Unit-IV

Mobile Agent

Mobile agents

- Introduction of Mobile agents
- Mobile agents characteristics
- requirement for Mobile Agent system
- Platform for Mobile agents
- Aglet object Model
- Agent Tcl architecture

Introduction of Mobile agents

- S/W program which travels from one platform to other platform in order to get their work done
- Carries its data & state with itself and resume its execution from the state it had left on the previous platform

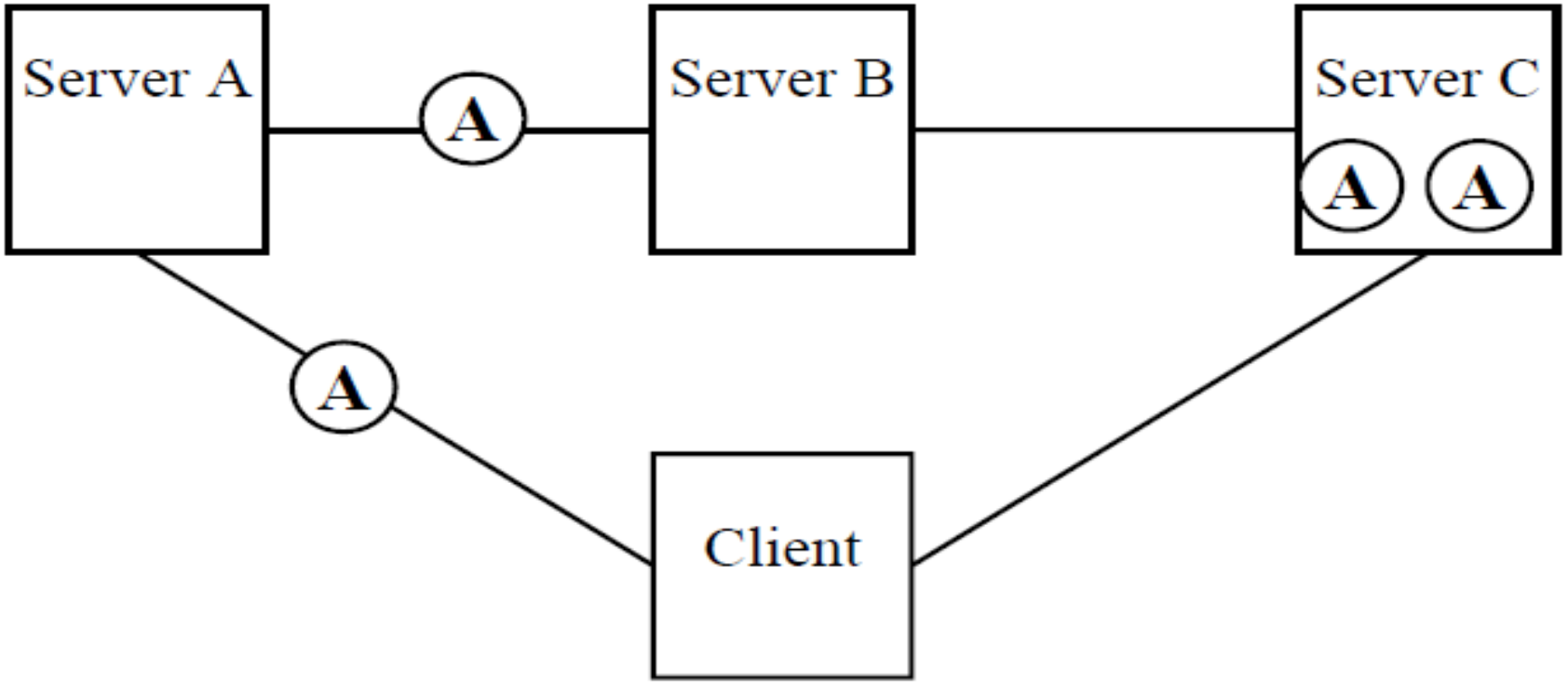
What is a Mobile Agent?

- Program that can migrate from system to system within a network environment to improve performance
- 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

A Mobile Agent Dissected

- A mobile agent contains the following 3 components:
  - **Code** - the program (in a suitable language) that defines the agent's behavior.
  - **State** - the agent's internal variables etc., 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



- *Ability to move from host to host,*
- *Executes at each place and then keeping the results before moving to the next server.*

Mobile Code Systems: Design

- **Four basic types**:(behavioral aspect)

  Client/Server - code do not move at all

  Code on Demand - code downloaded from distant site & executed on local m/c

  Remote Evaluation - code sent to other site for exe. & result sent back

  Mobile Agents – small programs that wander around in n/w

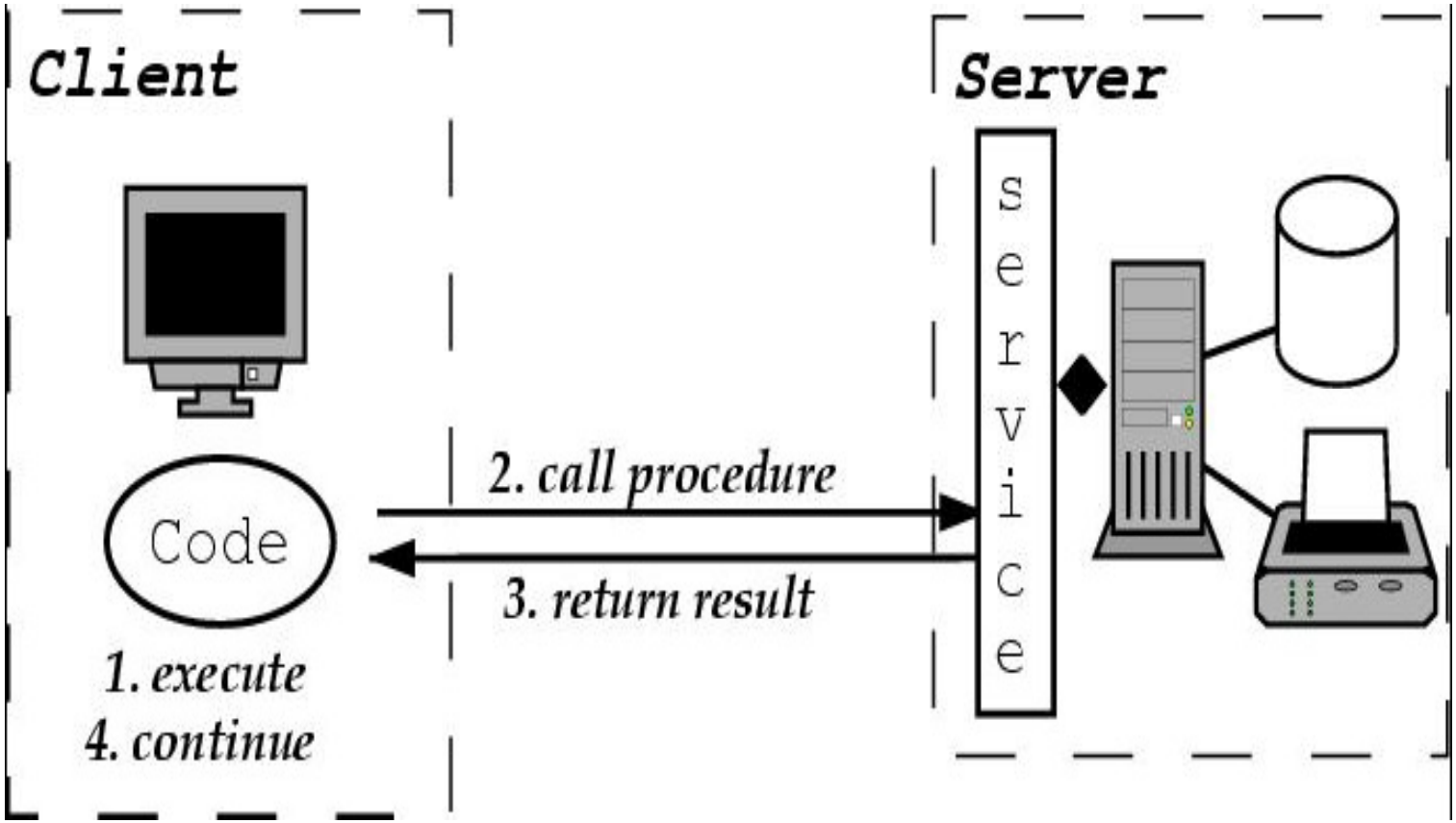
- **Elements**

  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

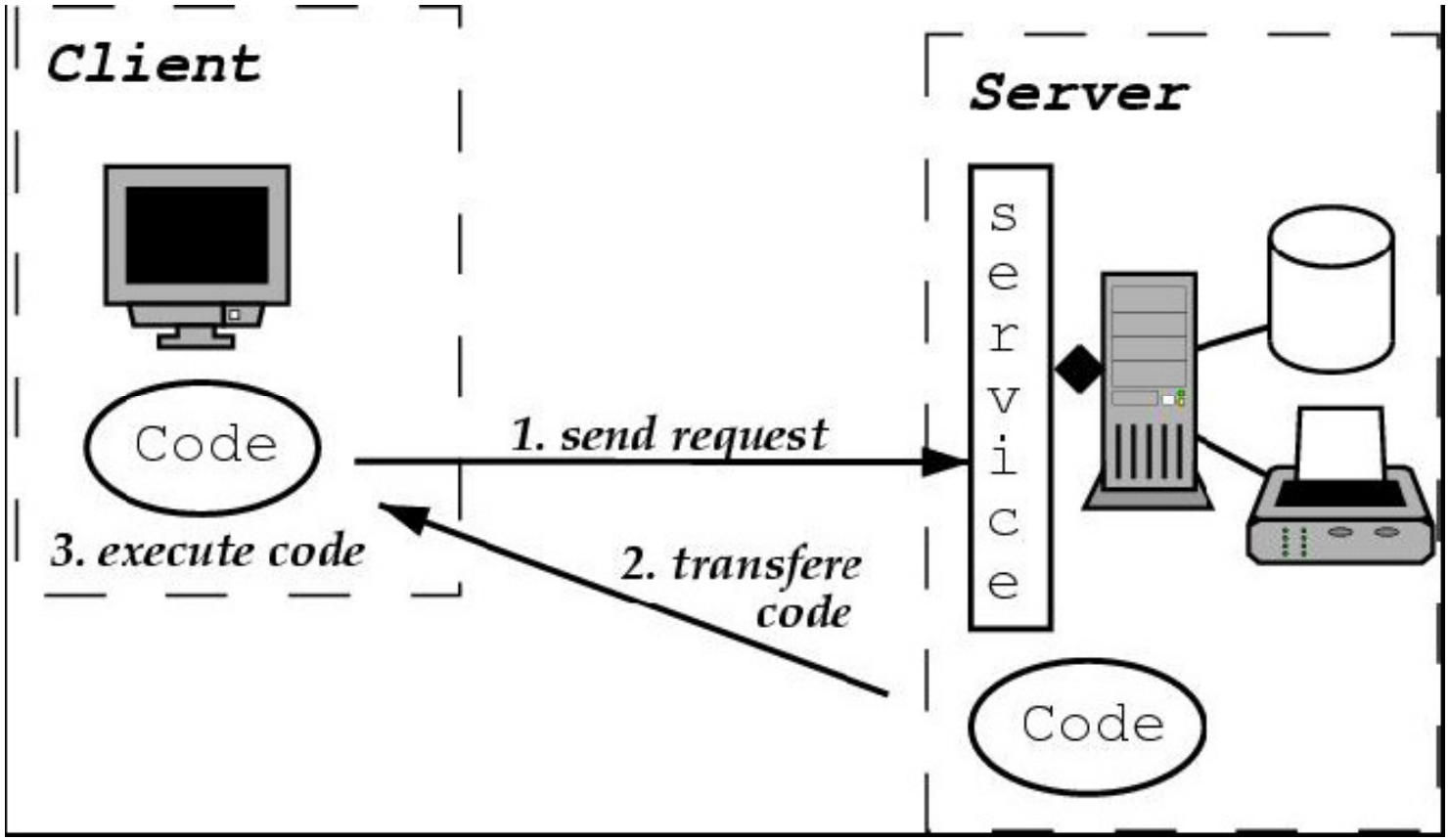
- **Disadvantages**

  Create bandwidth and network traffic problems for large volume of data

  There‘s no “one size fits all“

CORBA (Common Object Request Broker Architecture), EJB (Enterprise JavaBeans)

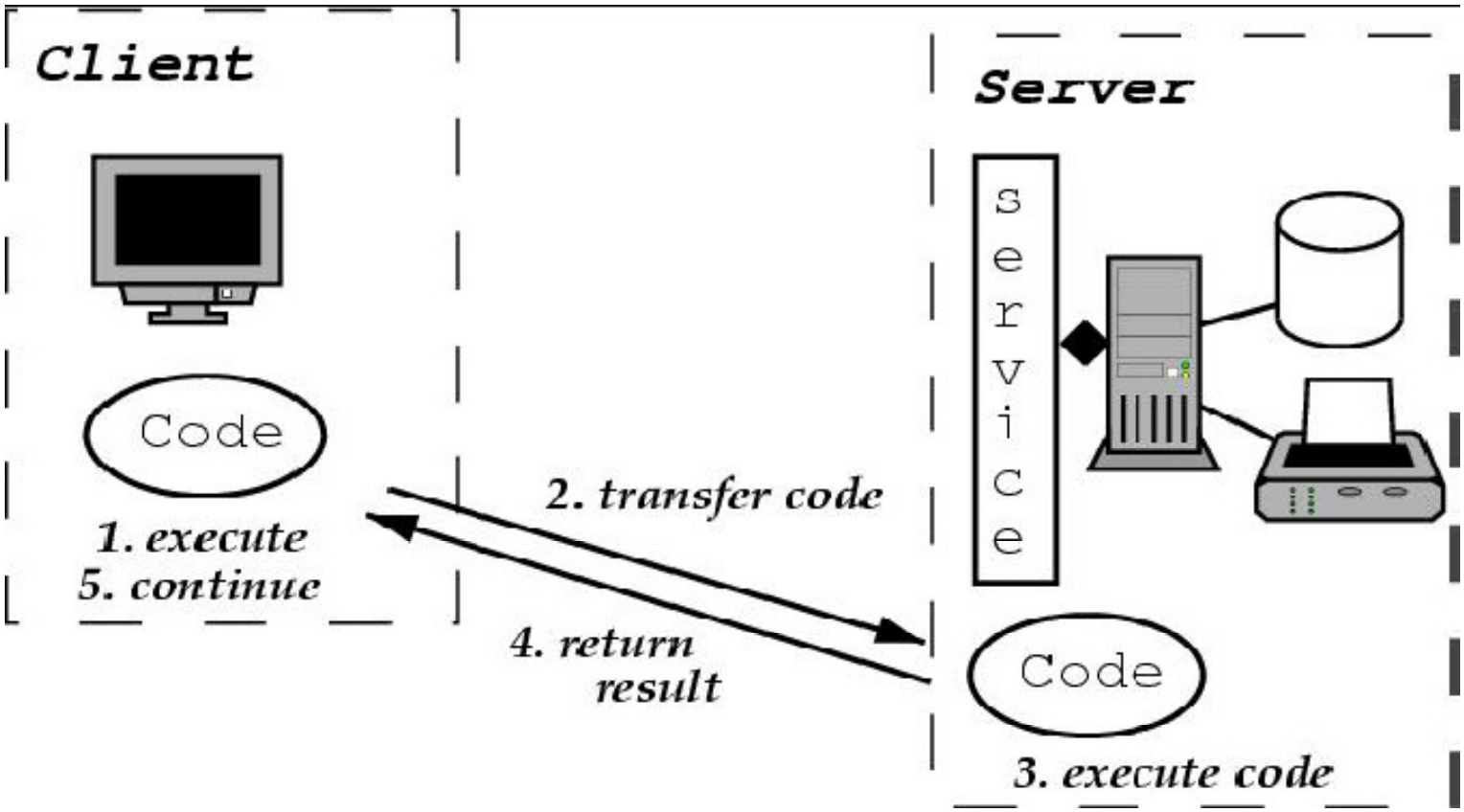
Code on Demand



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
- **Disadvantages**
  - network as single point of failure
  - long delay for start up

Remote Evaluation



Remote Evaluation

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

  data - static

  code - mobile

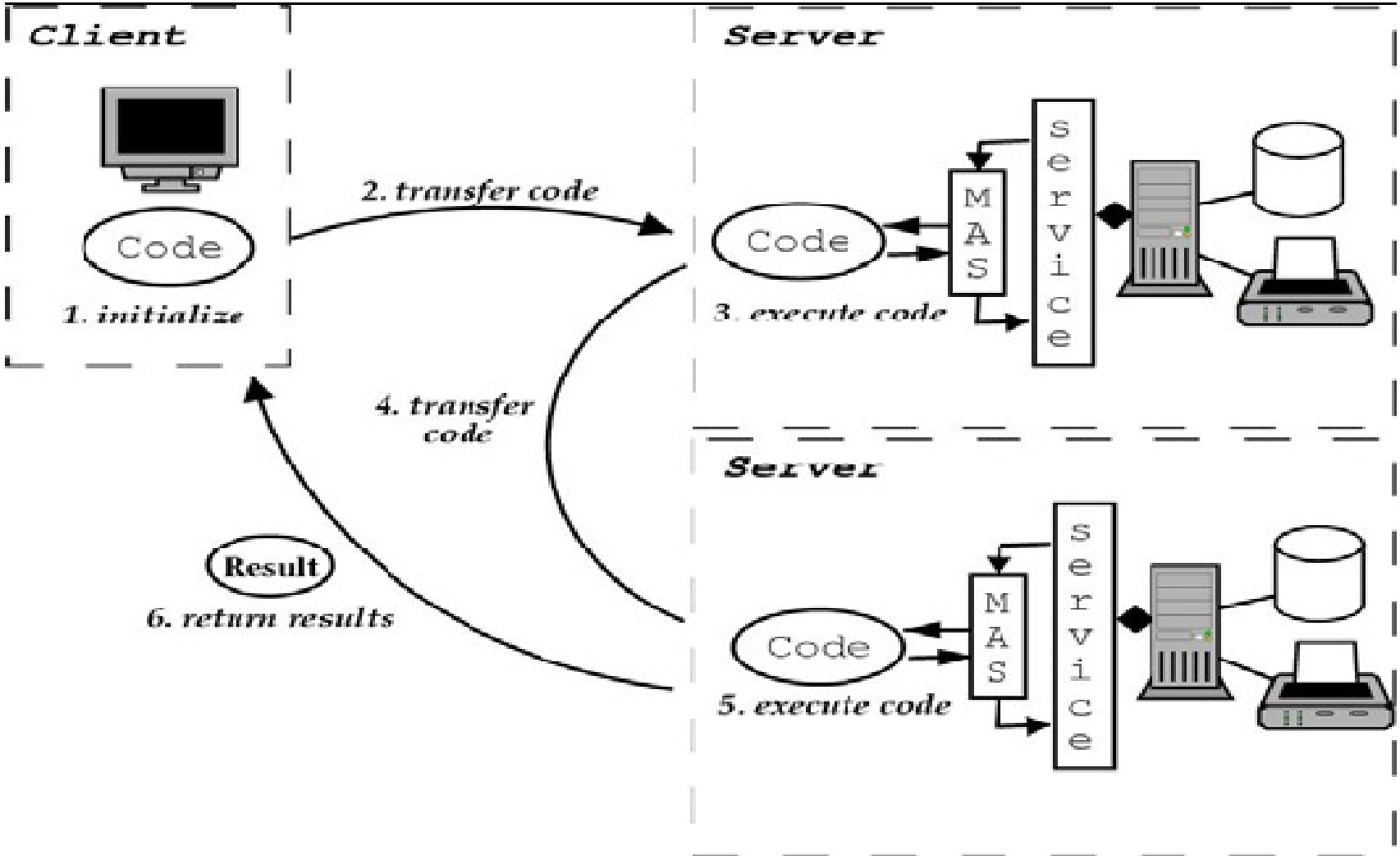
  program stack - static
- **Advantages**

  sometimes better to move the code and not the data (search video database, Postscript)
- **Disadvantages**

  difficult to debug

  security problems

Mobile Agents



Mobile Agents Discussion

- **Elements**

  data - semi-mobile (necessary data is mobile)

  code - mobile

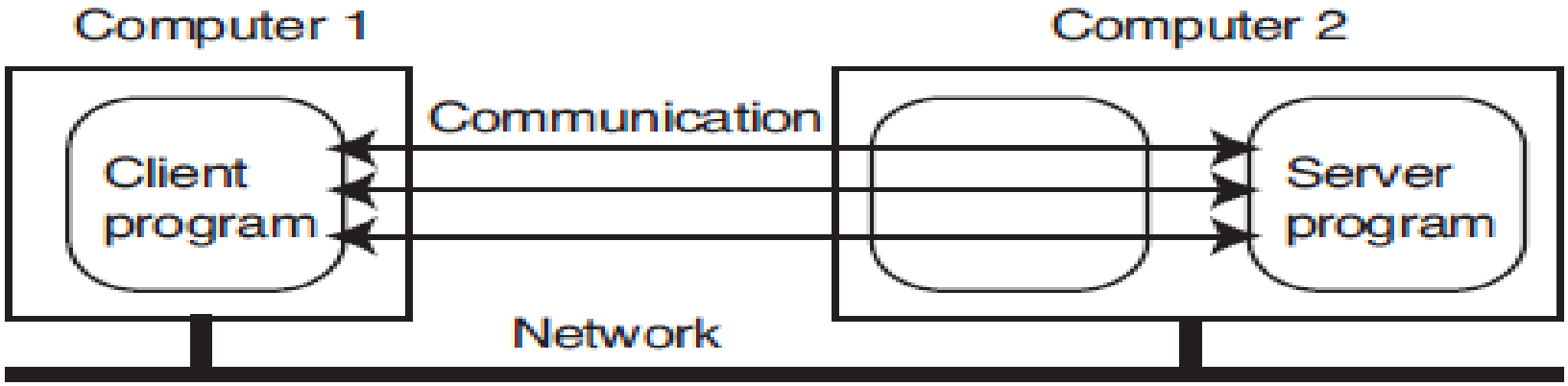
  program stack - mobile

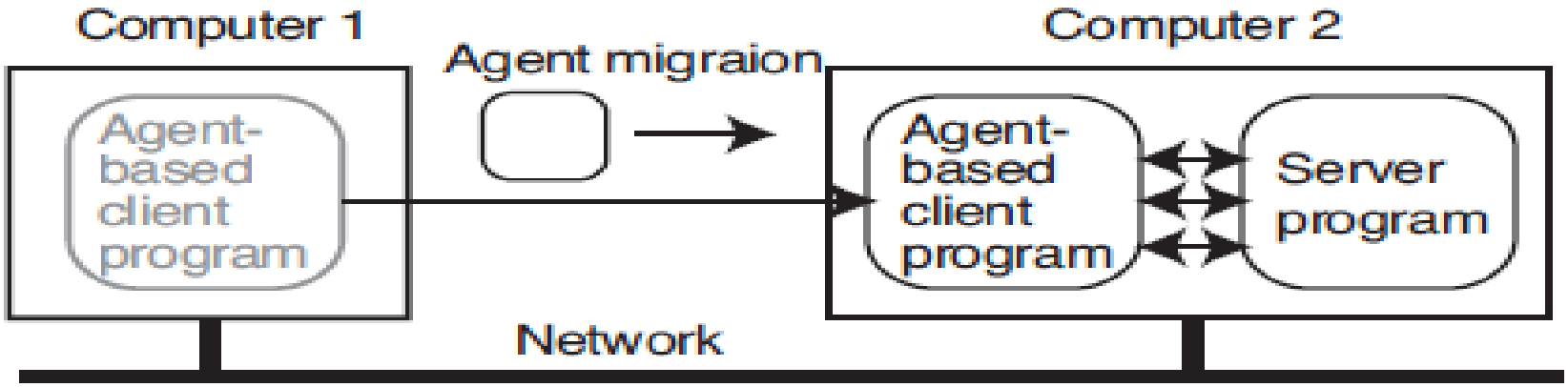
Mobile Agent Advantages

- They reduce the network load by sending the code to the data host instead of sending the data over the network
- They overcome network latency in real-time systems because they do not need a lot of bandwidth.
- They encapsulate protocols by bringing their own protocol code with them.

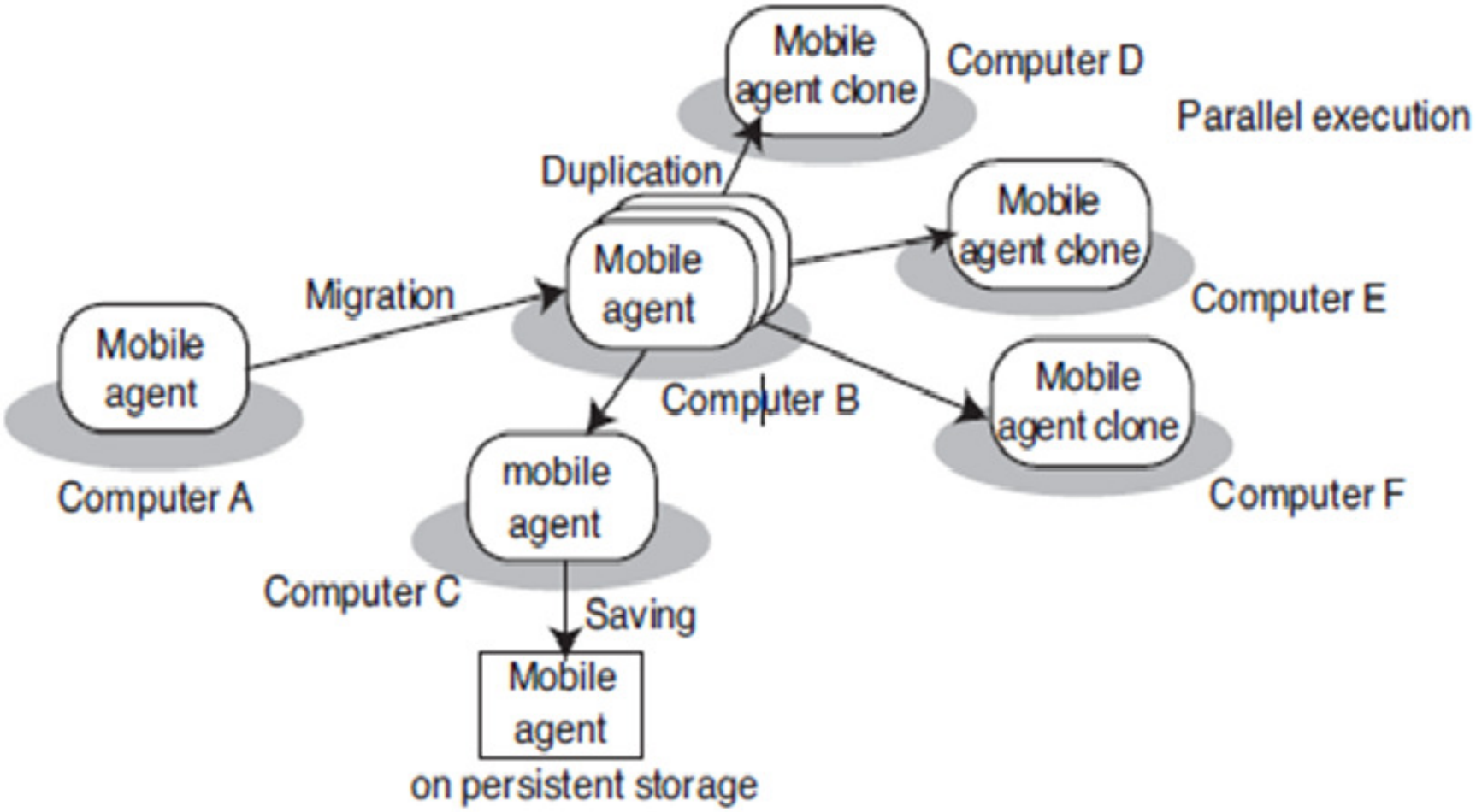
Mobile Agent Advantages

- They execute asynchronously and autonomously, and are therefore not dependent on a continuously open network connection.
- They adapt dynamically to new environments
- They are naturally heterogeneous, and work well in a heterogeneous network.
- They are robust and fault tolerant. If a host shuts down, the agents can move to another host. They can also duplicate themselves and execute on several hosts in parallel.





Reduced communication cost

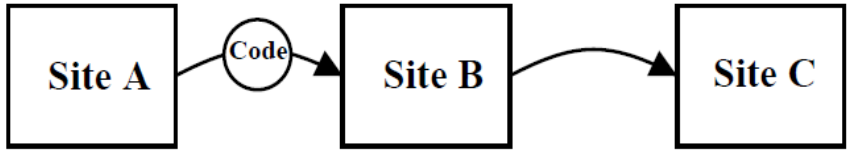


Functions of mobile agents in distributed system

Properties of Agents

- Autonomy- Autonomous and reactive
  - decides which message to respond
  - decides on its own
  - always listens and react to situations

  Ex. Automatic prg. Updates – connects to server,downloads new code,updates itself
- Intelligent functionality
  - more functional & intelligent
  - exhibits complex reasoning using inference engine / neural network
- Agent communication language
- Mobility
- Executability- level of executability text-user,marks-instructions(interpreted) macros,controls,scripts,bytecode,machine code



Usage of agents

- User assistance agents - used at user level to provide services to users, offer info./give advice to users. eg. – animated help characters used in Microsoft office
- Organizational structure agents

  - emitate human organization where agents performs various role like buyer, supplier etc in chain mgmt system.
- Network & system mgmt agents -eg. Like load balancing, failure anticipation etc.
- Interest matching agents - Helpful in commercial sites

Levels of Mobility

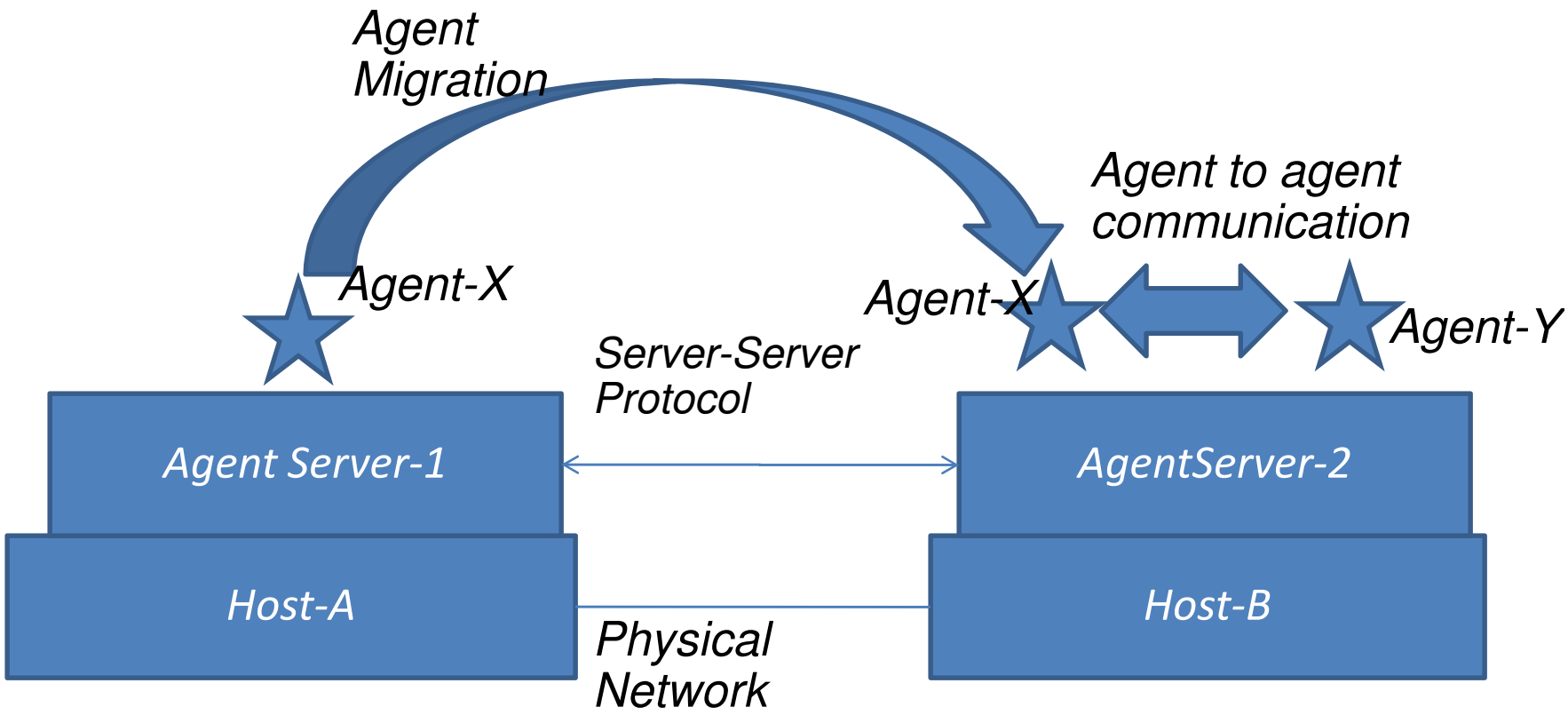
- **Weak Mobility**
  - When moving a mobile agent carries code + data state
  - Data State - global or instance variable
  - On moving, execution has to start from the beginning

Levels of Mobility

- **Strong Mobility**
  - When moving a mobile agent carries

    code + data state + execution state
  - Data State - global or instance variable
  - Execution State – local variables and threads
  - On moving, execution can continue from the point it stopped on the previous host

Simple M.A. Architecture



Mobile Agent Terms

- From the Object Management Group MASIF specification
- **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**.**

- **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.

MASIF (Mobile Agent System Interoperability Facilities)

Mobile Agent Terms

- **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 Call (RPC). The communication needs of stationary agents are met by current distributed object systems such as CORBA, DCOM, and RMI.

- **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.

DCOM (Distributed Component Object Model)

Mobile Agent Terms

- **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.

- **Place**

  Basic Environment in Which an Agent Executes, provides the Operating System for the Agent.

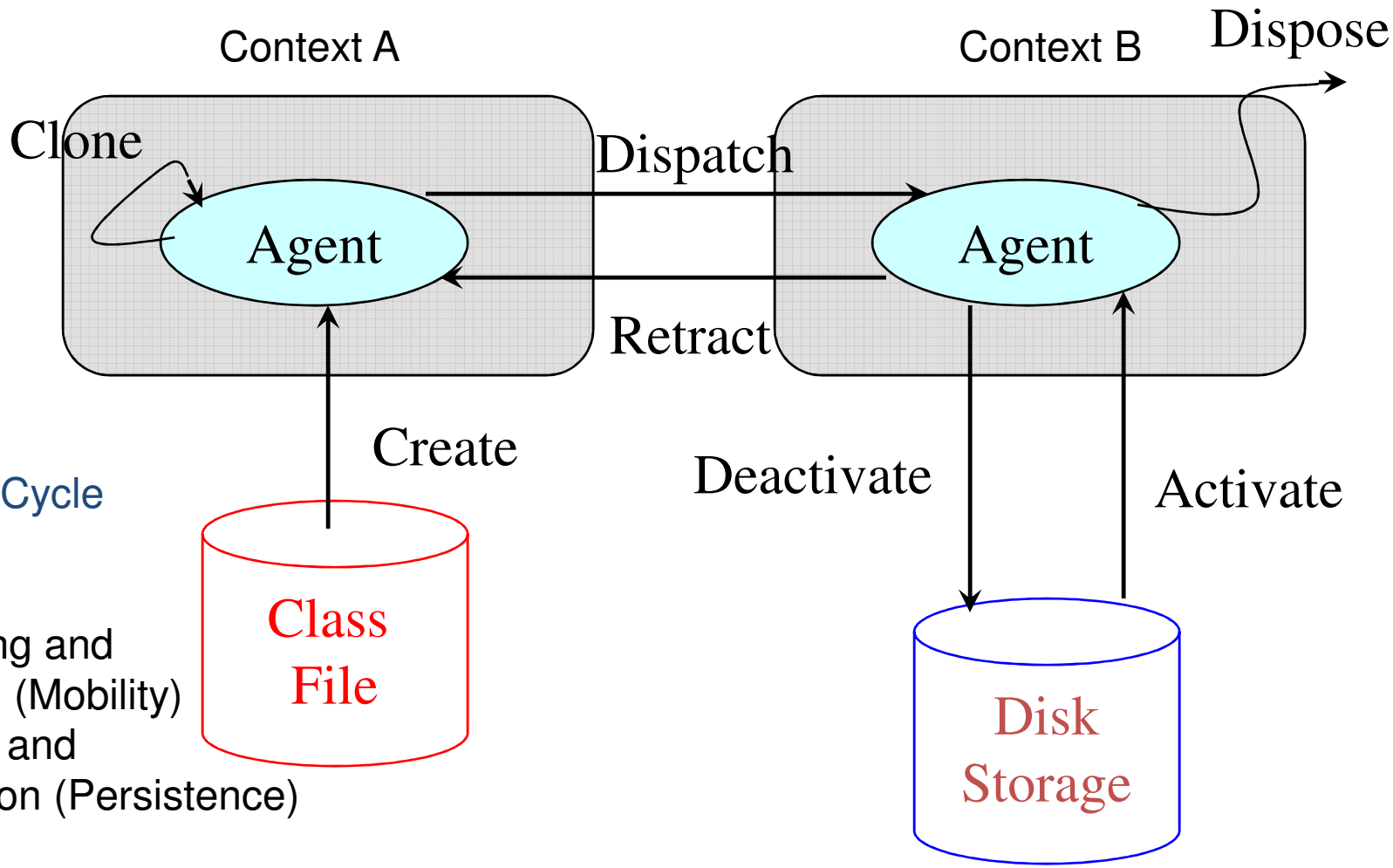
Events in Mobile Agent's life-time

- **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.

Events in Mobile Agent's life-time

- **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.

Agent Life-Cycle Model



Agent Life Cycle
-Creation
-Cloning
-Dispatching and Retraction (Mobility)
-Activation and Deactivation (Persistence)
-Disposal

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

Potential Mobile Agent Applications

- **Secure Brokering** - Allow parties to meet on a trusted host where collaboration can take place without worry that the host will aid one party or the other.
- **Distributed Information Retrieval** - Agents roam, collect desired information and eventually return with information of interest.
- **Personal Assistant** - Allows agent to act on the behalf of its creator at remote hosts without fear of connectivity problems.
- **Server Farm Maintenance** - Agent can roam connected computers performing a plethora of tasks - i.e. installs, upgrades, backups, monitoring logs.
- **Monitoring and Notification** - Agents can be dispatched to wait for certain types of information to become available then notify the user or act upon the information.

Application of mobile agents

- E-Commerce:
- Personal Assistance:
- Secure Brokering:
- Distributed Information Retrieval:
- Telecommunication Networks Services:
- Workflow Application and Groupware:
- Monitoring and Notification:
- Information Dissemination:
- Parallel Processing:
- Remote control

Quality Requirements of Mobile Agent

- **Interoperability**
- **Scalability**
- **Mobility**
- **Security**

Quality Requirements of Mobile Agent

Interoperability

Interoperability between agents can be achieved with help of three key elements

- Common agent communication language and protocol
- Common format of context of communication, and
- Shared ontology

Quality Requirements of Mobile Agent

Scalability

How well the capacity of a system to do useful work increases as the size of the system increases.

Mobility

Performance can be achieved by moving agents closer to the services available on the new host.

Quality Requirements of Mobile Agent

Security

- **Confidentiality:-** sensitive data must be secure.
- **Integrity:-** altering data must be detected.
- **Authentication:-** an agent must authenticate itself to the host, and an agent server must authenticate itself to the agent.
- **Authorization:-** host enforces strict access control to its resources.
- **Auditing:-** keeping track of the system, if an agent misbehaves, this should be logged.

Mobile Agent Security Concerns

- **Host security**
  - protecting a host against malicious agents
  - protecting a host against other hosts
- **Agent security**
  - protecting agents against malicious hosts
  - protecting agents from other agents
- **Secure communication between agent's**

Security Threats (Agent-to-Platform)

- **Masquerading :-** Agent poses as another agent to gain access to services or data at a host.
- **Denial of Service :-** Agents may attempt to consume or corrupt a hosts resources to preclude other agents from accessing the host's services.
- **Unauthorized Access :-** Agents can obtain access to sensitive data by exploiting security weaknesses.

Security Threats (Agent-to-Agent)

- **Masquerade :-** Agent may attempt to disguise its identity in an effort to deceive the agent with which it is communicating.
- **Denial of Service:-** Agents can also launch denial of service attacks against other agents.
- **Repudiation:-** Repudiation occurs when an agent, participating in a transaction or communication, later claims that the transaction or communication never took place.
- **Unauthorized Access:-** An agent can directly interfere with another agent by invoking its public methods (e.g., attempt buffer overflow, reset to initial state, etc.)

Security Threats (Platform-to-Agent)

- **Masquerade :-** An agent platform may be able to extract sensitive information from agent.
- **Denial of Service :-** A malicious agent platform may ignore agent service requests (simply not execute the agent's code, terminate the agent without notification).
- **Eavesdropping :-** Agent platform can not only monitor communi- cations, but also can monitor every instruction executed by the agent.
- **Alteration :-** A malicious platform can modify an agent's code, state, or data.

Security Threats (Other-to-Agent platform)

- **Masquerade :-** An agent on a remote platform can masquerade as another agent and request services and resources for which it is not authorized.
- **Unauthorized Access :-** Remote users, processes, and agents may request resources for which they are not authorized.
- **Denial of Service :-** Agent platforms are also susceptible to all the conventional denial of service attacks aimed at the underlying operating system or communication protocols.
- **Copy and Replay :-**A platform that intercepts an agent, or agent message, in transit can attempt to copy the agent, or agent message, and clone or retransmit it.

Technical Issues 1

- **Sending agents**:- mechanism?
- **Identifying agents** (and agents with their owners):- naming conventions (standards?), mechanism, authentication.
- **Scheduling on server-side**:- distribute limited CPU and memory resources – fairness.
- **Security**:- protect the server and protect the agent.

Technical Issues 2

- **Efficiency of agent execution:-** time matters, if Java or Python, performance penalties with interpretation (esp. for performance critical applications).

- **Platform independence:-** run anywhere?

- **Detect tampering of agent code and data at runtime.**

Technical Issues 3

- **Strong mobility**:- move full execution state (stacks etc).

- **Mobile Agent management**:-
  - how control and manage deployed agents
  - issues: fault tolerance (e.g., agent fails, host fails), recalling agents, tracking agents, servicing agents (esp. longer living agents)

- **Others**:- Interoperability between different agent toolkits, payment, authentication.

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
  www.fipa.org

Groups working on Mobile Agent Systems

- **Government Agencies**
  - Rock Island Army Arsenal (USA)
- **Academic Institutions**
  - Carnegie Melon University
  - Massachusetts Institute of Technology
  - Stanford University
  - University of Maryland Baltimore County
  - Tromso University
  - Dartmouth College
- **Computer Industry Companies**
  - IBM
  - Microsoft
  - Mitsubishi

Mobile Agent Systems

- **Tacoma** - Tcl based system developed at Cornell and Tromso University (1994-95)
- **Agent Tcl** - Tcl based system developed at Dartmouth College. (1994-95) D'Agents
- **Aglets** - Java based system from IBM. (1996)
- **Concordia** - Java based system from Mitsubishi Research. (1997)
- **Voyager** - Java based system from ObjectSpace
- **Odyssey** - Java based system from General Magic

See http://www.informatik.uni-stutgart.de/pvr/projekte/mole/mal/mal.html

Name	Description	Developer	Language	Application
Aglet	Java Class libraries	IBM, Tokyo	Java	Internet
Agent Tcl	Transportable agent system	R. Gray, U Dart.	Tcl Tk	Information management
Concordia	Framework for agent development	Mitsubishi E.I.T.	Java	Mobile computing, Data base
Odyssey	Set of Java Class libraries	General Magic	Telescript	Electronic commerce
OAA	**O**pen **A**gent **A**rchitecture	SRI International, AI	C, C-Lisp, Java, VB	General purpose
Ara	**A**gent for **R**emote **A**ction	U Kaiserslautern	C/C++, Tcl, Java	Partially connected c. D.D.B.
Tacoma	**T**romso **a**nd **C**ornel **M**oving **A**gent	Norway & Cornell	**C**, **UNIX-**based**,**	Client/Server model issues / OS support
Voyager	Platform for distributed applic.	ObjectSpace	Java	Support for agent systems
AgentSpace	Agent building platform	Ichiro Sato, O. U.	Java	General purpose

Name	Description	Developer	Language	Application
Kali Scheme	Distributed impl. of Scheme	NEC Research I.	Scheme	Distributed data mining, load balancing
The Tube	mobile code system	David Halls, UK	Scheme	Remote execution of Scheme
Ajanta	Network mobile object	Minoseta U.	Java	General purpose
Knowbots	Research infrastructure of MA	CNRI	Python	Distributed systems / Internet
AgentSpace	Mobile agent framework	Alberto Sylva	Java	Support for dynamic and dist. Appl.
Plangent	Intelligent Agent system	Toshiba Corporation	Java	Intelligent tasks
JATlite	Java Agent framework dev /KQML	Standford U.	Java	Information retrievial, Interface agent
Mole	First Java-Based MA system	Stuttgart U. Germany	Java, **UNIX**-based	General purpose
MOA	**M**obile **O**bject and **A**gents	OpenGroup, UK	Java	General purpose

Protecting the Agent Platform

- **Agent Integrity**
  - hash, MAC, proof-carrying code
- **Agent Authentication**
  - digital signatures (analogy: signed applets)
- **Authorization**
  - access control lists

Protecting the Agent Platform

- **Software-Based Fault Isolation**

  Technique allows untrusted programs written in an unsafe language, such as C, to be executed safely within the single virtual address space of an application.

R. Wahbe, S. Lucco, T. Anderson, “Efficient Software-Based Fault Isolation”
<URL: http://www.cs.duke.edu/~chase/vmsem/readings.html>

Protecting the Agent Platform

- **Safe Code Interpretation**

Commands considered harmful can be either made safe for or denied to an agent. One of the most widely used interpretative languages today is Java.

Static type checking in the form of byte code verification is used to check the safety of downloaded code.

A security manager mediates all accesses to system Resources.

John K. Ousterhout, "Scripting: Higher-Level Programming for the 21st Century," IEEE Computer, March 1998, pp. 23-30.

Protecting the Agent Platform

- **Signed Code**

  A fundamental technique for protecting an agent system is signing code or other objects with a digital signature.

Neeran Karnik, “Security in Mobile Agent Systems,”
<URL: http://www.cs.umn.edu/Ajanta/>

Protecting the Agent Platform

- **Path Histories**

Computing a path history requires each agent platform to add a signed entry to the path, indicating its identity and the identity of the next platform to be visited, and to supply the complete path history to the next platform.

To prevent tampering, the signature of the new path entry must include the previous entry in the computation of the message digest.

David Chess, Benjamin Grosof, Colin Harrison David Levine, Colin Parris, and Gene Tsudik
"Itinerant Agents for Mobile Computing" <URL www.cs.umbc.edu/kqml/papers/itinerent.ps >

Protecting the Agent Platform

- **Proof Carrying Code**

  A code receiver establishes a set of safety rules that guarantee safe behavior of programs

  Java system for example, the byte-code verifier can make such a guarantee, but only if there's no bug in the verifier itself, or in the just-in-time compiler, or the garbage collector, or other parts of the Java virtual machine (JVM).

Andrew W. Appel- “Foundational Proof-Carrying Code”

<URL www.cs.princeton.edu/~appel/papers/fpcc.pdf>

Protecting Agents

- **Execution tracing**

  All agent actions logged

  Logs must be stored securely

- **Drawbacks**

  Maintenance of potentially large logs

  Does not prevent the host from forging the log right when it is created

Giovanni Vigna, "Protecting Mobile Agents Through Tracing“
<URL: http://www.cs.ucsb.edu/~vigna/listpub.html>

Protecting Agents

- **Encrypted function computing**

  Host computes the value of f() without real knowledge about what f() is.

- **Drawbacks**

  No applicable cryptographic theory available.

Thomas Sander and Christian Tschudin, "Protecting Mobile Agents Against Malicious Hosts"
<URL: http//www.icsi.berkeley.edu/~tschudin/>

Protecting Agents

- **Reference states**

  Several states of the agent are recorded using a trusted reference host.

- **Drawbacks**

  Reference states may not be trivial given an agent with complex execution state.

Fritz Hohl - “A Framework to Protect Mobile Agents by Using Reference States”
<URL http://elib.uni-stuttgart.de/opus/volltexte/2000/602/pdf/TR-2000-03.pdf>

Protecting Agents

- **Obfuscated code**

  Rename classes name, method's name, package's name, object's name and other identifiers in the program.

- **Drawbacks**

  The agent's code, can be reverse engineered**.**

Fritz Hohl, "Time Limited Blackbox Security: Protecting Mobile Agents From Malicious Hosts"
<URL www.springerlink.com/index/7yutn3xpv5vm60g9.pdf >

Protecting Agents

- **Environment Key Generation**

  An agent takes predefined action when some environmental condition is true.

- **Drawback**

  Hostile Platforms can force the agent to execute by artificial generation of the environment key.

James Riordan and Bruce Schneier, “Environmental Key Generation Towards Clueless Agents” <URL http://portal.acm.org/citation.cfm?coll=GUIDE&dl=GUIDE&id=746194>

Mobile Agent Platform - Aglet

- Aglets is a Java mobile agent platform and library that eases the development of agent based applications. An *aglet* is a Java agent able to autonomously and spountanously move from one host to another.

- Developed at the IBM Tokio Research Laboratory

- Aglets includes
  - complete Java mobile agent platform
  - a stand-alone server called *Tahiti*
  - a library that allows developer to build mobile agents

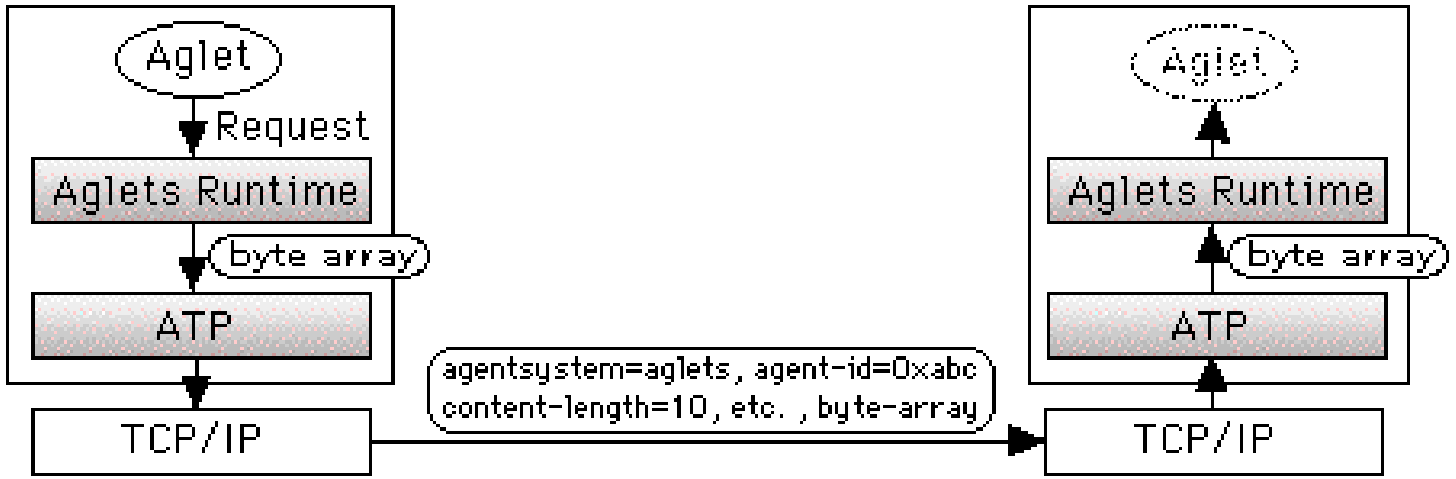
- Currently, stable release of Aglets are available in the 2.0 series, and **2.0.2** is the latest one.

- Resources

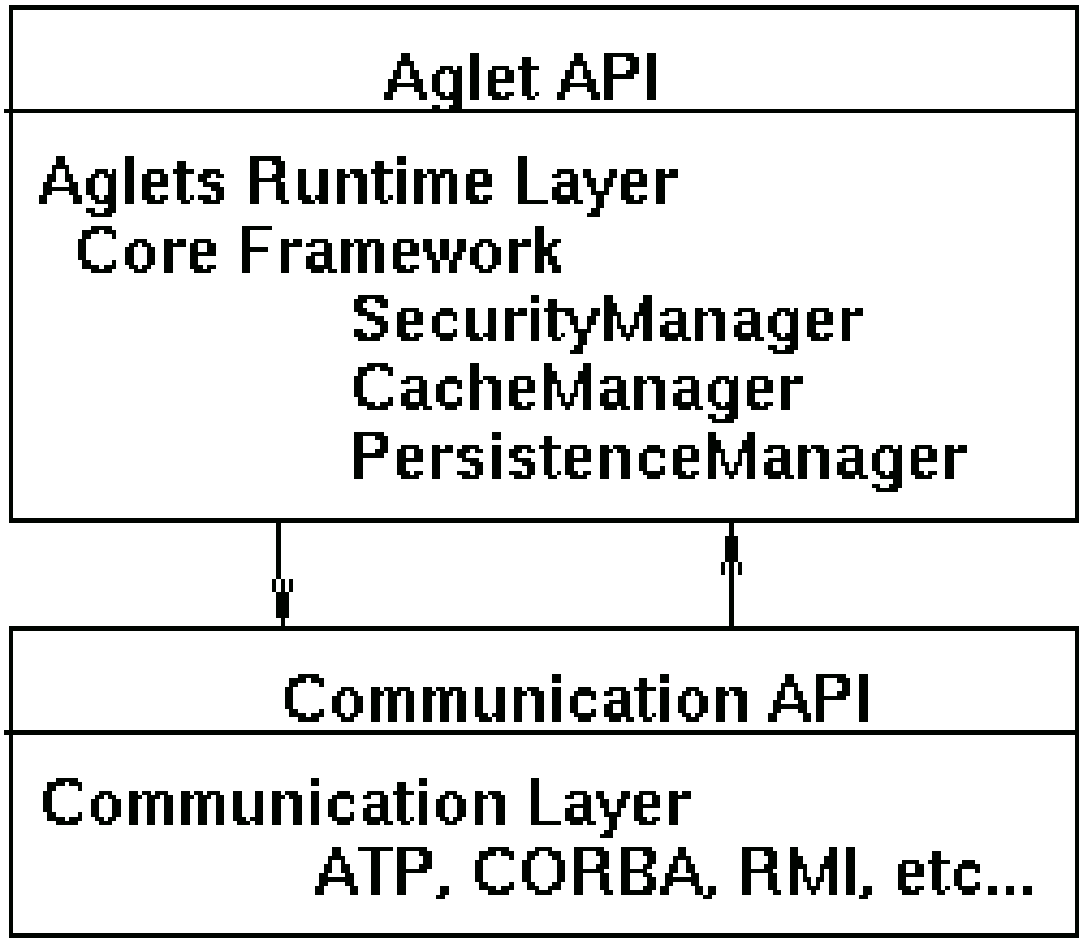
  http://aglets.sourceforge.net/ http://www.trl.ibm.com/aglets/

Aglets architecture

- Aglet API
- Aglets Runtime Layer - The implementation of Aglet API
- Agent Transport and Communication Interface (ATCI with ATP as an application-level protocol)
- Transport Layer



Aglets architecture



Aglet API: Classes and Interfaces

- Aglet API
  - Simple and Flexible
  - Represents Lightweight Pragmatic Approach to Mobile Agents
- Java Classes
  - Aglet
  - Message
  - Futurereply
  - Agletid
  - Agletproxy
- Java Interfaces
  - Agletproxy
  - Agletcontext

Aglet Class

- com.lib.aglet.Aglet
  - Defines the fundamental methods used to control the mobility and life cycles of mobile aglets.
  - Void Aglet.onCreation(Object init)
  - Void Aglet.dispatch(URL)
  - Void Aglet.run()
  - Object Aglet.clone()

AgletProxy Class

- com.lib.aglet.AgletProxy
  - Acts as a handle of an aglet and provides a common way of accessing the aglet behind it.
  - Since an aglet class has several public methods that should not be accessed directly from other aglets for security reasons.
  - Any aglet that wants to communicate with other aglets has to first obtain the proxy object.

  - AgletProxy. getAgletInfo()
  - AgletProxy. getAgletID()

Message Class

- com.ibm.aglet.Message
  - Aglet objects communicate by exchanging objects of the Message class.
  - Agletproxy Class is Responsible for Actually Sending and Receiving the Messages

  - AgletProxy.sendMessage(Message msg)
  - Aglet.handleMessage(Message msg)

AgletContext Class

- com.ibm.aglet.AgletContext
  - Is used by an aglet to get information about its environment and to send messages to the environment and other aglets currently active in that environment.
  - It provides means for maintaining and managing running aglets in an environment where the host system is secured against malicious aglets.
  - AgletContext. getAgletProxy(AgletID)
  - AgletContext. getHostingURL()
  - AgletContext.setProperty()
  - Aglet.getProxy()

AgletID Class

- com.ibm.aglet.AgletID
  - Represents the Identifier of the Aglet
  - The Identifier is Unique to Each Aglet
  - The Identifier Object Hides the Implementation Specific Representation of the Aglet Identity

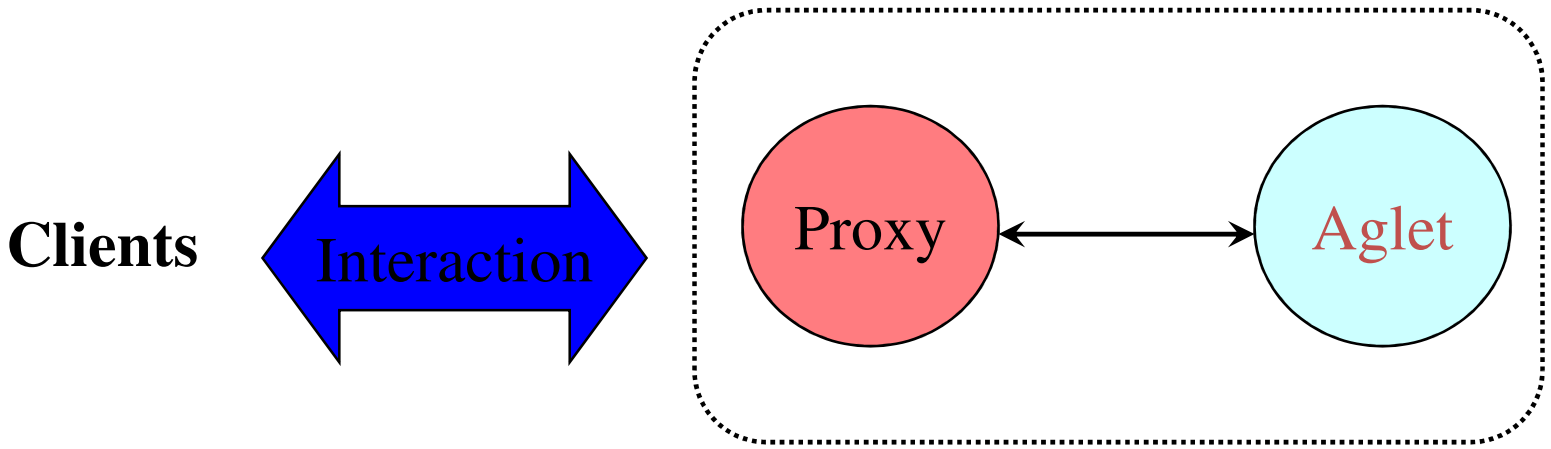
  - proxy.getAgletID();
  - AgletContext.getAgletProxy(aid);

Event

The event	*about to take place*	*After the event has taken place*
Creation		onCreation()
Clone	onCloning()	onClone()
Dispatch	onDispatching()	onArrival()
Retract	onReverting()	onArrival()
Dispose	onDisposing()	
Deactivate	onDeactivating()	
Activate		onActivation()
Message		handleMessage()

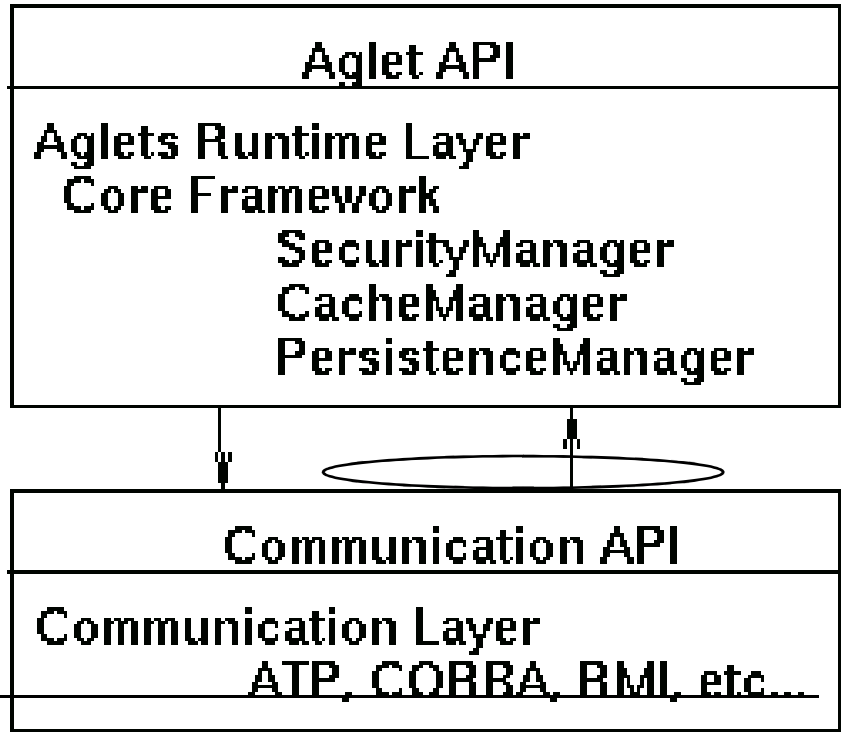
Relationship between Aglet and Proxy

- Proxy Represents the Aglet
- Shields Public Methods for Potential Misuse
- Can Hide the “Actual” Location of Aglet
- Proxy and Aglet on Different Computing Nodes



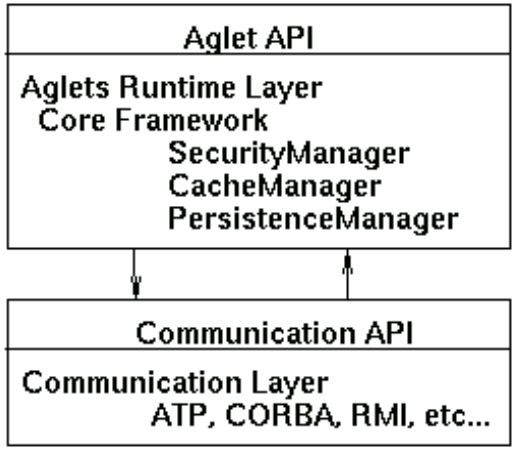
Aglets Runtime Layer

- The implementation of the Aglet API, and define the behavior of the API components.
- Provides the fundamental functions for aglets to be created, managed, and dispatched to remote hosts.



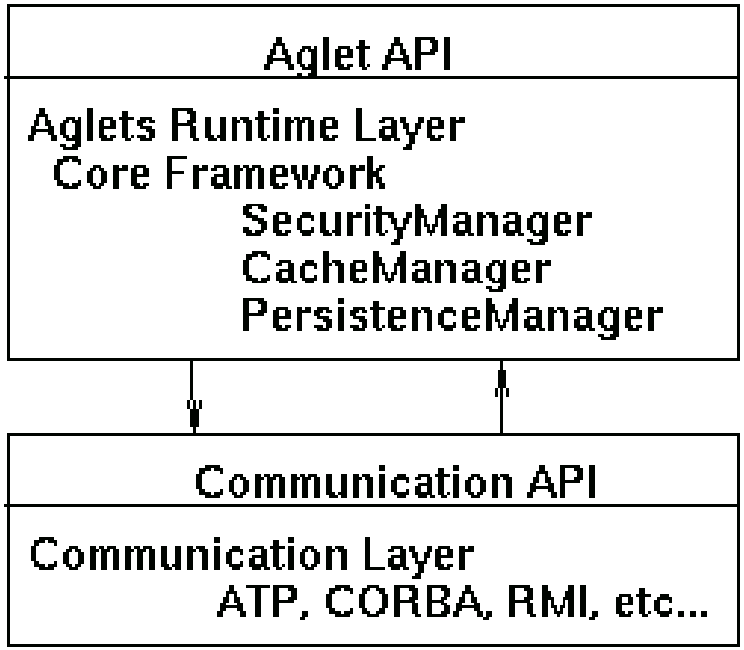
Core Framework

- Provides the following mechanisms fundamental to aglet execution:
  - Serialization and deserialization of aglets
  - Class loading and transfer
  - Reference management and garbage collection.



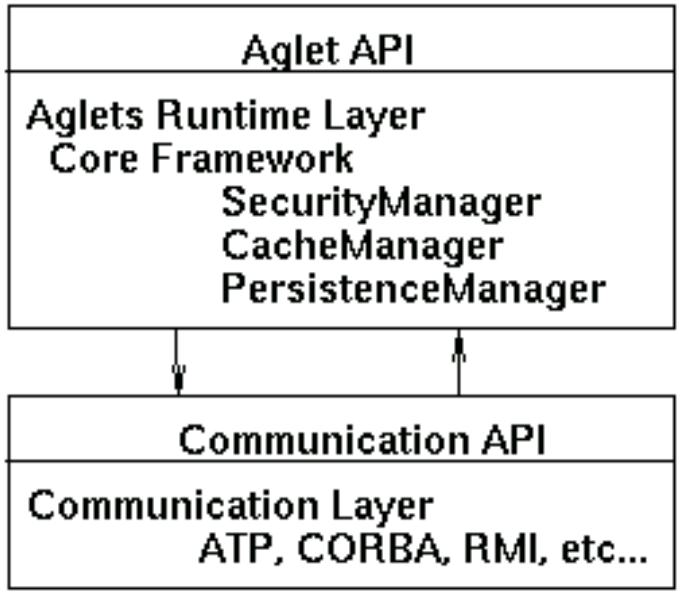
PersistenceManager

- Storing the serialized agent, consisting of the aglet's code and state into a persistent medium such as a hard disk.



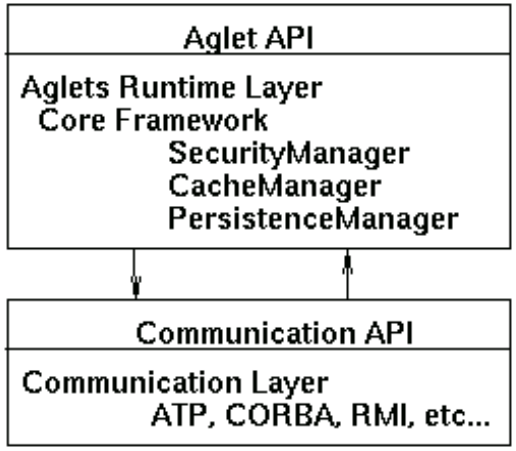
CacheManager

- Maintaining the bytecode used by the aglet.
- Because the bytecode of an incoming aglet needs to be transferred when the aglet moves to the next destination, it caches all bytecode even after the corresponding class has been defined



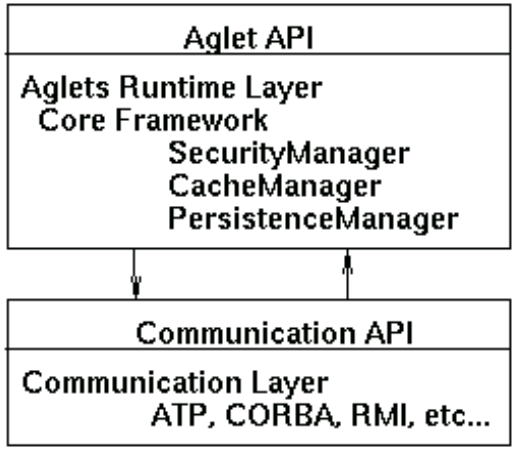
SecurityManager

- Protecting hosts and aglets from malicious entities.
- It hooks every security-sensitive operation and checks whether the caller is permitted to perform it.
- There is only one instance of SecurityManager in the system, and it cannot be altered once it has been installed.



Communication Layer

- Communication API defines methods for creating and transferring agents, tracking agents, and managing agents in an agent-system-and protocol independent way.
- The Agent transfer Protocol is the default implementation of the communication layer.
- ATP is modeled on the HTTP protocol.



Aglet Security Model

- **Principals** - authenticated identities used to enforce the security policy: Consists of Aglet and Context, and their associated Manufacturers and Owners, and the Network Domain (group of trusted servers).

- • **Permissions** - Sets actions available on resources by principals. Based on JDK 1.2 policy file definition. For example, an aglet may be given access to a specific file.

- • **Protections** - Principle's means to protect its resources. For example an agle t may request that only itself (or its owner) be able to dispose of it.

- • **Security Policy** - Set of rules defining permissions and protections. Set by policy authority (the principal responsible for a particular resource). For example, the Context authority is responsible for keeping the server safe from malicious agents. It might have the following policy:

  grant codebase "http://thishost" signed by "IBM" owned by "Alex" {permission java.io.FilePermission "/tmp/file.dat "read"; java.net.SocketPermission "www.trl.ibm.co.jp:80" "connect";

Tahiti

- An application program that runs as an agent server.
- Multiple servers on a single computer by assigning them different port number.
- Provides a user interface for monitoring, creating, dispatching, and disposing of agents and for setting the agent access privileges for the agent server.

Current mobile agents research activities

- **Service provisioning in the mobile network**

  Service provisioning is defined as the setting in place and configuring of the hardware and software required for activating a telecommunications service to a customer.

  More details can be found in project A-STORM.

- **Remote Software Operation and Management**

  Management of the software which is distributed over the network is demanding task as the number of computers in it and/or geographical distance between them grows. More details in project ROPE

- **Service Management in Grid**

  Service installation, starting and testing on a large-scale system with many nodes has become the serious problem. Mobile agents It supports managing multiple remote systems at the same time, support for complex software installation procedures and installation of a large software .

  More details in project ROPE.

Current mobile agents research activities

- **Agents in e-commerce**

  The Trading Agent Competition (TAC) is an international forum that promotes research in the trading agent problem.

- **Mobile Agent Platform Development**

  Develop a core platform with basic functionalities and easy to use in order to achieve good performance and execution in heterogeneous environments.

  More details in project Crossbow.

- **Formal Specification and Verification of Mobile Agent System**

  In the new generation network, where users and services are mobile, some entities change locations and are connected to different other nodes at different point of time. Designing of this kind of network, where the entities are moving, is more complicated than in static environments.

- Designing an algorithm for mobile agent security preventing attacks at the application level
  - Encryption of mobile code
  - Run-time, on-demand decryption for execution
  - No decrypted information stored at the guest machine

Agent Tcl

Sources of Information

- Mobile Agents Introductory
  http://www.infosys.tuwien.ac.at/Research/Agents/intro.html

- The Mobile Agent List
  http://mole.informatik.uni-stuttgart.de/mal/mal.html

- Mobile Agent Applications
  http://www.computer.org/concurrency/pdf/p3080.pdf

- Software Engineering Concerns for Mobile Agent Systems
  http://www.elet.polimi.it/Users/DEI/Sections/Compeng/GianPietro.Picco/ICSE01mobility/papers/cook.pdf

Sources of Information

- http://whatis.techtarget.com
- Http://www.trl.ibm.com
- http://www.aajc.thu.edu.tw/
- http://turtle.ee.ncku.edu.tw
- www.aajc.thu.edu.tw/Services/JavaLetter/aglet.pdf
- ftp.mayn.de/pub/unix/devel/MIRROR.docs/aglets.pdf
- http://ftp.mayn.de/pub/unix/devel/MIRROR.docs/aglets.pdf

THANKS