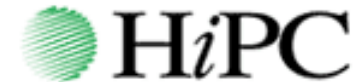


Designing Distributed Applications using Mobile Agents



Sridhar Iyer
Vikram Jamwal

KR School of IT,
IIT Bombay, INDIA



**International Conference on
High Performance Computing**

December 17, 2001
Hyderabad, INDIA

Outline

- Motivation
- Mobile Agent technology
- Application domains
- MA frameworks overview
- MA based Structuring
- MA Framework Issues
- MA Application Case Studies
- Conclusion

Distributed Computing Outlook

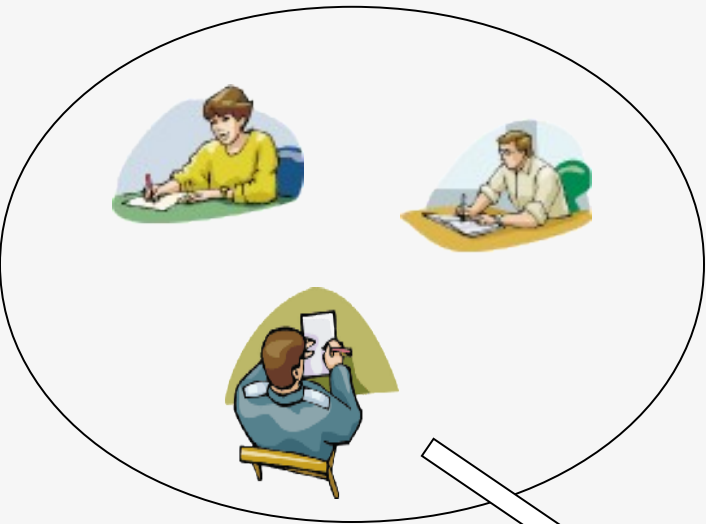
- Peer-Peer computing
- Context-aware computing
- Mobile computing
- Distributed communities
 - networks of mobile and fixed people, devices and applications
 - Virtual communities for e-business, e-government and e-education
 - Real-time 3D environments
- Intelligent environments

Required ...

- Dynamic adaptations in heterogeneous environments
- Self-organizing systems
- Metadata, ontologies and the Semantic Web
- More than simple client-server interactions
 - From **one-one** or **many-one** interactions to
 - One-many
 - Many-many
- Support for collaborations
- **We shall pick one representative application viz. Distance Evaluation**

Distance Evaluation

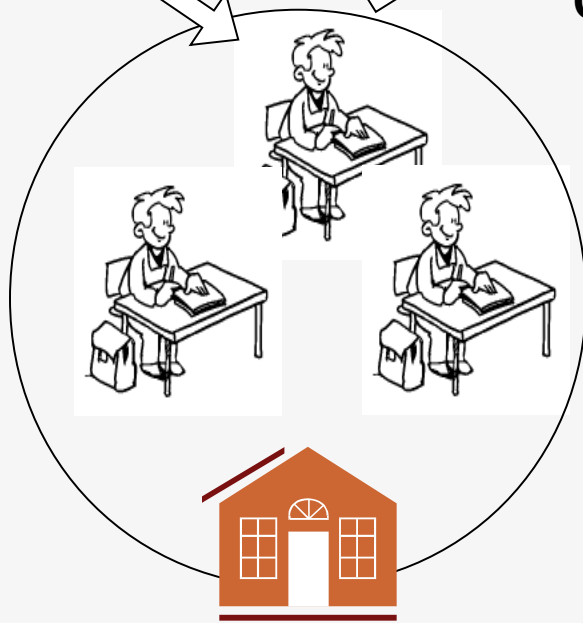
- Emergence of distance education
 - Need for distance evaluation mechanisms
- Alternatives to paper-based exams
 - Computer based and Internet based
- Scheduled and uniform exams
- Scenario
 - IIT-JEE type of examination
- Stages
 - Paper Setting
 - Distribution and Testing
 - Evaluation and Result publication



Paper Setting



**Evaluation and Result
Compilation**



**Distribution and
Testing**



**Result
Publication**

Design Goals

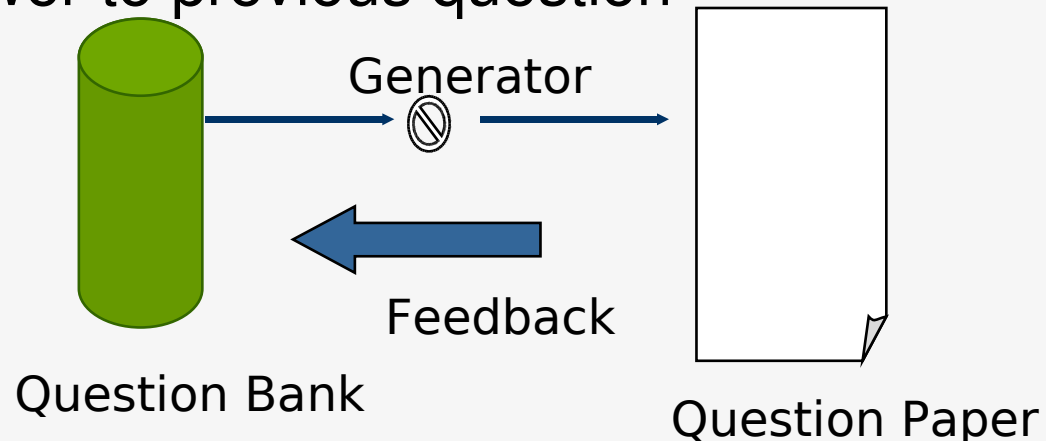
- Map to real life scenario
- Automate as much as possible
- Minimize the infrastructure at different ends
- Include all the stages

Types of e-testing mechanisms

- Where does the database reside?
- Locally
 - Computer Based Testing (CBT)
 - Examples
 - GRE and GMAT
- Remote
 - Internet based testing
 - Example
 - www.netexam.com
- These are *any time* exams

Computer Based Testing (CBT)

- Different Question-Paper for each examinee
 - Generated dynamically
- Adaptive
 - Different weights given to different questions
 - Next question decided on the basis of
 - difficulty level and
 - correctness of answer to previous question



Internet Based Testing

Existing Schemes

■ Front End

- Mostly “HTML - form based”
- answers sent using ‘GET’ and ‘POST’ methods
- Java applets, Java Script, flash

■ Back End

- CGI scripts
- Java servlets

■ Security

- Authentication done using ‘login - password’
- May use “https” for secure exchange

■ Some issues

- Web Servers are basically stateless

Important Points

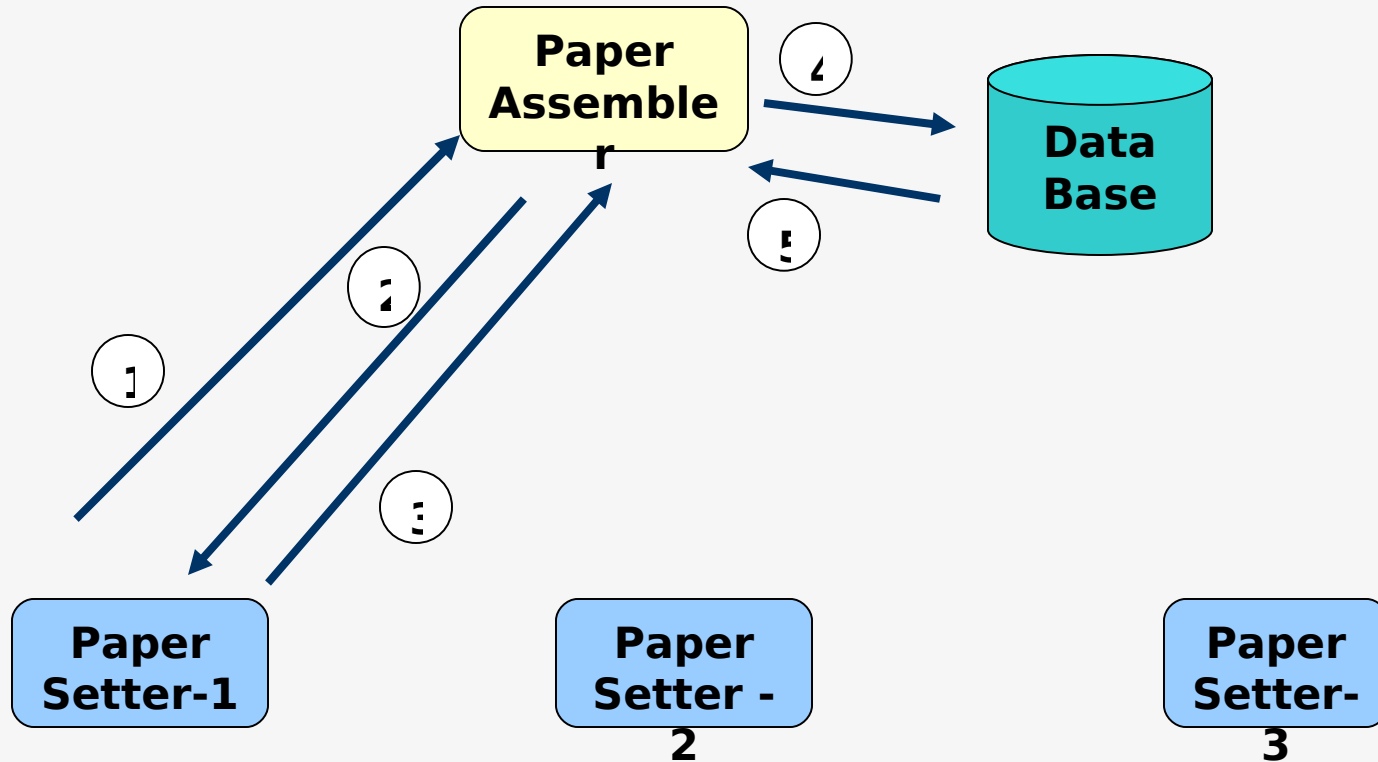
- Existing models are basically
 - *Pull* based
 - Client-Server
- Extending Internet based evaluation techniques
 - Push model
 - Different kinds of content
 - Dynamic organization of content
 - Off-line examination V/S on-line examination
 - Subjective answers (not just objective)
- Key technical issues
 - How to deliver the exam content?
 - How to evaluate the answers?

Client-Server Solution :

Paper Setting Stage

- Paper Setters (PS) are distributed over a large area.
- PS may want to work offline (why?)
- PS need to be sent notification by the Paper Assembler (PA) from time to time
- At appropriate time the Question Paper(QA) needs to be gathered, even if partially done

C S Design: Paper Setting



CS Design

- After supplying his Login id and password (1)
 - (which he has procured from e-mail or any other source),
- each examiner accesses a web-form from the server (2).
- After filling the form, he submits it back to the paper assembler(3).
- PA server stores each PS's QP in a database with appropriate indexing (4).
- At a later time, PA queries the database for all partial QPs and builds an comprehensive paper (5)

Drawbacks

- PA cannot send notifications to the PS
- PA cannot get the partial QPs if PS don't respond
- The functionality / type of content is limited by client capability
 - Rest of the info has to be uploaded in the form of files.
- There is no provision for
 - the local storage of partial data on the form.
 - This might be required if the examiner is coming back a later date to complete his remaining work.

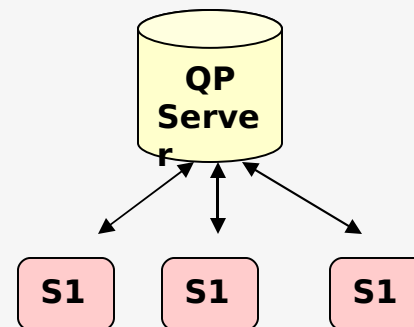
Client Server Solution:

Paper Distribution and Testing

- Papers should be distributed to the centers *just-in-time*
- QPs contain dynamic content
- Students cannot contact any other machine as long as the examination is going on
 - or they can contact only the supervisor
- Notifications might need be sent
 - to the specific, group or all of students during the course of examination.
- Students access QPs only from their terminals
- Students get to answer only for a specific time
- The center needs to certify the students answer-sheet

CS-Design: Distribution and Testing

- Each student makes a request to the QP Server (which can be web server to supply it the Question paper).
- After validation, the QP server supplies a web page containing the question papers and a form.
- The student, if he requires a further section repeats the request (or gets it after submitting the previous answers).



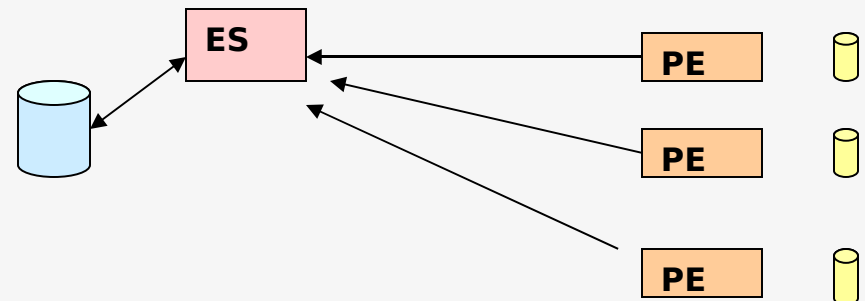
Drawbacks:

- The web-server (QP server) gets overloaded by various clients requesting at the same time
- Web server needs to maintain a state information for each student
 - the number of sections he has been offered,
 - the time elapsed for each student.
- A simple web-form offers various limitations like the kind of multimedia content it can support
- The information (e.g. question papers or corrections) cannot be pushed to the clients.
- If the paper collection site is different
 - user need to push the answers to a different web-server.
- The responsibility that an answer paper is submitted properly is now with the student.

Client – Server Solution:

Paper Evaluation and Result Publication

- The Paper evaluators (PE) contact the Evaluation Server (ES) for their set of papers.
- The ES retrieves the information from the database, prepares a web-page and send it the PE.
 - ES possesses the logic about which sub-set of which student should be sent to a particular PE.
- After all responses have come, ES compiles results and publishes them



Drawbacks:

- ES cannot push the information to the PEs
 - depends upon them to fetch the information
- ES has to maintain information about
 - each students, subsection, and the part forwarded to a PE and its status.
- Where a paper needs to pass multiple PEs
 - a ES has to coordinate passively.
 - As ES does not get the evaluated copy till PE decides to send it.
- mode of push
 - Mechanisms like e-mail
 - Do not tightly tie the system

Observations

- Client – Server Solutions do not always scale
- Do provide the solution in many cases but tend to create
 - Cumbersome solutions when complexity increases
 - Unintuitive designs
- **Need for alternate structuring mechanisms**

Can Mobile Agents Help ?

- What are they ?
- How to exploit their advantage ?
- Constituents
 - Agent
 - Mobility
- We shall first discuss the agent in general and then focus on mobile agents

Agents: An Introduction

- One of the most hyped Software technologies of recent times
- We shall now try to:
 - Define Software Agents
 - Classify Software Agents
 - Discuss their relevance
 - Look at their enabling technologies

What Software Agents

are and aren't

- Exact definition difficult
 - term is over-used in various contexts.
- The English word "agent"
 - shares the same Latin root as the word "act"
 - "someone (or something) that has the ability, power or authority to *act*",
- When applied to software
 - "a program that does something on behalf of the user"
 - too general
 - can actually describe *any* running program!
- Alternative approach
 - look at the common list of *ideal characteristics* that most Software Agent Systems seem to share

Experts say:

Agent should display

■ **Autonomy**

- system does its work in a *largely pro-active* manner, without explicit control from the user at every step
- In practice, most agents will check with the user, at least for major decisions and actions

■ **Intelligence**

- system does something smart
- exhibits behaviour that humans would consider intelligent
- typically solving some non-trivial problem, and
- exhibiting some kind of *learning*, that is, the ability to adapt and improve over time

Experts say:

Agent should display

- **Cooperation**

- the system *collaborates*, minimally with the user, and usually with *other agents*, to perform the task.

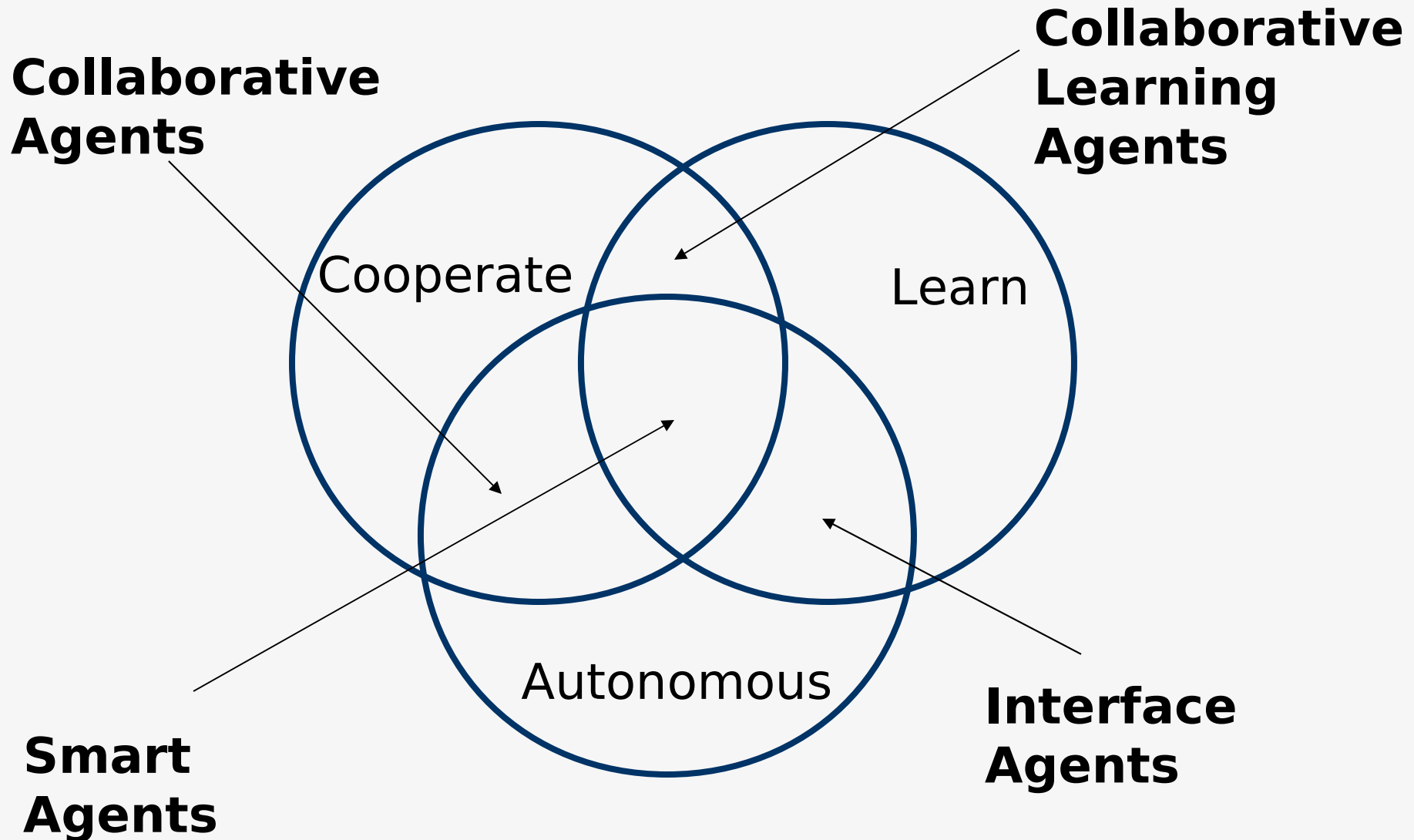
Based on this criterion, we can categorize agents into the four classes

[see figure]



Part-View of Agent Topology

[Nwana 96]



Agent – Non Agent:

according to above classification

- Should exhibit *at least two* of the above features to a reasonable degree
- those *in the non-overlapping area* of the circles *are not* considered to be Software Agents
- Non-Agent examples
 - expert system
 - exhibit intelligence
 - not cooperation or autonomy
 - a web indexing robot
 - might be autonomous,
 - may not very intelligent
 - a distributed system
 - might display some collaboration among the components
 - may be neither autonomous nor intelligent

Classification of Agents:

- **Single-Agent vs Multi-Agent (Collaborative)**
 - A multi-agent system involves a collection of agents collaborating to perform a task.
 - This model is more complex, but more modular, and is well-suited for concurrent or distributed tasks.
- **Static vs Mobile**
 - Static agents work on a single system,
 - mobile agents (also known as "bots") move from system to system.
 - Depending on the degree of autonomy and the nature of the task, mobile agents may keep sending intermediate status to the base.

Classification of Agents:

■ Homogeneous vs Heterogeneous

- Both are multi-agent systems
- In a homogeneous system: agents are all similar
- In a heterogeneous system: are of different types.

■ Deliberative vs Reactive

- A **deliberative** agent has an explicit representation of its domain
 - and uses *symbolic reasoning* to do its task.
 - often has an explicit representation of its own **beliefs**, **desires** and **intentions** (called the **BDI** model), and uses these in its reasoning.
- Deliberative agents involve explicit programming of the knowledge and problem-solving in the conventional way, and is the way most agents currently work.

Classification of Agents:

- A **reactive** agent consists of many agents,
 - each of which has a very simple stimulus-response type behaviour.
 - A single agent typically has no clue about the actual task to be performed, but the collective action of the group has an *emergent* behaviour which causes the required task to be accomplished.
- This type of behaviour, for example, is shown in ant and bee colonies, where each insect works independently and seemingly chaotically, but the overall effect is quite dramatic.
- **Hybrid** agents are those which combine more than one philosophy within the same agent.

Classification of Agents:

reactive school of thought

■ Real-world problem-solving

- rarely involves explicit and elaborate reasoning and planning,
- more of local responses based on the current situation, which keeps changing.

■ Advantage

- allows a simple model
- potentially more responsive to rapid change and to automatic learning.

■ Disadvantage

- black-box model
- does not allow inspection and debugging of knowledge

Different types of agents

- Agents exist in a multi-dimensional space
- **A representative flat-list**
 - Collaborative agents
 - Interface agents
 - Mobile agents
 - Information/Internet agents
 - Reactive agents
 - Hybrid agents
 - Smart Agents
- **Collaborative Agents**
 - These emphasize autonomy, and collaboration with other agents to perform their tasks.
 - They may need to have “ *social* ” skills in order to communicate and *negotiate* with other agents.

Collaborative Agents

- example
 - Pleiades Project at CMU.
 - Visitor-Hoster:
 - helps a human secretary to plan the schedule of visitors to CMU
 - matches their interests with the interests and availability of the faculty and staff.
 - organized as a number of agents that retrieve the relevant pieces of information from several different real-world information sources, such as finger, online library search etc.
- Collaborative agents are good for problems
 - too large for a single system,
 - inherently distributed in nature.
 - main challenge
 - coordination among multiple agents, particularly when they are autonomous and heterogeneous.

Interface (Personal) Agents

- Emphasize autonomy, and learning in order to perform useful tasks for their owners.
- Examples
 - personal assistants that handle your appointments
 - Office Agents in Microsoft Office.
- focus is more on interacting with the user
 - "learn" to serve the user better,
 - by observing and imitating the user,
 - through feedback from the user, or
 - by interacting with other agents.
 - The main challenge here is how to assist the user without bothering him, and how to learn effectively.
- normally have a distinctive *personality*,
- **Avatars** are an interesting subclass

Information / Internet Agents

- focus on
 - helping us to cope with the sheer "*tyranny of information*" in the Internet age.
- help to
 - manage, manipulate or collate information from many distributed sources.
- interface agents or mobile agents
- share their
 - respective motivations and challenges
 - functional challenges of managing information.

Why Software Agents?

- Agents are a useful, and sometimes necessary way to build systems.
- Particularly true when one or more of the following scenarios hold:
 - The task can be *automated*, and *delegated* to a software system
 - The task is *very large*, and *modularization* is possible.
 - The *information* needed is *vast*, and/or *widely distributed*, as with the Internet.
 - The application or service *needs to learn* and improve with time, or be *customized* for each user.

Example domain: E-commerce

- Many e-commerce tasks have one or more of these features
- Agents: a key technology for e-commerce.
- **buyers**
 - locate relevant goods and services, and to identify the best deals for them
- **sellers**
 - identify prospective customers and their needs,
 - help them to select products and services,
 - customize products and services for them,
 - handle the sale and subsequent customer relation management
- in B2C, C2C as well as B2B scenarios.

Enabling Technologies

- Agents is a highly multi-disciplinary technology combining inputs from
 - Software Technology
 - Artificial Intelligence
 - Networking
 - Human Computer Interaction
 - and even Sociology
 - Management and Economics
 - in addition to the actual domain of the task
 - e.g. Business and Commerce in case of e-commerce

Trends: *OMG Agent Technology Green Paper*

- The growth similar to many earlier technologies
 - such as DBMS, OO and GUI
- Not a single, new, technology
 - integrated application of multiple technologies.
- Not necessarily a new isolated application
 - can add a new set of capabilities to existing applications.
 - may strengthen HCI
- Initially
 - agent functions will emerge within applications,
- Later (with experience)
 - become part of the operating system or application environment.

Trends: OMG Agent Technology Green Paper

- Ultimately (might happen)
 - applications that do not exploit agent support in the operating system will be severely disadvantaged.
- Current state:
 - still an active research area.
 - isolated pioneer products are emerging.
 - full set of technologies are not available.
 - technologies not integrated with one another.
 - no consensus on operating system level support
 - despite hype, not in widespread use, nor has it been widely accepted as an inevitable trend.
 - early adopters who can demonstrate the value

Mobile Code

■ **Definition:**

- Capability to dynamically change the bindings between code fragments and the location where they are executed

■ **Approaches:** (Not a totally new concept)

- Remote batch job submission & use of PostScript to control printers
- Distributed OS led to more structured approach
 - Process Migration
 - Object Migration (Mobile Objects)
- Mobile Code Systems (Mobile Agents)

Process Migration

- Transfer of OS process from one m/c to other
- Migration mechanisms handle bindings between
 - process and execution environment
 - (e.g. open fds, env variables)
- Provide for load balancing
- Most of these facilities provide *transparent* process migration
- Other like Locus provide for some control
 - like external signal or migrate() system call

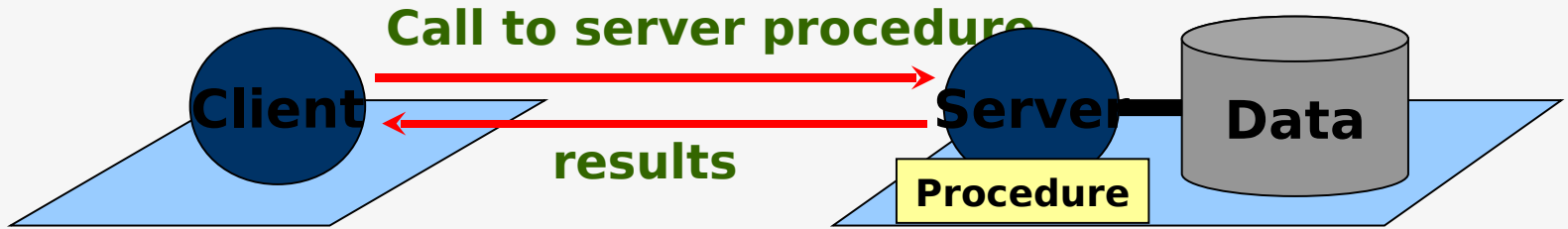
Object Migration

- Makes possible to move objects among address spaces
 - finer grained mobility with respect to processes
 - e..g Emerald system : Different granularity levels - small to complex objects
 - does not provide complete transparency
 - COOL (oo extension of Chorus OS) allows total transparent migration
- Process and Object migration address issues when
 - code and state are moved among hosts of loosely coupled, small scale distributed systems
 - insufficient when applied to large scale settings

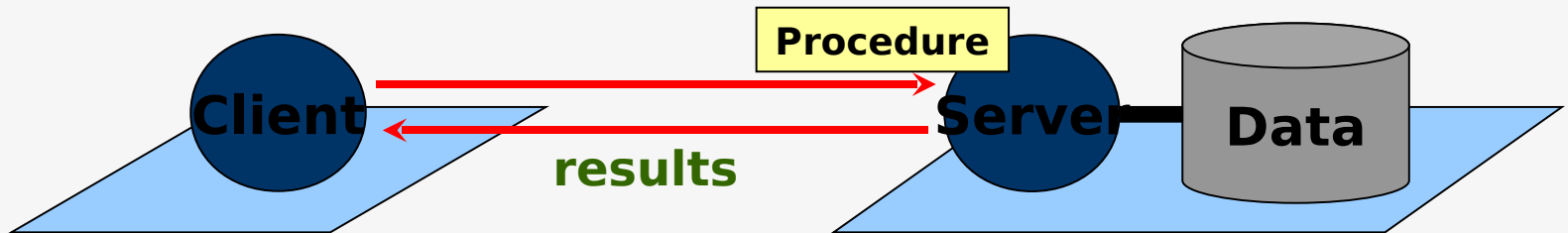
Mobile Code Systems

- Code mobility is exploited on Internet Scale
 - Large scale, heterogeneous hosts, technologies
 - Strong v/s weak mobility
- Mobility is location aware
 - Programming language
 - provides mechanisms and abstractions that enable shipping/ fetching of code to/from nodes
 - Underlying run-time
 - supports marshalling, code, check in , security etc
 - no knowledge of migration policies
- Applications
 - Not just for load balancing
 - E-commerce, distributed information retrieval, workflow etc.

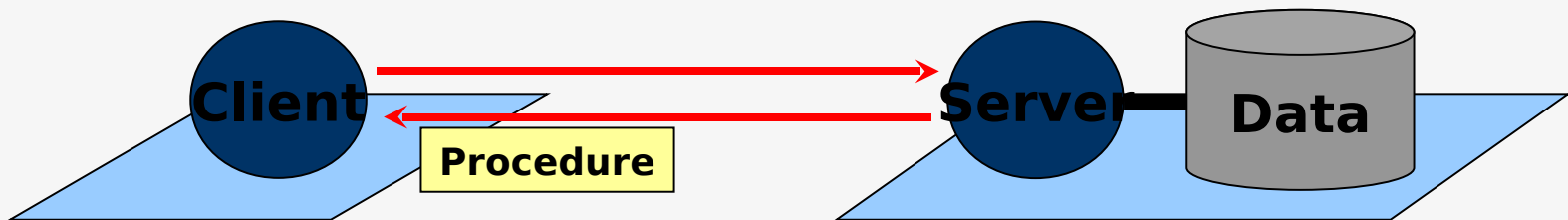
Distributed System Structuring Mechanisms



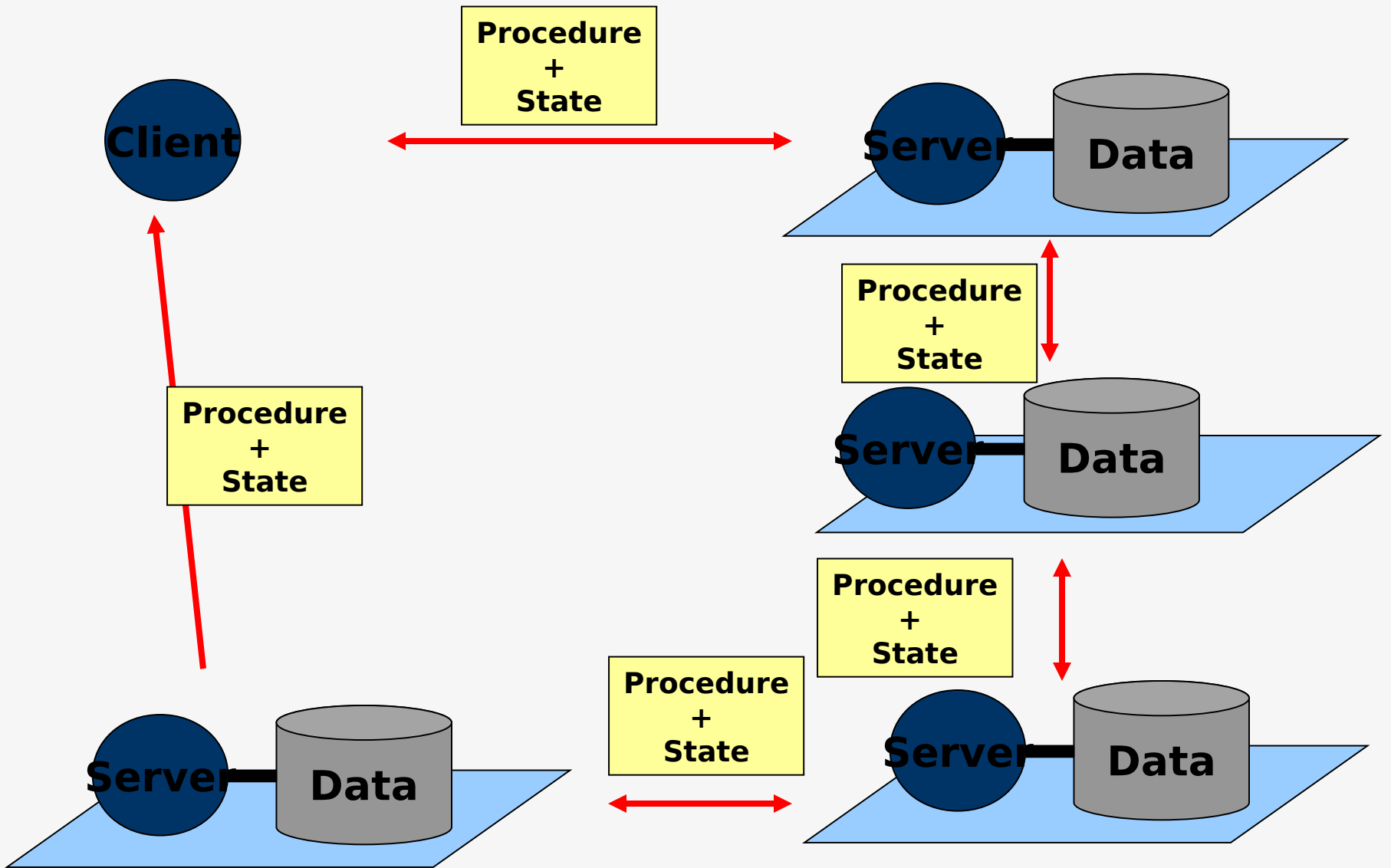
Client Server



Remote Evaluation



Code on Demand



Mobile Agents

Remote Evaluation v/s MA

- A **one hop** solution
- Mobile agents an extension of REV
- REV involves just a movement of code from one host to a host which is capable of carrying out the process where as for MA we have active entities been shipped over the network (data, code and execution state).
- Performance wise it both will have same performance except that MA's execution environment is comparatively heavy.

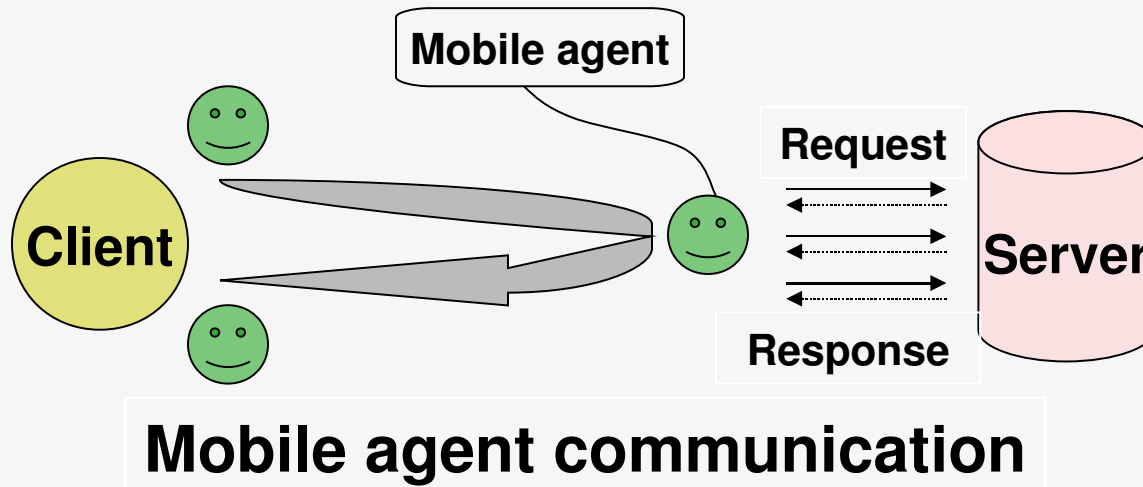
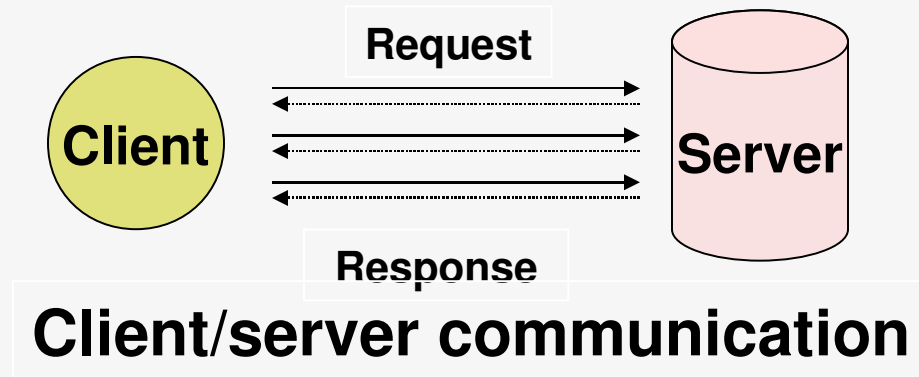
Process migration v/s MA

- Not the desire of the process to move, but the desire of the distributed operating system to have it moved
- Mainly to improve load distribution, fault resilience, parallelism and so on.
- The programmer (user) has neither control no visibility of migrating process.

Mobile Agents: Example



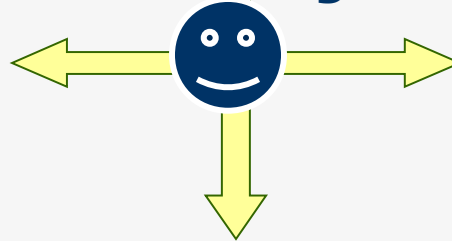
Interaction Model



A generic Mobile Agent Server

- Event notification
- Agent collaboration support

**Event
Manager
Mobile Agent**



- Execution environment
- Communication (agent dispatching)
- Agent life cycle (creation, destruction)

**Agent
Manager**

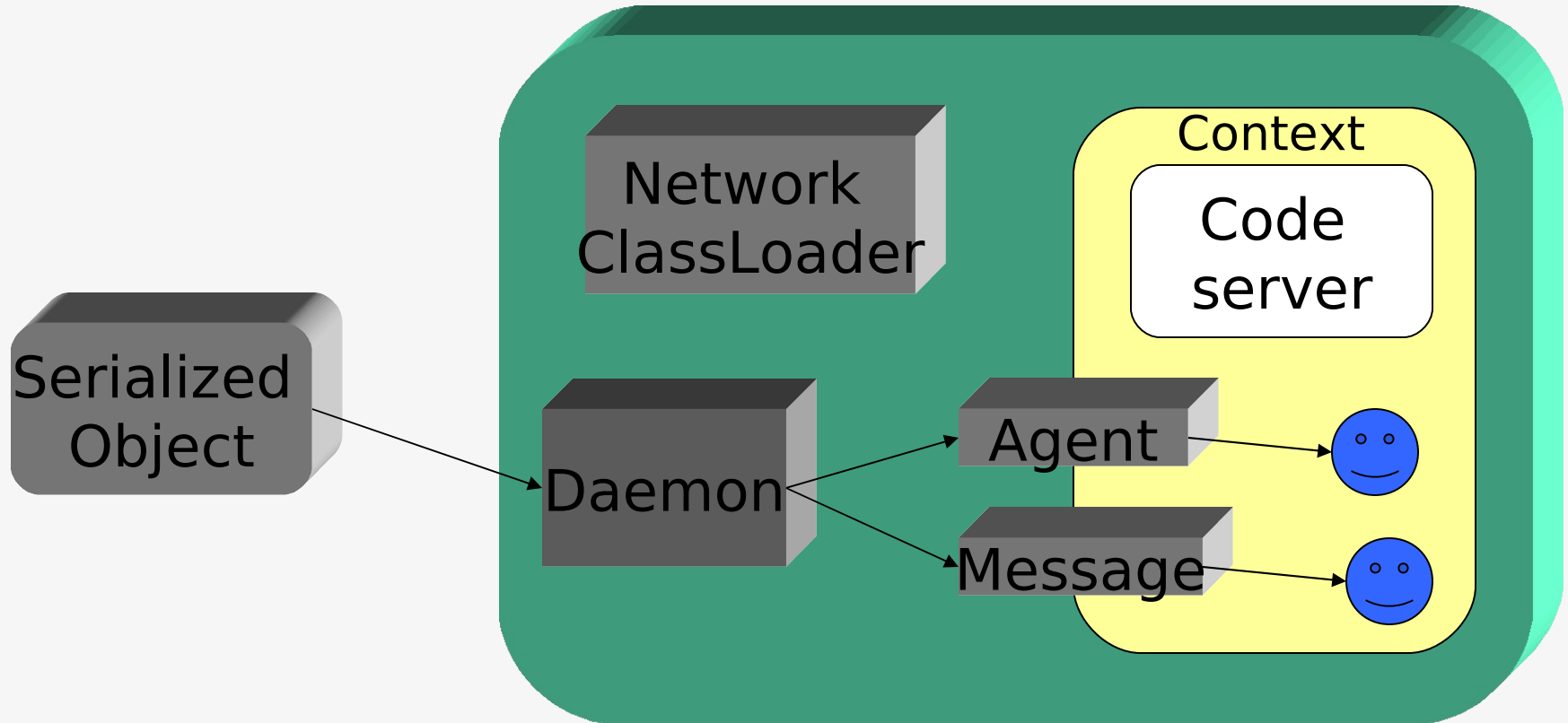
- User identification
- Protection (agent, server)
- Authentication

Security Manager

- Agent state
- Agent checkpoint (fault tolerance)

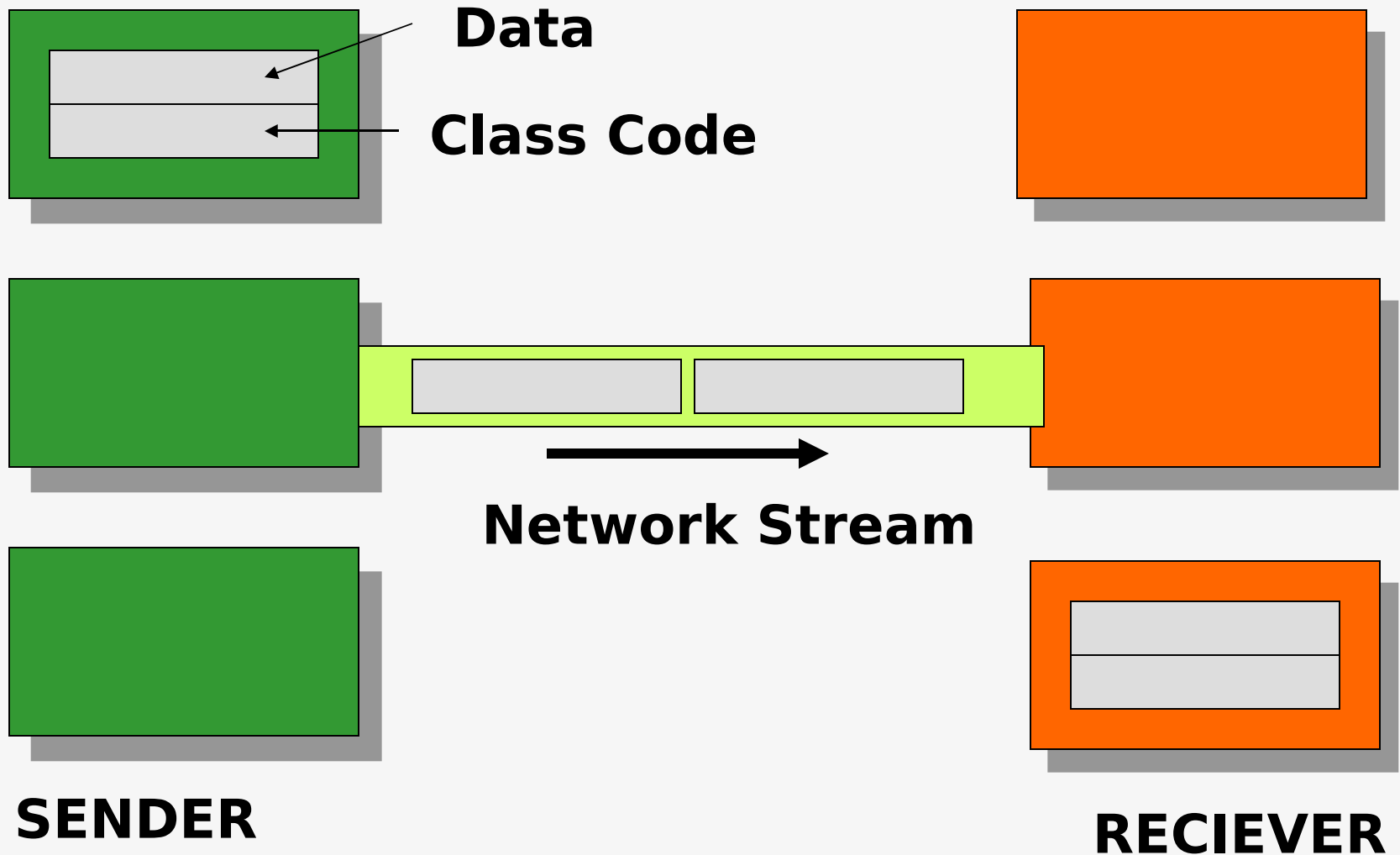
Persistent

Java based Agent server



Java-based Agent server

Agent Transfer



Bag of a traveling agent

Agent Source

```
public class MyAgent extends Agent {
    private String      name;
    private Vector      someData;
    private AgentObject someObject;
    private AnotherObject anotherObject;

    public void method1() {
        . . .
    }

    public void method2 () {
        . . .
    }
}

class AgentObject {
    private String data;

    void calculate() {
        . . .
    }
}

class AnotherObject {
    private Integer data;

    void someMethod() {
        . . .
    }
}
```

Travelling Agent (Concordia platform)

Agent State

	Info
String	name = "agent name";
Vector	someData = ... ;
AgentObject	someObject = ...;
AnotherObject	anotherObject = ...;

Agent Byte Code

MyAgent.class

AgentObject.class

AnotherObject.class



MA based Structuring

Who is affected?

and look for:

- Designers
 - ‘metaphor’ which best captures the problem and provides a neat solution
- Implementers
 - Ease of implementation, testing and extension
- Users
 - solution and performance
 - system which is easily deployed, easy to use and maintain
 - and possibly fun to work with 😊

5 Steps to MA based structuring

- Step 1: Application Evaluation
 - Which application is a good candidate for MA based design ?
- Step 2: Component Design
 - Mobile v/s static components
- Step 3: Choosing (designing?) Framework
 - Underlying mobility support
- Step 4: Detailed Component Design
 - Component placement + Management
- Step 5: Implementation and Deployment
 - Coding, testing and infrastructure requirements

Realizing Step 1:

Application Evaluation

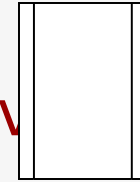
- Case 1:
 - You have an application and you want to test its candidacy for MA based design
- Case 2:
 - You want to invent an application that best exploits MA paradigm
- In both cases: Required understanding of
 - advantages that MA's bring
 - Issues that they raise



metaphor [G.Picco]

Two friends Asha and Latha interact to make a cake (**results of service**)

- recipe is needed (**know-how about service**)



- also ingredients (**movable resources**)

- oven to bake (**hard to move resource**)



- a person to mix ingredients following recipe (**a computational component responsible for execution of code**)

- prepare cake (**execute the service**)

- where cake is prepared (**site of execution**)

The Client-Server Paradigm

- Asha would like to have chocolate cake, but
 - she doesn't know the recipe
 - she does not have at home either the required ingredients or an oven.
 - Fortunately, she knows that her friend Latha knows how to make a chocolate cake, and that she has well supplied kitchen at her place. Since Latha is usually quite happy to prepare cakes on request,
- Asha phones Latha asking: *“Can you make a chocolate cake please?”*.
- Latha makes the chocolate cake and delivers it back to Asha.

Remote Evaluation

- Asha wants to prepare a chocolate cake.
 - She knows the recipe
 - She has at home neither the required ingredients nor an oven.
 - Her friend Latha has both at her place, yet she doesn't know how to make a chocolate cake.
 - She knows that Latha is happy to try new recipes
- She phones Latha asking, “*Can you make me a chocolate cake? Here is the recipe: Take 3 eggs...*”.
- Latha prepares the chocolate cake following Latha's recipe and delivers it back to her.

Code on Demand

- Asha wants to prepare a chocolate cake.
 - She has at home both the required ingredients and an oven
 - She lacks the proper recipe.
 - However Asha knows that her friend Latha has the right recipe and she has already lent it to many friends.
- Asha phones Latha asking, “*Can you tell me your chocolate cake recipe?*”.
- Latha tells her the recipe and Asha prepares chocolate cake at home.

Mobile Agents

- Asha wants to prepare a chocolate cake.
 - She has the right recipe and ingredients,
 - She does not have the oven at home.
 - However she knows that her friend Latha has an oven at her place, and that she is very happy to lend it.
- So, Asha prepares the chocolate batter and then *goes to Latha's home, where she bakes the cake*

Good Reasons [Dennis Lange]:

- Reduce the network load
- Help in overcoming Network latency
- Encapsulate protocols
- Execute asynchronously and autonomously
- Adapt dynamically
- Naturally heterogeneous
- They are robust and fault-tolerant

Realizing Step 2

Component Design

- OO principles still apply
- Two aspects that affect design
 - Autonomous entities
 - What advantage do they bring?
 - Mobile Components
 - Does it make sense to move the component ?
 - What is good mobile component?
- Question 1: To move or not to move?
- Question 2: Passive or Active Mobility?

Classical MA definition:

- “A mobile agent is a program that represents a user (or user task) and can *autonomously* migrate between the various nodes of a network to perform computations on her behalf”
- Much powerful paradigm
 - Need not be restricted to above definition
 - Can/should be extended to include
 - MAs which work in background
 - MA that provide structuring glue

Mobility Patterns

- Itinerary
- Order
- Static Itinerary Dynamic Order (SIDO)
- Static Itinerary Static Order (SISO)
- Dynamic Itinerary (DI)
 - Dynamic Itinerary implies dynamic order

Mobility Patterns

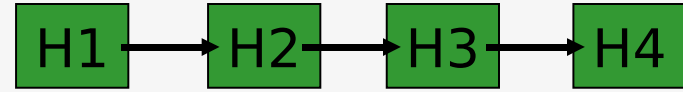
Definitions

- **Itinerary** the set of sites that an MA has to visit
 - static
 - dynamic
- **Order** the order in which an MA visits sites in its itinerary.
 - static
 - dynamic

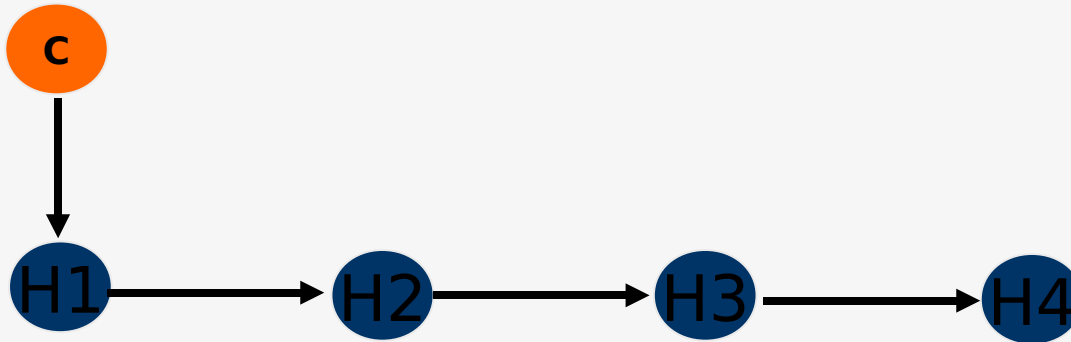
Static Itinerary Static Order



Itinerary



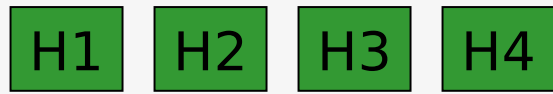
Order



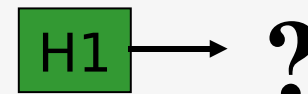
Applicable Implementation Strategies

- Sequential CS
- Sequential MA
- ~~Parallel CS~~
- ~~Parallel MA~~

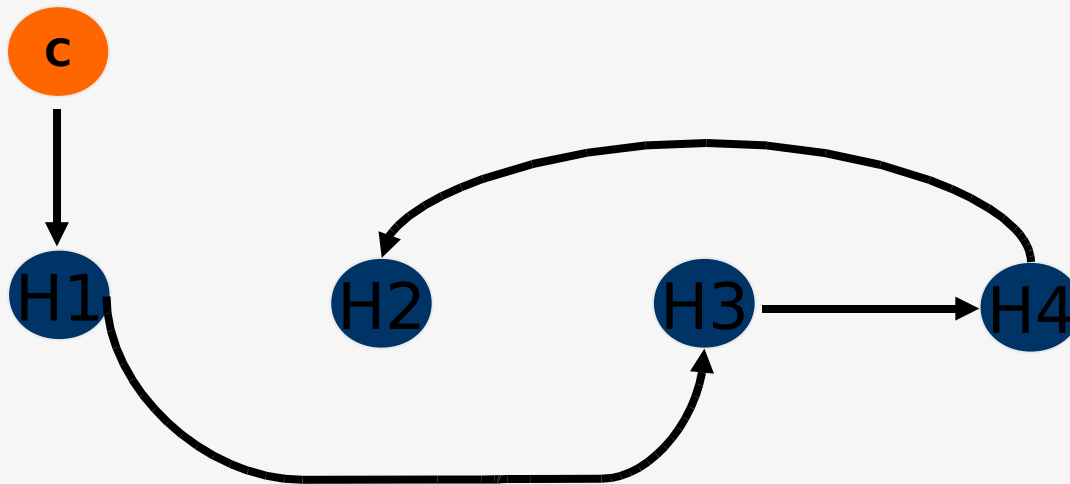
Static Itinerary Dynamic Order



Itinerary



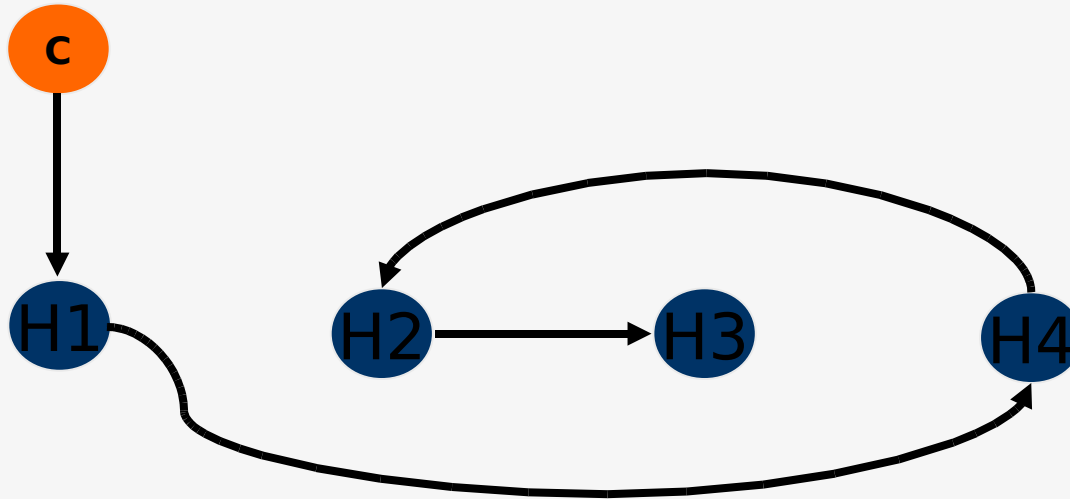
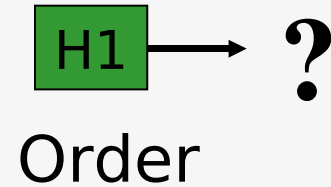
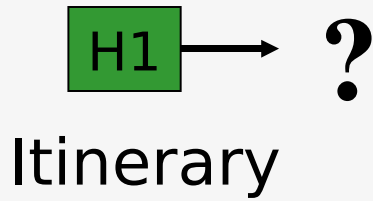
Order



Applicable Implementation Strategies

- Sequential CS
- Sequential MA
- Parallel CS
- Parallel MA

Dynamic Itinerary



Applicable Implementation Strategies

- Sequential CS
- Sequential MA
- ~~Parallel CS~~
- ~~Parallel MA~~

MA Applications

- Electronic Commerce
- Personal Assistance
- Secure Brokering
- Distributed Information Retrieval
- Telecommunication networks services
- Workflow Applications and groupware
- Monitoring and notification
- Information Dissemination
- Parallel Processing

Realizing Step 3

Choosing a MA framework

- Understanding what a MAF provides
- Two aspects that affect design
 - Autonomous entities
 - What advantage do they bring?
 - Mobile Components
 - Does it make sense to move the component ?
 - What is good mobile component?
- Question 1: To move or not to move?
- Question 2: Passive or Active Mobility?

Mobile Agent Frameworks

■ Need

- language, execution environment, messaging, resources, migrate, persist, collaborate, control, trace, protect, create, destroy etc.
- Framework is the mechanism to support these facilities

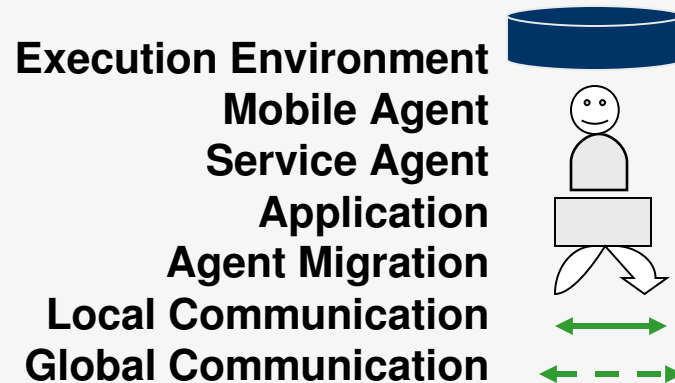
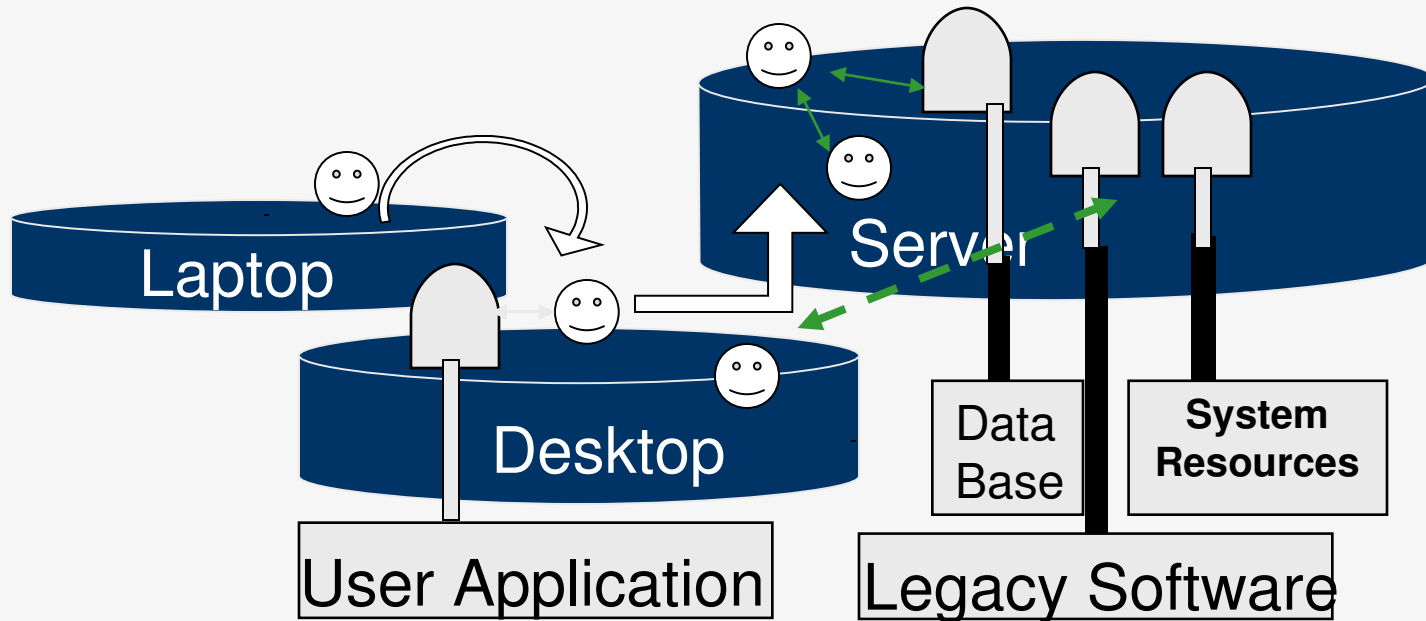
■ Components

- Life Cycle
- Navigation
- Communication
- Security

■ Systems

- 60+ frameworks
- Notable: Aglets, Concordia, Voyager, Grasshopper, D'Agents, Mole

Typical Mobile Agent Framework [F. Hohl]



Mobile Agent Frameworks

Design Issues

Mobility

■ Weak Mobility

- Permits mobility of code and data state
- After the movement, the agent is restarted and the values of its variables are restored, but its execution restarts from the beginning of a given procedure (a method in case of objects).

■ Strong Mobility

- Mobility of code, data state and execution state
- Restart the execution exactly from the point where it was stopped before movement.

Mobility support in Java

- Dynamic class loading, Applets
- Weak mobility could be implemented by serializing objects and sending them to another JVM via sockets or RMI.
- Restored at the other end and a method is called (Ex run(); onArrival())
- JVM from SUN does not support a strong kind of agent mobility

Problems with strong mobility in Java

- Java stack and the program counter of the thread(s) need to be moved
- Each microinstruction in the stack, whose elements are of a generic type `stack_item`.
- Since it is written in C language, it is not assured that the same type has the same internal representation, in terms both of number of bytes and order of bytes (little or big endian)

Code fragment for weak mobility

```
void main(String args[]) {  
    ...  
    // some instructions  
    go("NewNode", "NewMethod");  
    // not reached  
} //end of main  
void NewMethod() {  
    // the execution restarts HERE  
    ...  
} //end of NewMethod
```

Code fragment for strong mobility

```
void main(String args[]) {  
    ...  
    // some instructions  
    go("`NewNode");  
  
    // the execution restarts HERE  
    ...  
} //end of main
```

Repetitive job using weak mobility

```
public static void main(String args[]) {
    ... // go to the first node
    go(Itinerary.nextElement(), ``ExecuteOnArrival");
}

public void ExecuteOnArrival() {
    // execution restarts HERE after a travel
    if (GoHome)
        ... //execute here when the agent is back home
    else {
        ... //do some repetitive jobs on the current node
        if (Itinerary.hasMoreElements())
            go(Itinerary.nextElement(), ``ExecuteOnArrival");
        else {
            GoHome = true;
            go(HomeNode, ``ExecuteOnArrival");
        }
    }
}
```

Repetitive job using strong mobility

```
public static void main(String args[]) {  
    ...  
    while (Itinerary.hasMoreElements()) {  
        go(Itinerary.nextElement())  
        // execution restarts HERE after a travel  
        ... // do something on the current node  
    }  
    go(HomeNode);  
    ... // execute here when the agent is back home  
}
```

Code Shipping

- Carried by the agent
 - Any type agent can run anywhere
- Pre-installed on destination host
 - Less run time transfer overhead
 - New types cannot be added at run-time
 - When and how would you pre-install ?
- Available on code-base server
 - Easy to maintain
 - Location of code-base server ?

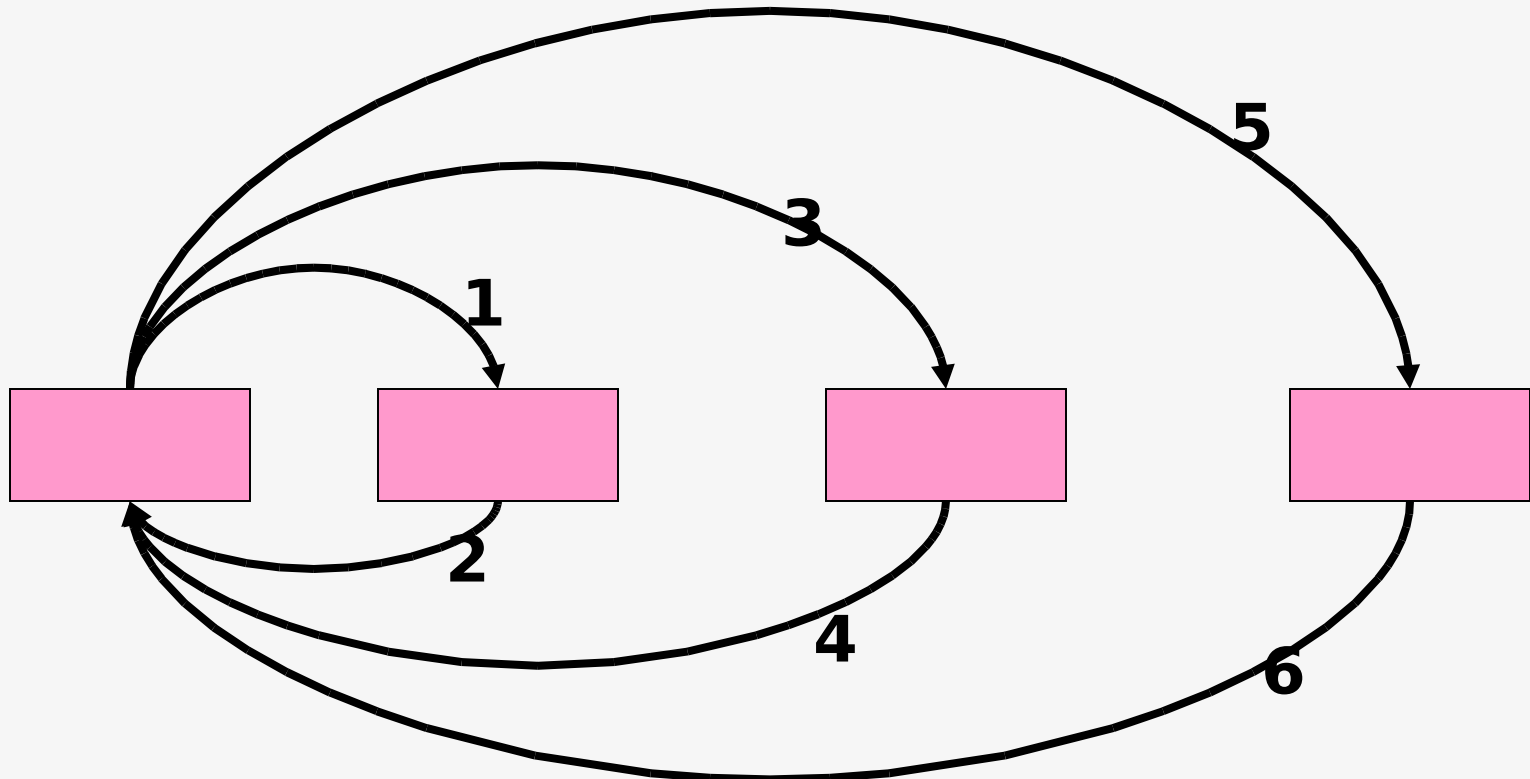
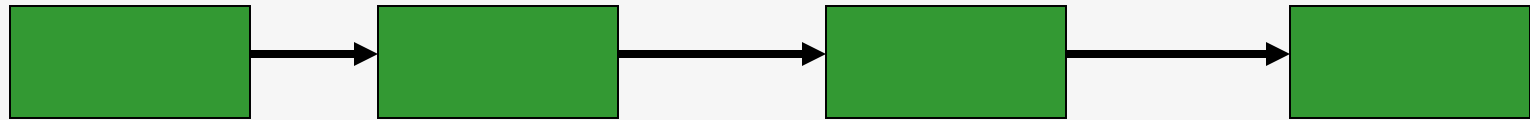
Naming and Addressing

- Location dependent
 - e.g. <hostname> + <local id/ port no>
 - when agent migrates its name changes
 - application task of tracking the agent becomes cumbersome
- Location independent
 - system has to keep track of the agent
 - **local proxies** with current location information
 - **naming service** which resolves name to current location

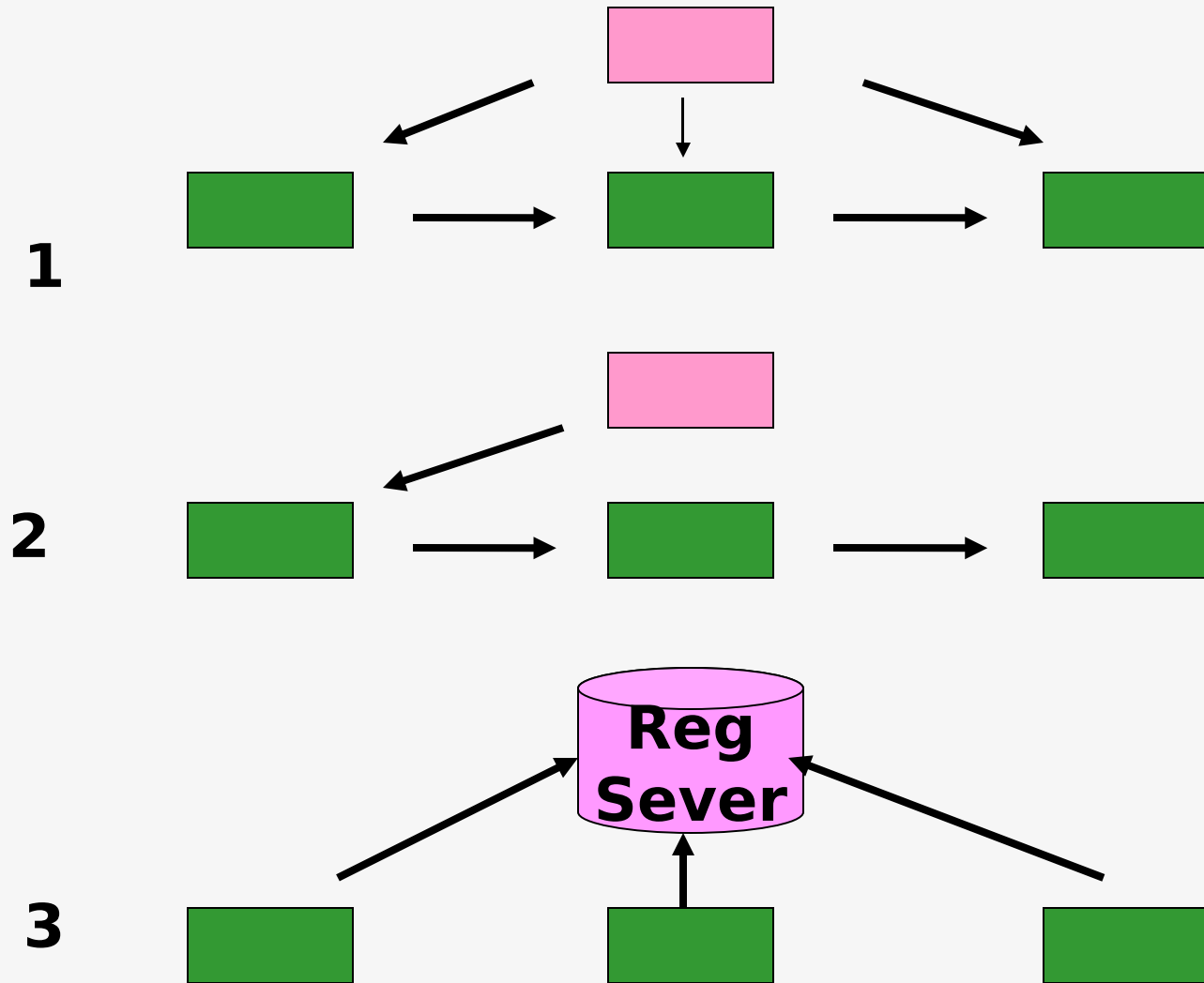
Agent Tracking: locating an agent

- Brute force
 - Search in multiple location
 - Sequential or parallel search
- Logging
 - Agent located by following trail information
 - Tracking
 - Redirection
- Registration
 - Communicating parties need to agree on a common Directory Server
 - Agent updates information in a Directory Server
 - Other agents use this information to locate
 - Useful when unknown parties have to communicate

Logging: Tracking and Redirection



Bruteforce(1,2) registration (3)



Message delivery

messaging an agent

- Locate and transfer
 - Two separate phases are used
 - More efficient if messages are big
 - May not always be accurate
- Forwarding
 - Single phase
 - More efficient if messages are small

Communication Mechanisms

- Method invocation
 - Call method on another object
 - Parameters and return values
 - Achieved by
 - Direct reference to the method (same address space)
 - LPC (object on local host)
 - RPC (object on remote host)
- Message passing
 - Message encapsulates the protocol
 - Parsed and interpreted

Communication Mechanisms

- Black board
 - Interactions via shared-data spaces local to each EE
 - Need for common message format/identifier understood by each agent
 - Temporal uncoupling
 - When you cannot create/predict a agent schedule
- Tuple spaces
 - Extension of black-board model
 - Information stored in tuple-space
 - Retrieved by associative pattern-matching
 - Useful as MAs have to adaptively deal with
 - Heterogeneity, dynamicity, uncertainty
 - Mechanism for agent coordination
 - Simplifies programming and reduces complexity of application

Type of interactions

- MA-Execution Environment
 - MA needs services like transport, file, naming
 - EE needs to control and track the agent
 - Client-server (request-response)
 - RPC like mechanism
- MA-MA
 - Peer-peer patterns
 - Agent has its own agenda (needs and goals)
 - Message passing mechanism more suitable
 - Higher level communications may be used
 - KQML / KIF
- MA-User
 - Act on behalf of user
 - Report result back to user
 - Interaction usually through a GUI
 - Details of Human-Computer Interaction

Communication

Other features

- Event Handling
 - Anonymous communications are supported
 - Event Handling service
 - Suppliers: generators of events
 - Consumers: user of events
 - Event Channel
 - Decouples the system
- Group Communications
 - Broadcast, multicast, anycast
 - Application need / hierarchy for system administration purpose

Issues

- Message ordering
 - When agents move rapidly
 - Out of order messages
 - Need for higher level-protocol over simple message delivery
 - Sequence Number overhead

Issues

- Double Identity
 - Agents migrating to different host might get different names / identities
 - Makes certain operations difficult
 - E.g. if secure channel is set up between two agents
 - If change in place, how do you ensure that new agent is not an imposter of the previous one
- Agent Tracking
 - After being located, the agent can move
- Lost Agents
 - Agent might disappear without deregistering
 - Provide monitors on agent-handles

Security Issues

Attacked	Type of Attack
Host	Host compromised by arriving agent
Host	Host compromised by external third party
Agent	Agent is compromised by another agent or Host
Agent	Agent is compromised by third party
Network	Network compromised by incoming agent

Security: Agent to Host

- Exploit security weakness of host
- Execute programs from potentially untrusted sources
- Masquerading
 - Take identity of another agent
 - To get unauthorized access
 - To shift blame for actions
- Denial of Service
 - Consume excessive amount of computing resources
 - May be caused by bugs in the code

Security: Agent to Host

- Unauthorized Access
 - Access control mechanisms
 - Resource allocation done according to platform (host) policy
 - Agent is issued privileges based on authentication
 - How to authenticate an agent which has visited many untrusted hosts?

Security: Host to Agent

- Most difficult to detect and prevent
- Host has full access to agent data and code
- Masquerading
 - Posting as another host
 - e.g. make a buyer agent believe that others are charging more
- Denial of Service
 - Ignore service requests
 - Introduce unacceptable delays
 - Terminate agent without notification
 - Deadlock other agents / platforms
 - Livelock by generating more work continuously

Security: Host to Agent

- Eavesdropping
 - Classical threat in electronic communication
 - More serious in MA systems as agent platform can
 - Monitor communications
 - Read every instruction executed by agent
 - Read all unencrypted data
 - May contain proprietary algorithms, trade secrets
 - Infer from service requests
 - E.g. agent communicating with a travel agent
- Alteration
 - Modification of data, state, code
 - Cannot be prevented
 - only detection possible in some cases
 - Typically using digital signature
 - Only for code and static data

Security: Agent to Agent

- Exploit security weakness of other agents
- Masquerading
 - Harms both the attacked agent and the agent whose identity is stolen
- Denial of Service
 - E.g. sending repeated messages to another agent
 - Cause undue burden on message-handling techniques
 - If agent is being charged for resource-utilization
 - Monetary loss
- Repudiation
- Unauthorized Access
 - Get hold of modify sensitive information

Security: Other

- Masquerading
 - collective
- Network Denial of Service
- Copy and Replay

Counter measures

- Convention techniques can be employed
 - Public key cryptography
 - Digital signatures
 - Session keys
- But need adaptation
 - cannot be directly employed
- Some difficulties

Protecting

the agent from Host: some efforts

- Computing with encrypted functions
 - For computational privacy
 - Remote signature without revealing the key
- Environmental key generation
- Partial result encapsulation
- Mutual itinerary recording
- Itinerary recording with Replication and voting
- Obfuscated code
- Cryptographic containers for Data Protection

Protecting the Agent Platform

- Software-Based Fault Isolation
- Safe code interpretation
- Signed code

Mobile Agent Framework

real world examples

- Voyager
- Aglets
- Concordia

Aglets

- Weak mobility
- Event driven programming model (dispatch, onDispatch ..
- Proxies for location transparency



IBM

Voyager

- An ORB supporting mobility
- Built on top of Corba
- Weak mobility
- Federated directory service and multicast support



Concordia

- Weak mobility
- Event driven programming mode
- Uses Java RMI for mobility



Mitsubishi Electric IT

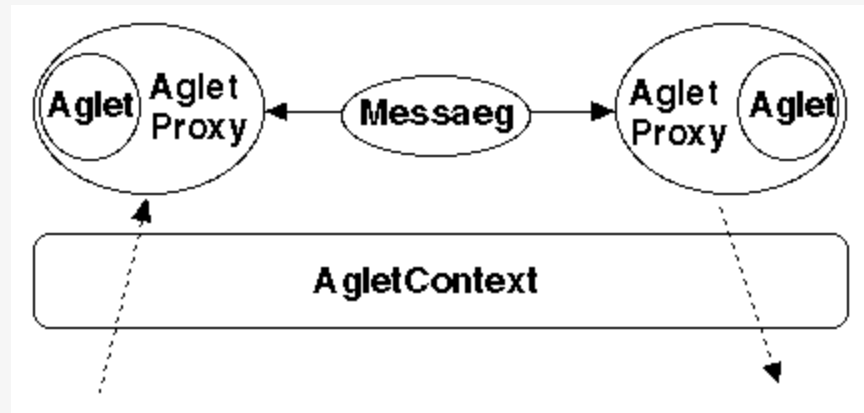
Communication

mechanisms

- Agelts:
 - Java RMI, ATP(Agent Transfer Protocol), CORBA
- Voyager
 - Java RMI, Corba
- Concordia:
 - TCP socket
 - Java RMI

Communication

■ Aglets



- Event, Message-based communication
- Communication is made through Proxy Object
- Group-oriented communication is not available
- A white board mechanism allowing multiple agents to collaborate and share information asynchronously

Communication

- Concordia
 - Supports group communication
 - Group can be formed but it is not possible to join a group arbitrarily
- Voyager
 - Supports scalable group communication
 - Is based on Java reflection mechanism

Feature Comparison

Features	Voyager	Aglets	Concordia
Category	ORB	MA based framework	MA based framework
Multicast	Yes	No	No
Publish/Subscribe	Yes	No	No
Authentication and security	Strong implementation	Weak implementation	Strong implementation
Agent persistence	Yes	No	Yes
Naming service	Federated	No	No
Remote agent creation	Yes	No	No
Garbage collection	Yes	No	No

Writing you own Framework:

RMI 64 example

- Simplistic case
- Uses Java RMI as the base platform

```
import java.rmi.*;
import java.rmi.server.*;

public interface RMI64Server
extends java.rmi.Remote {
    public void runAgent(Agent agent)
        throws java.rmi.RemoteException;
}
```

```
public class RMI64 extends UnicastRemoteObject
implements RMI64Server {
    public RMI64() throws RemoteException {
        super();
    }

    public void runAgent(Agent agent) {
        new Thread(agent).start();
    }

    public static void moveAgent(Agent agent, String dest) {
        try {
            RMI64Server ds=(RMI64Server)Naming.lookup(dest);
            ds.runAgent(agent);
        } catch (Exception e) {
            System.err.println("unexpected exception: "+e);
            e.printStackTrace();
        }
    }

    public static void main(String[] args) {
        try {
            Naming.rebind(args[0], new RMI64());
        } catch (Exception e) {
            System.err.println("unexpected exception: "+e);
            e.printStackTrace();
        }
    }
}
```

RMI 64: Agent Class

```
public interface Agent
extends java.lang.Runnable, java.io.Serializable {
    public void run();
}
```

```
public class HelloAgent implements Agent {
    private Vector places=null;
```

```
    public HelloAgent() { places=new Vector(); }
```

```
    public void run() {
        System.out.println("Hello World");
        if (places.size()==0) {
            System.out.println("terminating...");
        } else {
            String dst=(String)places.elementAt(0);
            places.removeElementAt(0);
            RMI64.moveAgent(this, dst);
        }
    }
}
```

```
    public static void main(String[] args) {
        HelloAgent a=new HelloAgent();
        a.places.addElement("//localhost/rmi64.1");
        a.places.addElement("//localhost/rmi64.2");
        a.places.addElement("//localhost/rmi64.3");
        a.run();
    }
}
```

Standardization efforts

■ MASIF

- Mobile Agent System Interoperability Facility
- From the Object Management Group (OMG)
- Relates MAs to CORBA

■ FIPA

- Foundation for Intelligent Physical Agents
- Defines extensions that are necessary to AMS (Agent management system) to support mobility

MASIF

- Interfaces between
 - Agent systems
 - Not between agent applications and agent systems
- Not language interoperability
- Defines
 - Agent Management
 - Agent Transfer
 - Agent and Agent System Names
 - Agent System Type and Location Syntax

Realizing Step 4

Detailed Component Design

- Security
- Who owns the component ?
- Who will pay ?
- What is the cost of a mobile component failure/ malfunction on the overall system reliability ?
- Interoperability

Realizing Step 5

Implementation and Deployment

- Coding, Debugging and Testing
 - as easy / difficult as any other distributed system development
- Many surprises during the run-time
- Managing run-time entities
- Infrastructure requirements
 - Resource control
 - Agent Environment Uptimes

Application Case Studies

E-commerce

Distance Evaluation

Characteristics

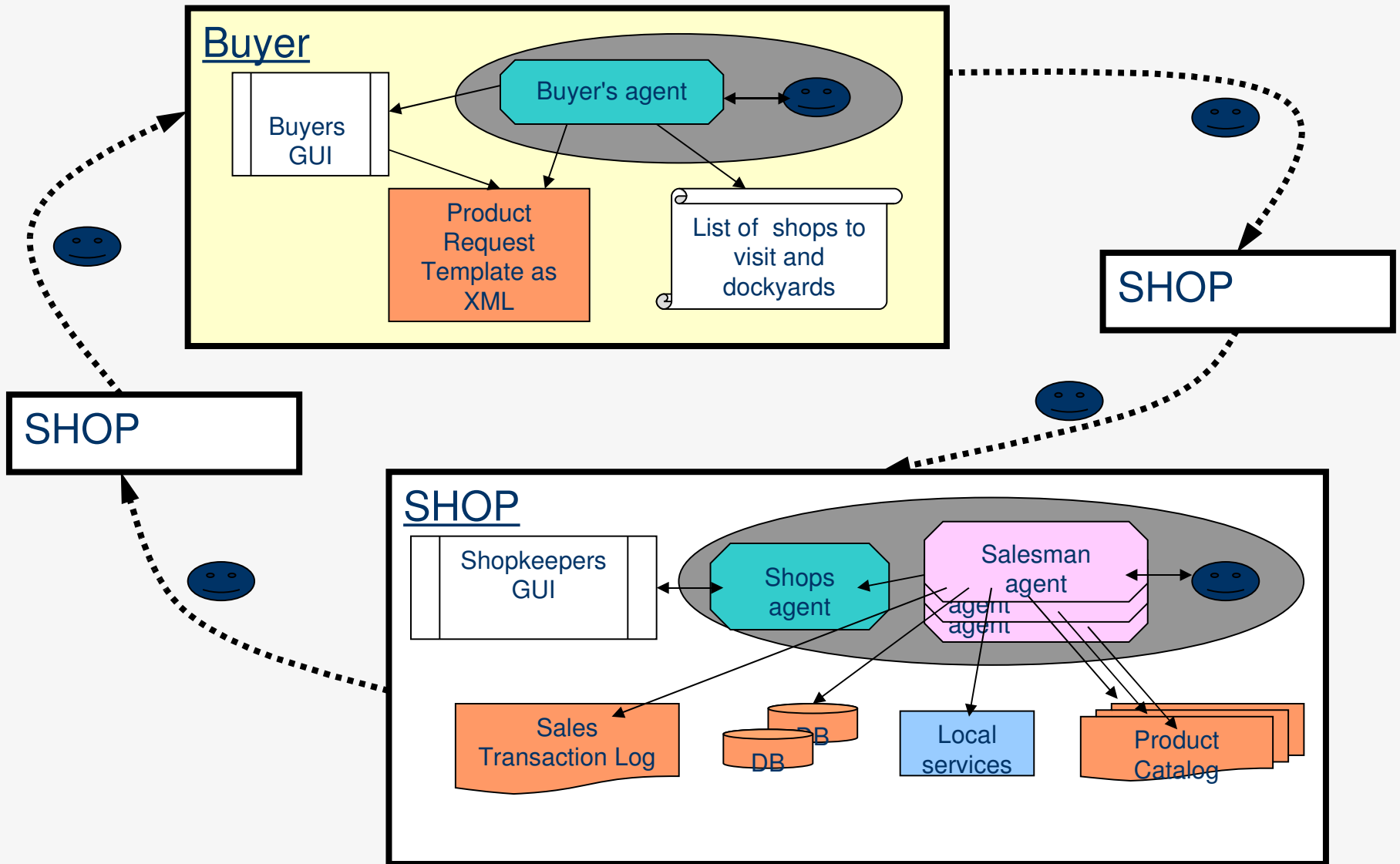
E-commerce applications

- Aim
 - Determine the availability of products; to place and confirm orders and to negotiate delivery.
- Large amount of data exchange over the network in fetching information(catalog)
- Client specific request of products
- To reduce delays that hamper tight interaction
- Disconnected (low B/W) shopping

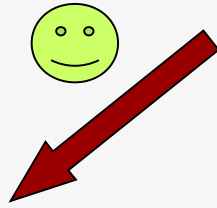
Mobile agents in E-commerce

- Shopping Agent
 - Customer-driven market place
 - Elimination of large amount of information exchange over the network
- Salesman Agent
 - Supplier-driven market place
 - For products with short shelf-life, advertising a product,
..
 - Network delays in servicing orders is reduced
- Auction Agent
 - Supports disconnected operations and quicker response

Architecture e-comm Prototype



Component Interactions



Machine Vision

Owner: rahul@iitb.ernet.in Authors Name: Ramesh

Year of Publication: 1999 Delivery within: 7 Days Category: Networking and OS Level: intermediate

Option: LE Price (Rs): 800

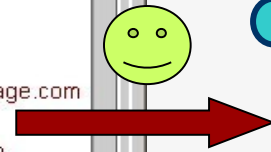
Launch Implementation Strategy: Mobile Agents

- Mobile Agents
- Client Server

Products at Shop

- BOOK
- BOOK
- BOOK
- BOOK
- BOOK
- BOOKID A-11
- TITLE Artificial Intelligence, 2/e
- AUTHORLIST
 - AUTHOR
 - FIRSTNAME Elaine
 - LASTNAME Rich
 - EMAIL Rich@usa.net.in
 - HOMEURL http://www.Elaine.homepage.com
 - AUTHOR
 - FIRSTNAME Kevin
 - LASTNAME Knight
 - EMAIL Knight@usa.net.in
 - HOMEURL http://www.Kevin.homepage.com
- PUBLISHER
 - ISBN 0-07-460081-8
 - PRICE
 - CATEGORY Computer Science
 - LEVEL ext
 - REVIEWS
 - AVAILABILITY

TITLE : Artificial Intelligence, 2/e
AUTHORLIST :
AUTHOR :
FIRSTNAME : Elaine
LASTNAME : Rich
EMAIL : Rich@usa.net.in
HOMEURL : http://www.Elaine.homepage.com
AUTHOR :
FIRSTNAME : Kevin
LASTNAME : Knight
EMAIL : Knight@usa.net.in
HOMEURL : http://www.Kevin.homepage.com
PUBLISHER :
PUBLINE : Tata McGraw -Hill Edition
EDITION : 1
PUBURL : http://www. Tata McGraw -Hill
Edition .com
PUBDATE :
DATE : 02



Filtered Result

Products at Shop

- ProductResult
 - shopName
 - BOOK
 - BOOK
 - BOOK
 - shopName
 - BOOK
 - BOOK
 - BOOK

Machine Vision

by

- Ramesh Jain [Jain@usa.net.in] [Authors Home page](#)
- Rangachar Kasturi [Kasturi@usa.net.in] [Authors Home page](#)
- Brian G.Schunck [G.Schunck@usa.net.in] [Authors Home page](#)

McGraw-Hill International Editions , 7 [Publishers Home page](#) on 02 - 12 - 1999
 ISBN :0-07-113407-7
Price : USD 0.50
Category : Certification Central **Level :** intermediate
Reviews :

Availability : Usually ships with in 8 days
Cover : paper 549pages
Dimension : 7296433 916 x 10.1651080663006 x 1.40979801511158

Customers who bought this book also bought:

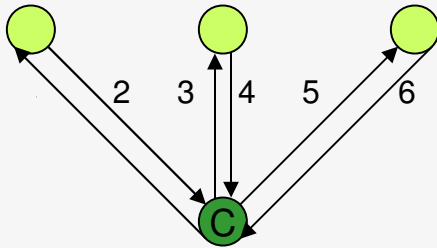
- *Hard times by - Rahul*
- R.David - Tom Mayers
- *similar books by - Rahul*
- R.David - Tom Mayers



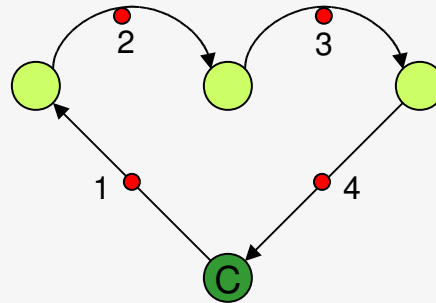
Why MAs?

- Helps user with tedious repetitive job and time consuming activities.
- Faster and real time interaction at shops
- Reduce network load
- Support disconnected operation.
- Introduce
 - concurrency of operations
 - client specific functionalities at the shops

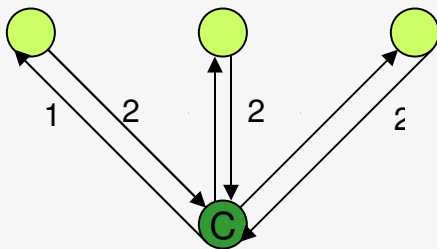
Implementation strategies



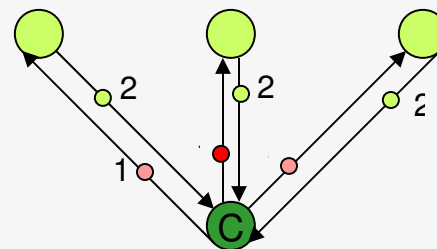
(a) Sequential Client Server



(b) Sequential Mobile Agent



(c) Parallel Client Server



(d) Parallel Mobile Agent



Client



Server



Mobile Agent



Message exchange

1 2 3 4 5 6 Numbers along the arrows indicate the sequence of messages./ MA movement.

Implementation

different mobility patterns

- SISO
 - Sequential CS
 - Sequential MA
- SIDO
 - Sequential CS
 - Sequential MA
 - Parallel CS
 - Parallel MA
- DI
 - Sequential CS
 - Sequential MA

Experimentation

- Experimental setup
 - Voyager™ Framework for MA implementations
 - Java™ socket based implementation for client server interaction
 - On Pentium-III, 450 MHz workstations connected through a 10 Mbps LAN with typical student load

Parameters

Parameters	Range
number of stores	1 to 26
size of catalog	20 KB to 1 MB
size of client-server messages	~ catalog size
processing time for servicing each request	10 ms to 1000 ms
network latencies on different links	assumed constant (all workstations on the same LAN)

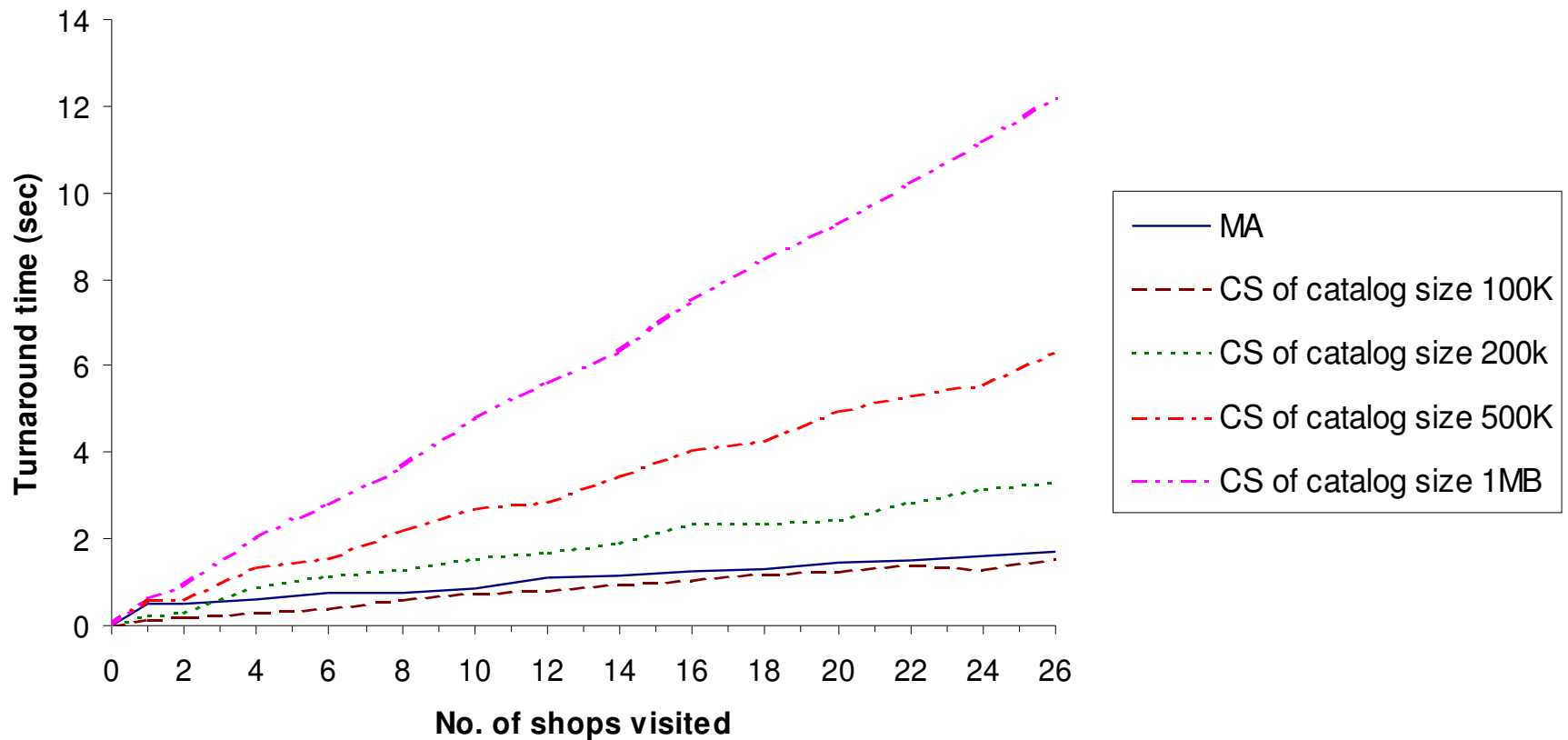
Performance metric

User Turnaround Time

- Time elapsed between
 - a user initiating a request and receiving the results.
- Equals time taken for
 - agent creation +
 - visit / collect catalogs +
 - processing time to extract information.

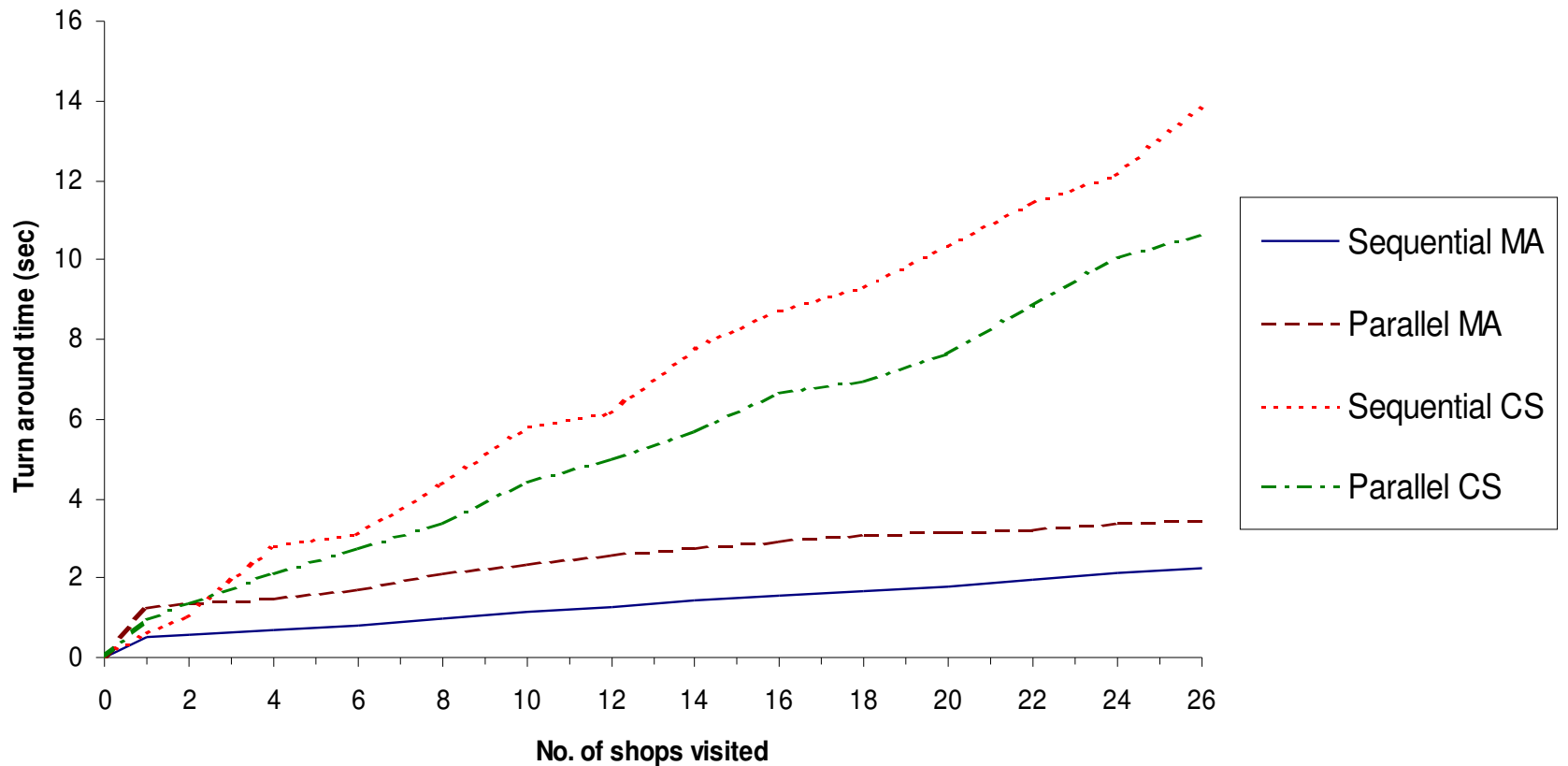
Turnaround time

Effect of catalog size



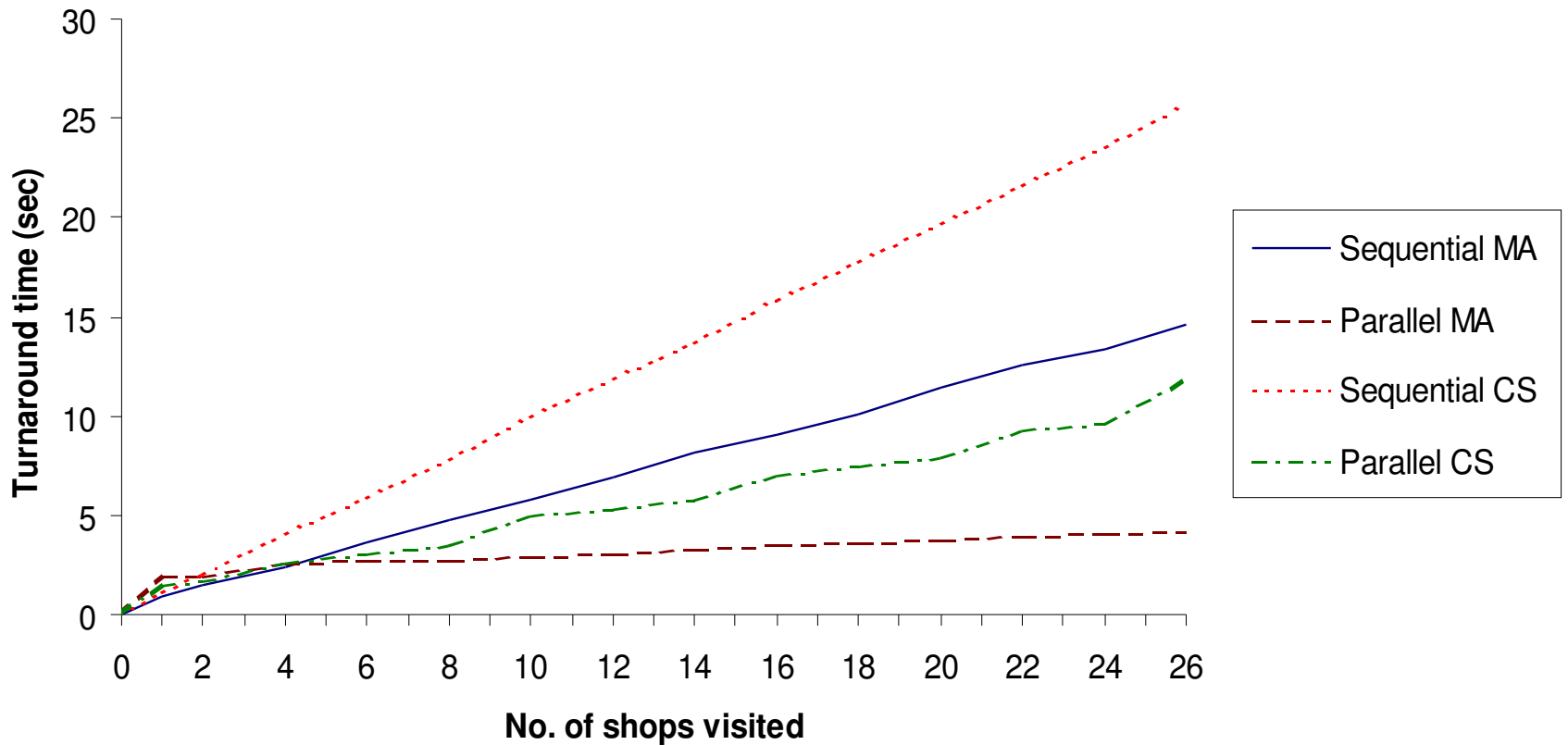
Turnaround time

for processing time of 20ms



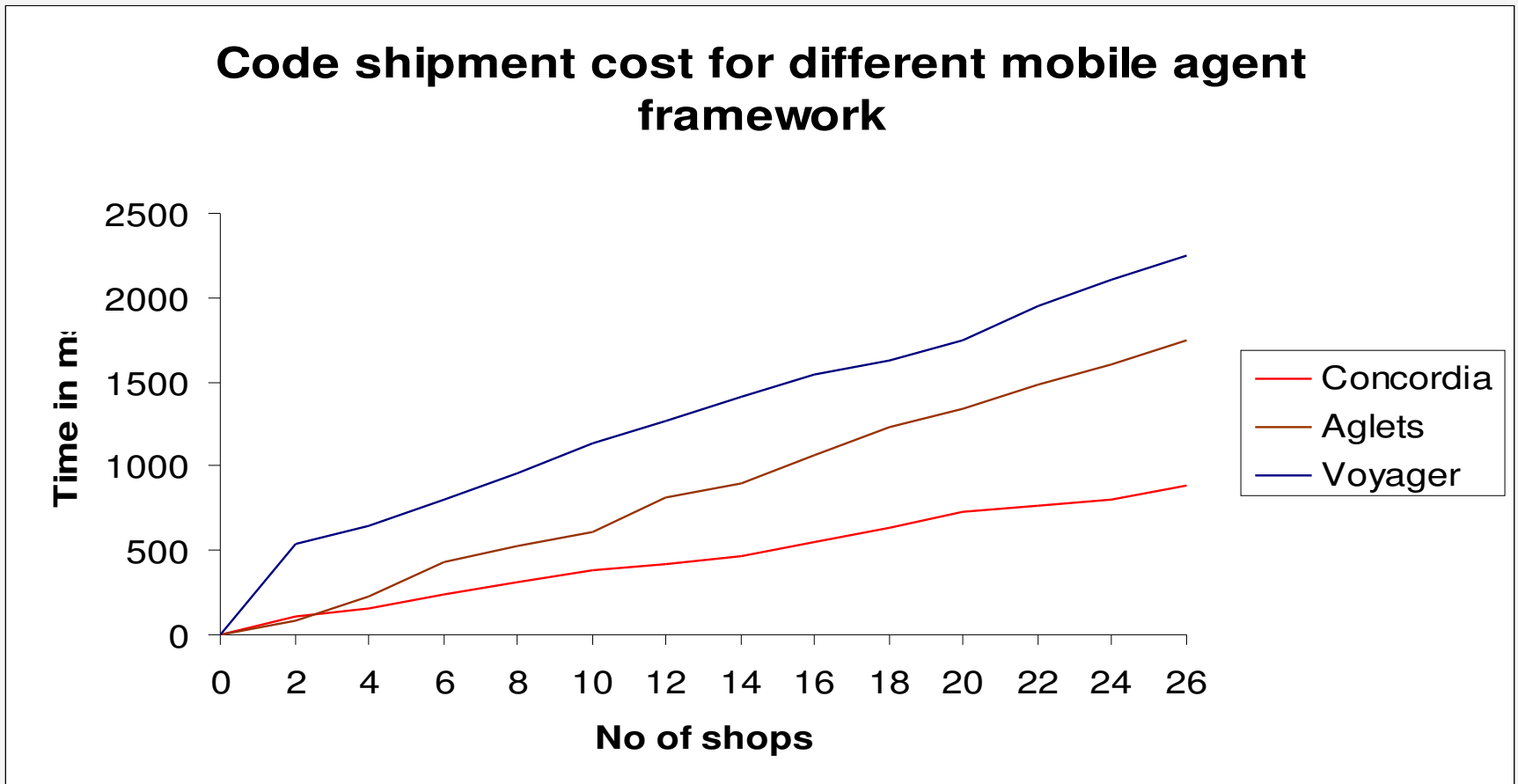
Turnaround time

for processing time of 500ms



Code shipment cost

for different framework



Observations

- Mobility patterns determine the implementation strategies
- Sequential CS most suitable where
 - a small amount of information has to be retrieved from few remote information sources.
- Parallel implementations effective when
 - processing information contributes significantly to the turnaround time.

Observations

- Mobile agents out perform traditional approaches when
 - When the cost of shipping MAs $<$ message exchange size.
- MAs scale effectively across the parameters of E-commerce application

MADE

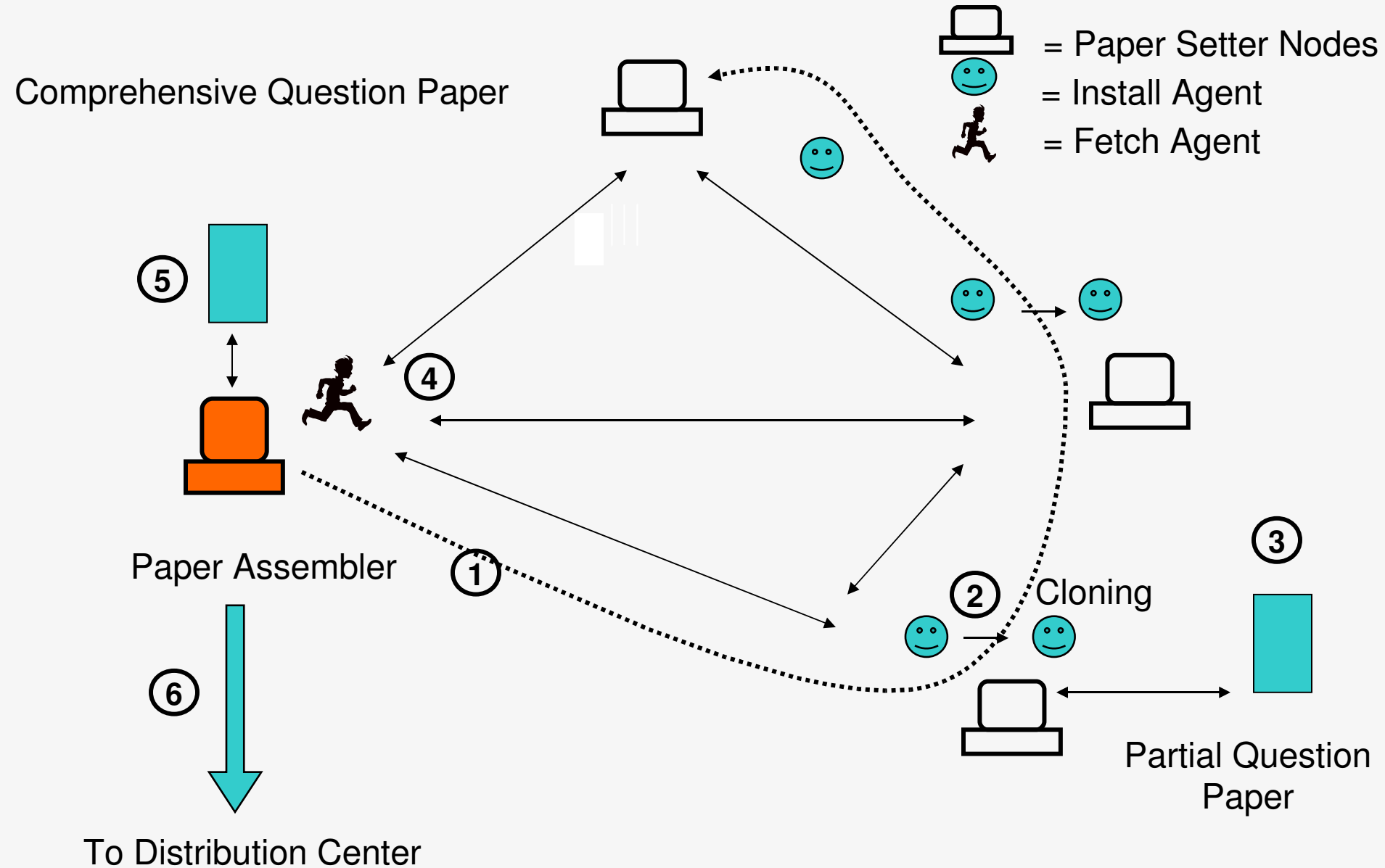
Mobile Agents for Distance Evaluation

Design + Implementation

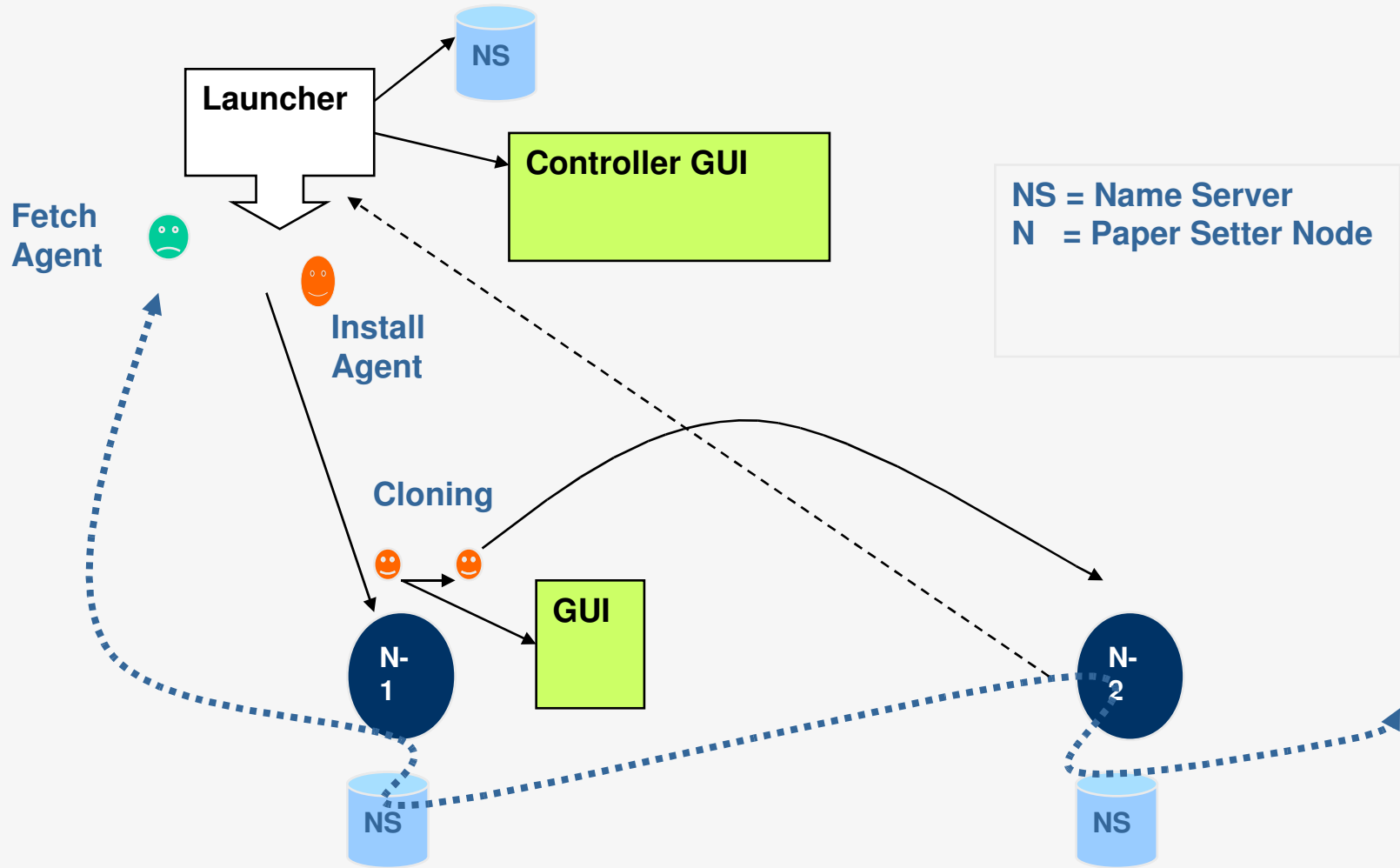
How Mobile Agents Help

- Map directly to real life situations
- Need a generic execution environment
- Can work in both modes
 - push
 - pull
- Can work off-line
- Provide local interactions
- Provide multi-hop solutions

Paper Setting



Paper Setting: Details



Create Questions [Close] [Maximize] [Minimize]

☀️ ⏪ ⏩ OK DEL Finished Q No 1

Enter Your Question

What the name of the weightlifter from India who won the medal in Sydney olympics?

Enter Option 1

P.T.Usha

Enter Option 2

Shiny Abraham

Enter Option 3

Malleshwari

Enter Option 4

Shakti Singh

Correct Option

Four

Three

Create Questions [Close] [Maximize] [Minimize]

☀️ ⏪ ⏩ OK DEL Finished Q No 3

Enter Your Question

Number of Indians who have won Nobel Prize is:

Enter Option 1

4

Enter Option 2

6


Enter Option 3

Enter Option 4

Correct Option

OK

Agent Messages



Should I wait?
Press <WAIT>

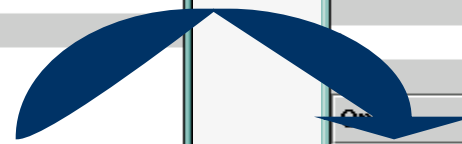
Should I come little later?
Press <LATER>

If you are done with questions,
Press <FINISHED>

WAIT

LATER

Dynamic Upgrade



Create Questions [Close] [Maximize] [Minimize]

☀️ ⏪ ⏩ OK DEL Finished

Enter Your Question

HTTP port is usually one of t

Enter Option 1

20

Enter Option 2

40

Enter Option 3

60

Enter Option 4

80

Correct Option

Four

AGENT

Status Messages here

Paper Collected from...../localhost:5000

Paper Collected from...../localhost:6000



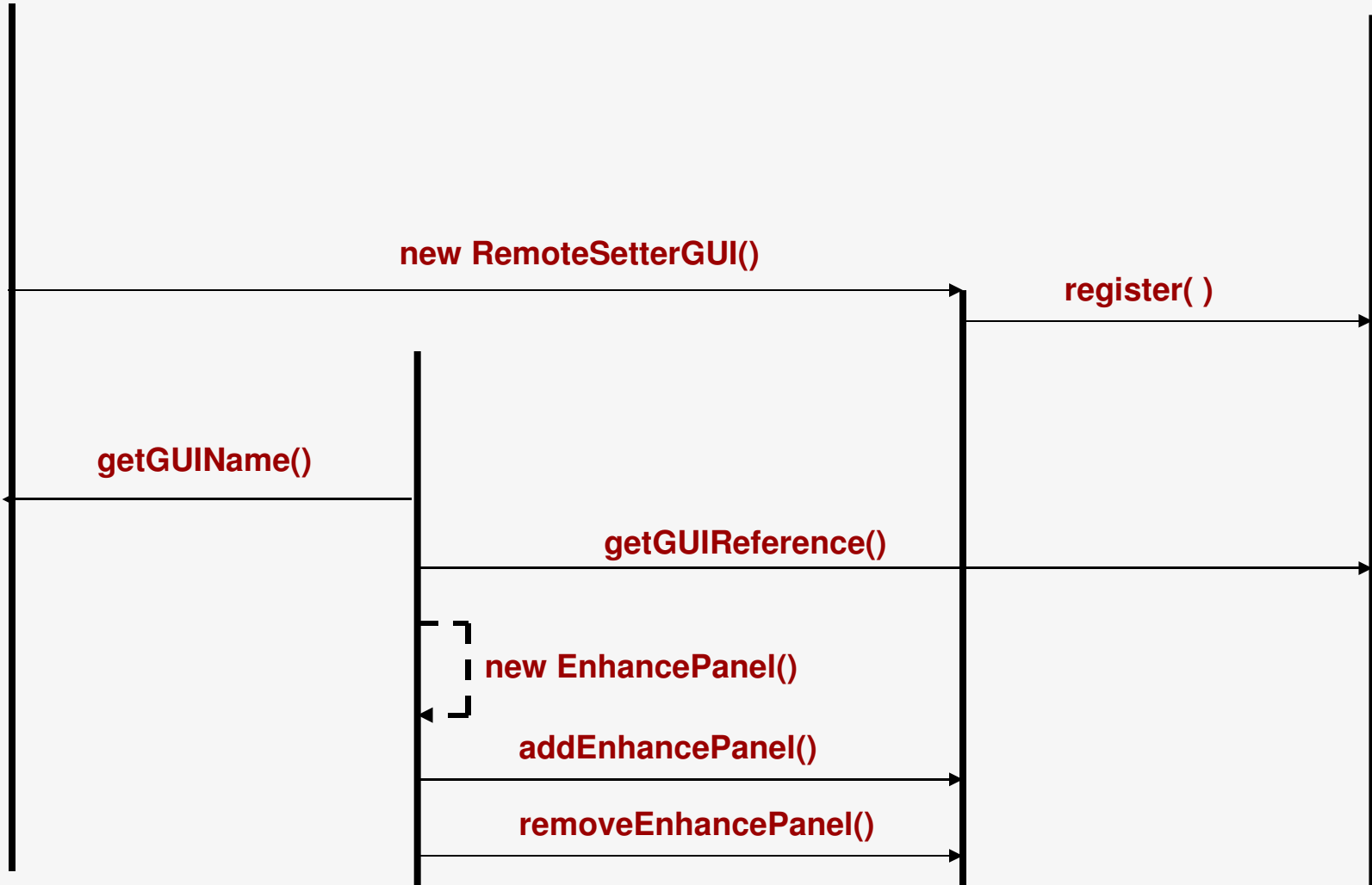
Dynamic Upgradation

InstallAgent

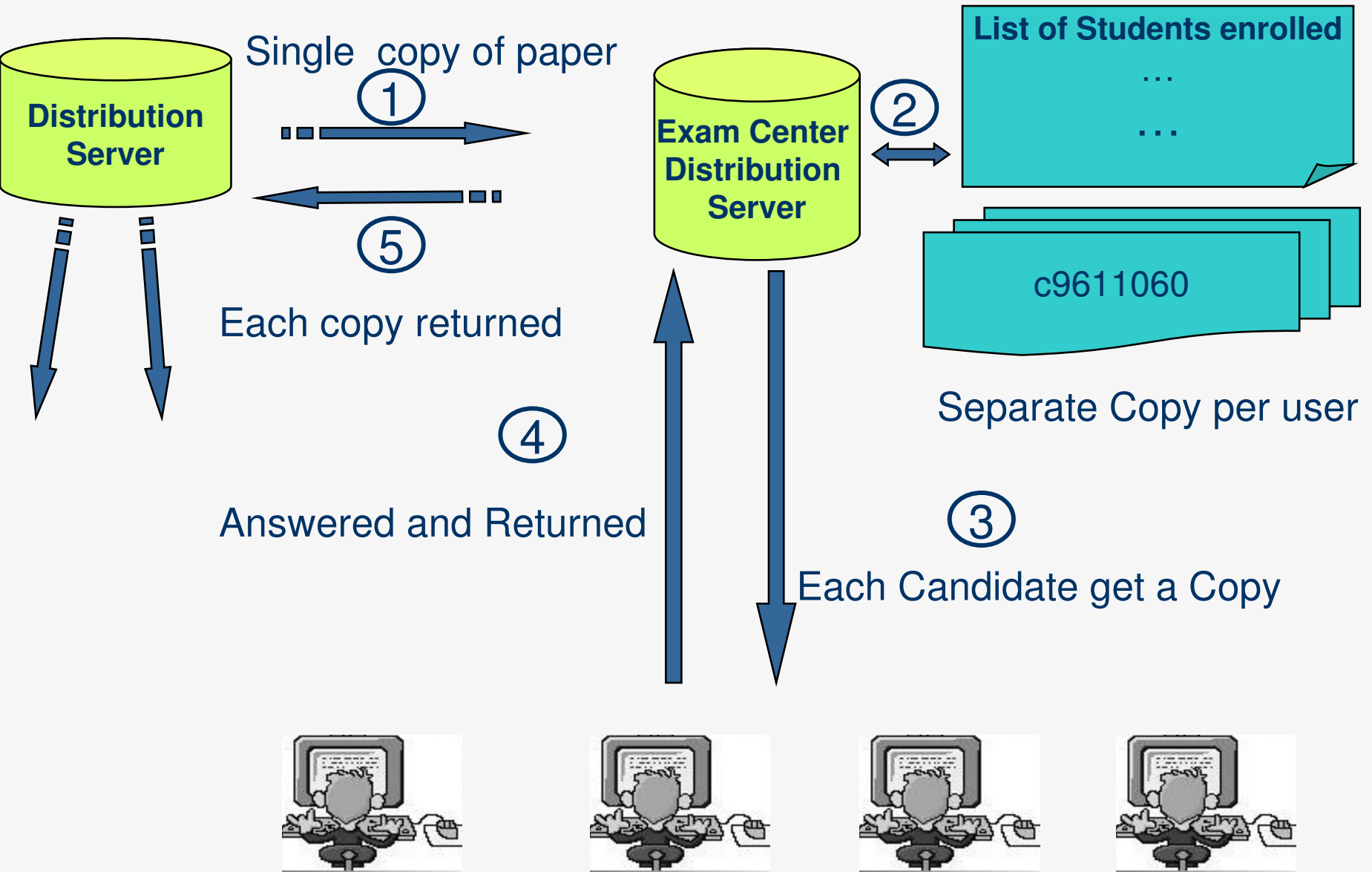
FetchAgent

RemoteSetterGUI

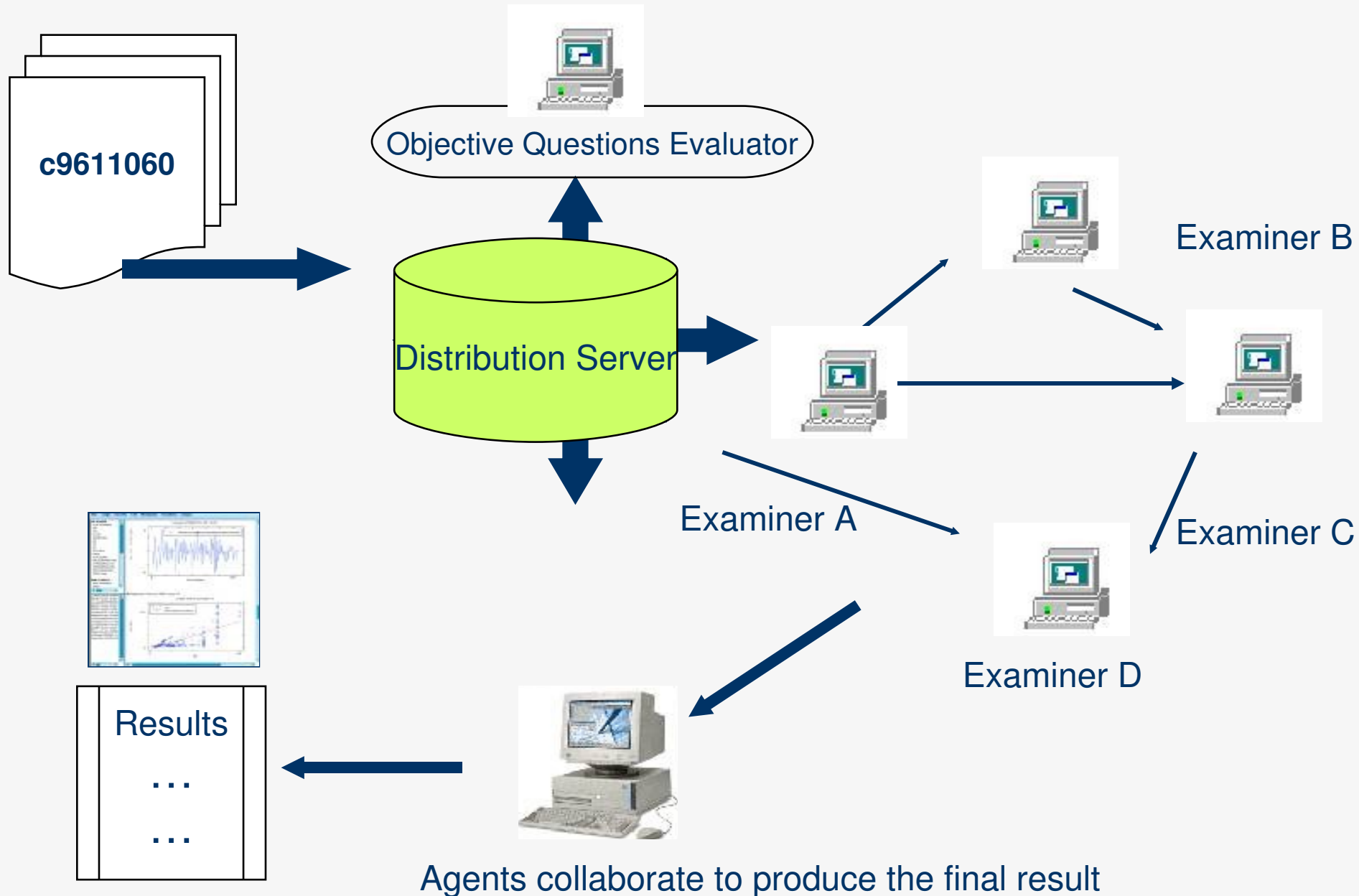
NamingService



Distribution and Testing



Evaluation and Result Compilation



Salient Features

- Generic execution environments on each machine
 - Remote code installation
- After distribution and before collection
 - The students work off-line
- Agent creation by distribution servers
 - Not student machines
- Workflow between examiners
- Automated compilation of results

Voyager: Implementation Platform

- Generalized distributed object computing platform
- Compatibility with latest java version
- Easy creation of remote objects
- Moving objects
 - relative and absolute
- Other
 - Federated directory service
 - Different kinds of messaging (sync, one-way, future)
 - Object and agent persistence support
 - Distributed event handling
 - Security manager
 - Compatible with CORBA and DCOM

Measuring Response Times

Performance Comparison

StudentID Write ID Start Q No 15

Answer this Question

DNS is which layer protocol?

REMOTE Question

DNS is which layer protocol?

<--Response Time MOBILE AGENT

Time Step	Response Time (MOBILE AGENT)
1	500.0
2	10.0
3	10.0
4	10.0
5	10.0
6	10.0
7	10.0
8	10.0
9	10.0
10	10.0

Response Time REMOTE

Time Step	Response Time (REMOTE)
1	35.0
2	10.0
3	10.0
4	10.0
5	10.0
6	10.0
7	10.0
8	10.0
9	10.0
10	10.0
11	10.0
12	10.0
13	10.0
14	10.0
15	10.0
16	10.0
17	10.0
18	10.0
19	10.0
20	10.0
21	10.0
22	10.0
23	10.0
24	10.0
25	10.0
26	10.0
27	10.0
28	10.0
29	10.0
30	10.0
31	10.0
32	10.0
33	10.0
34	10.0
35	10.0
36	10.0
37	10.0
38	10.0
39	10.0
40	10.0
41	10.0
42	10.0
43	10.0
44	10.0
45	10.0
46	10.0
47	10.0
48	10.0
49	10.0
50	10.0
51	10.0
52	10.0
53	10.0
54	10.0
55	10.0
56	10.0
57	10.0
58	10.0
59	10.0
60	10.0
61	10.0
62	10.0
63	10.0
64	10.0
65	10.0
66	10.0
67	10.0
68	10.0
69	10.0
70	10.0
71	10.0
72	10.0
73	10.0
74	10.0
75	10.0
76	10.0
77	10.0
78	10.0
79	10.0
80	10.0
81	10.0
82	10.0
83	10.0
84	10.0
85	10.0
86	10.0
87	10.0
88	10.0
89	10.0
90	10.0
91	10.0
92	10.0
93	10.0
94	10.0
95	10.0
96	10.0
97	10.0
98	10.0
99	10.0
100	10.0

(A) Application

(B) Transport

(C) Link

(D) Internet

N

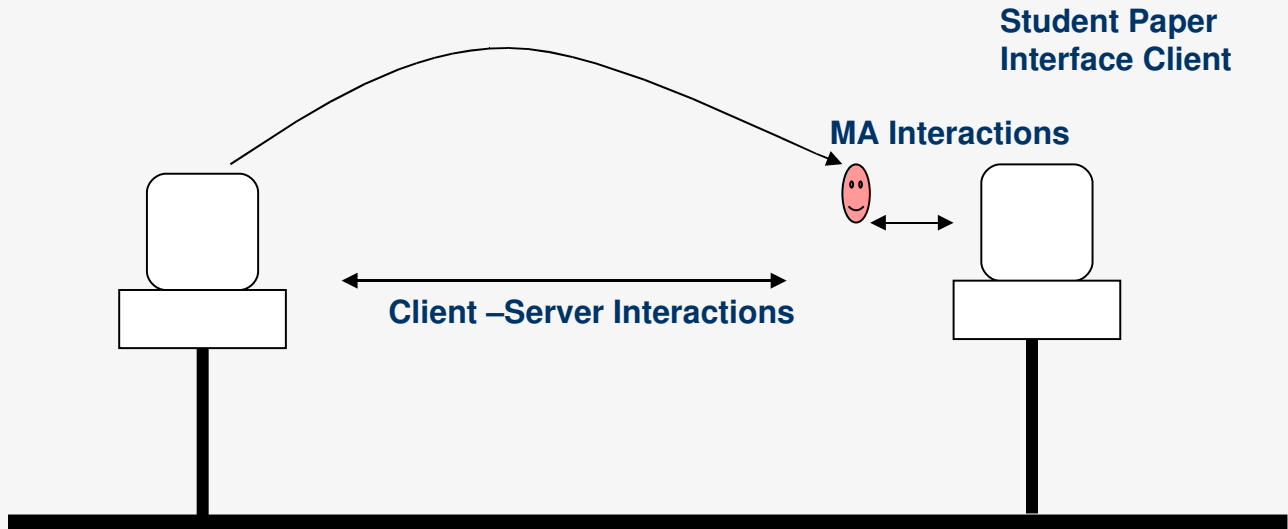
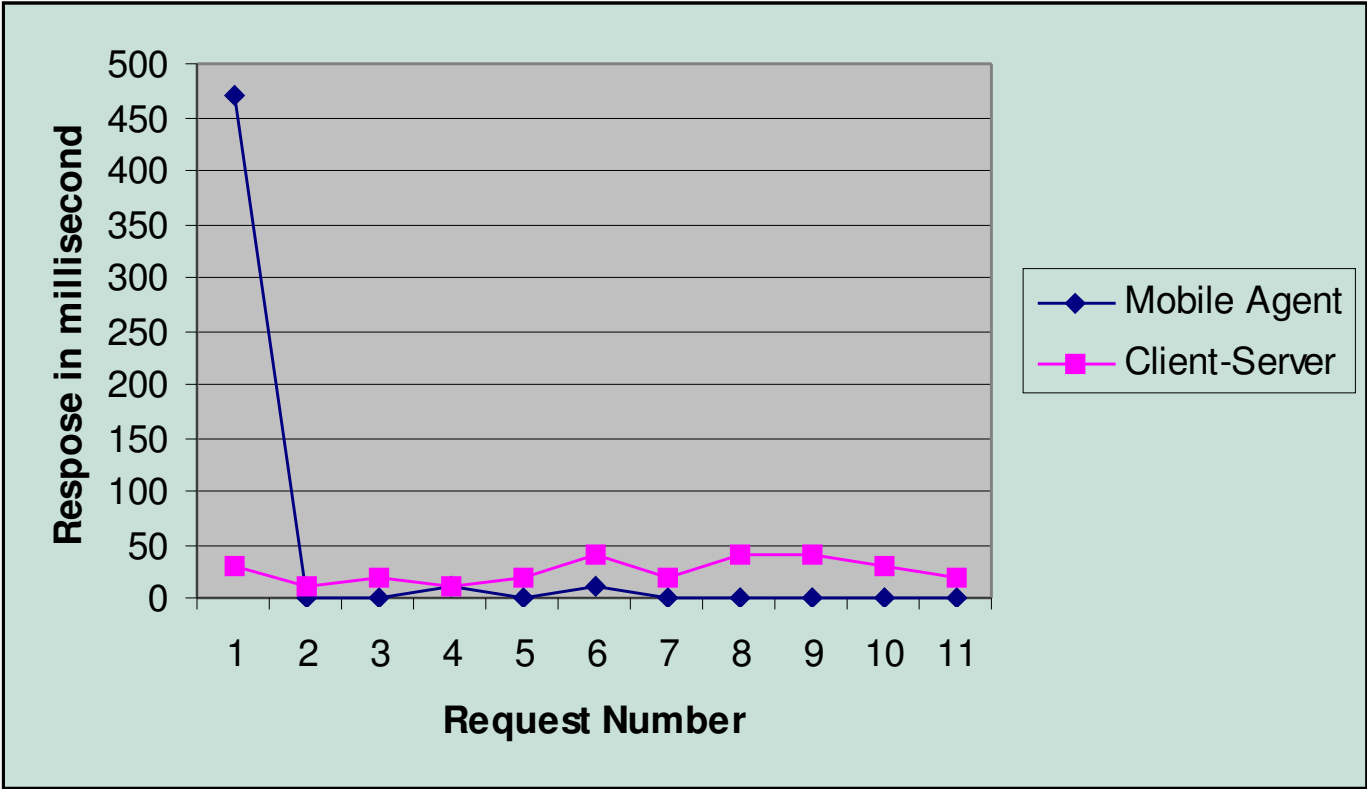
(A) Application

(B) Transport

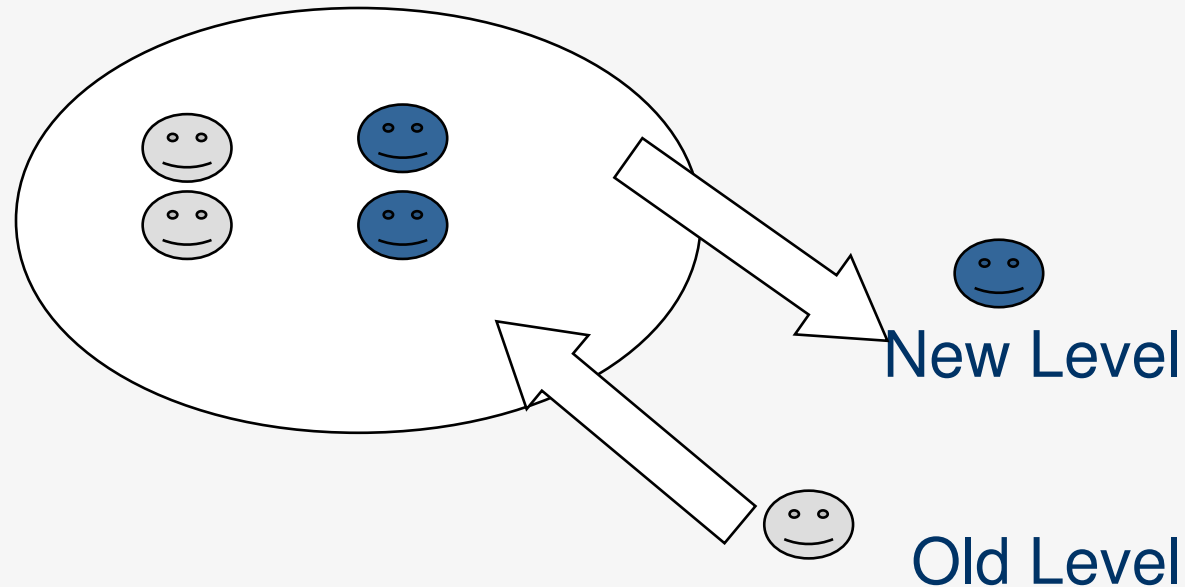
(C) Link

(D) Internet

N



Incorporating Dynamic qp



- MQPs can be organized into various skill levels
- Once a person has finished one level, a new level 2 can be sent

Characteristics of application

- Large-scale
 - Number of nodes
 - Geographical spread
 - Complexity of relationships
- Experience extendible to similar large scale applications
 - e.g. workflow B2B in global environment

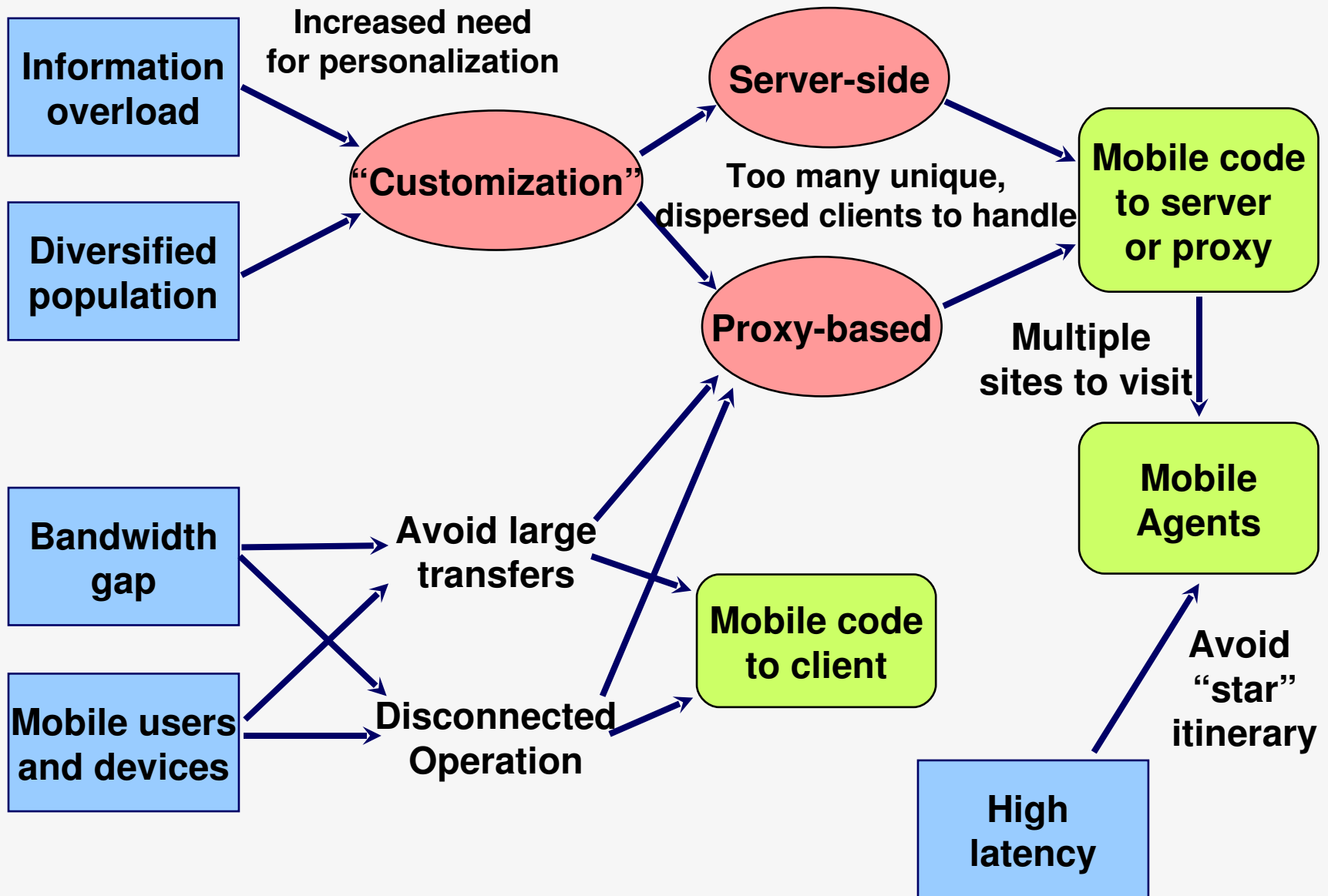
Structuring GAINS

- Scalable applications
- Flexible structuring of applications
- Dynamic extensibility
- Push-pull modes
- Adapting to varying communication channels
- Application layer multicasting
- Variety of delivered content

MADE Experience: Summary

- Mobile Agents provide effective and flexible mechanisms for structuring distributed systems like *distance evaluation*
- Advantages
 - Fast response times
 - Handling objective and subjective contents
 - Application level multicasting
 - Dynamic upgradation of applications
 - Centralized control and management of logistics
- Some Issues
 - Reliability
 - Persistence
 - Security
 - Infrastructures

Current trends lead to mobile agents [Kotz]



Conclusions

- New Paradigm
 - To design and implement
- Powerful Structuring mechanism
 - Flexible, scalable and extensible systems
- At present ready for closed systems
- Need for
 - Application centric development
 - Judicious mix of *elements*

Thank You



updated version of the tutorial

www.it.iitb.ac.in/~sri/talks/hipc01tut.ppt