - Similar to doMove(), but maintains original copy in container
- void beforeClone()
  - Last instruction to be executed before agent clones;
- void afterClone()
  - First instruction to be executed after the agent clone is created;

JADE Mobility Support

```
// Create variables
String containerName = “Container1”;
ContainerID destination = new ContainerID();

// Init destination object
destination.setName(containerName);

// Change of agent state to move
myAgent.doMove(destination);
```

```
// Create variables
String containerName = “Container1”;
String newAgentName = “cloneAgent”;
ContainerID destination = new ContainerID();

// Init destination object
destination.setName(containerName);

// Change of agent state to clone
myAgent.doClone(destination, newAgentName);
```

Universidade do Minho
Escola de Engenharia
Departamento de Informática

Mestrado Integrado em Engenharia Informática
Mestrado em Engenharia Informática
Sistemas Inteligentes
2020/2021

Filipe Gonçalves, Paulo Novais, César Analide