Understanding Code Mobility

Présenteur: Tran The Tung

Plan

- Introduction
- Mobile Code Technologies
- Design paradigms
- Applications
- Personal remarks

The authors

Alfonso Fuggetta

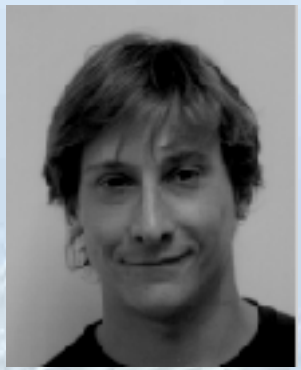
Politecnico di Milano

http://www.elet.polimi.it/~fugetta

Gian Pietro Picco

Politecnico di Torino

http://www.polito.it/~picco



Giovanni Vigna

Politecnico di Milano

http://www.elet.polimi.it/~vigna

Context

- The size of computer network are increasing
- The ways using network is changing
- The communication infrastructure is developing, but the computational infrastructure is not

Open Problems

- Scalability
- Customizability
- Physical mobility
- Extensibility

Growing size of the network

Customize the specific needs

Support to roaming users (mobile)

Ability to add new features

What is code mobility?

- Code mobility is the capability to dynamically change the bindings between code fragments and the location where they are executed

- Involves:
  Change in bindings dynamically
  Relocation of code

Theme of this paper

- Presents a framework to understand code mobility
- 3-dimensions discussed:
  Technologies
  Design paradigms
  Applications

Languages and systems that facilitate Code Mobility

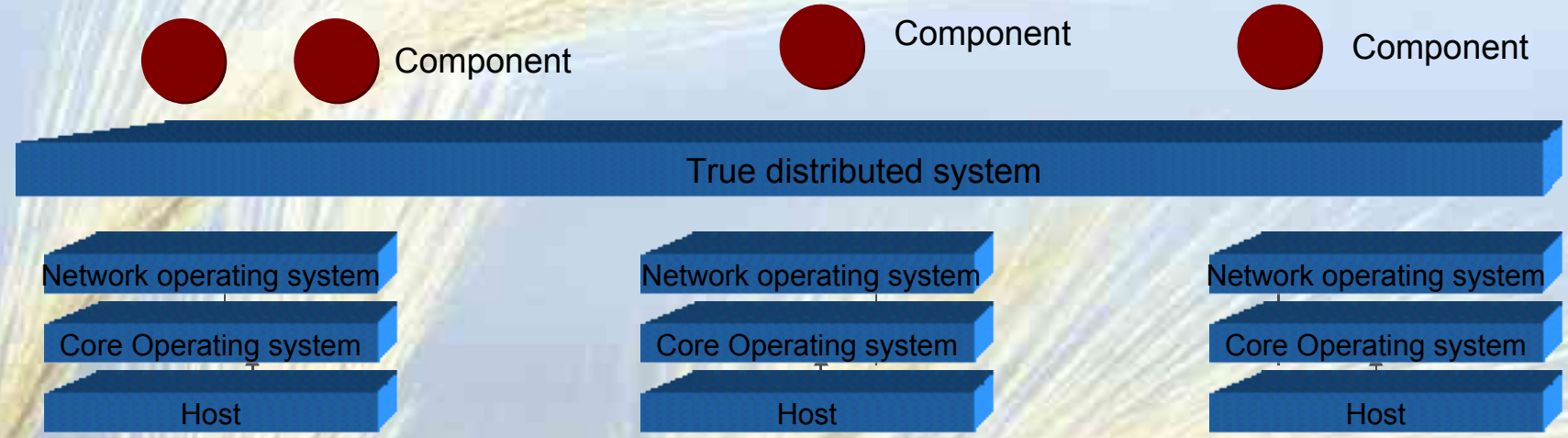
Identify specific configuration of components and their interaction

Applications that share same general role

Mobile CodeTechnologies

ditional Systems

True Distributed System (TDS) provide network transparency and distributed components are perceived as local



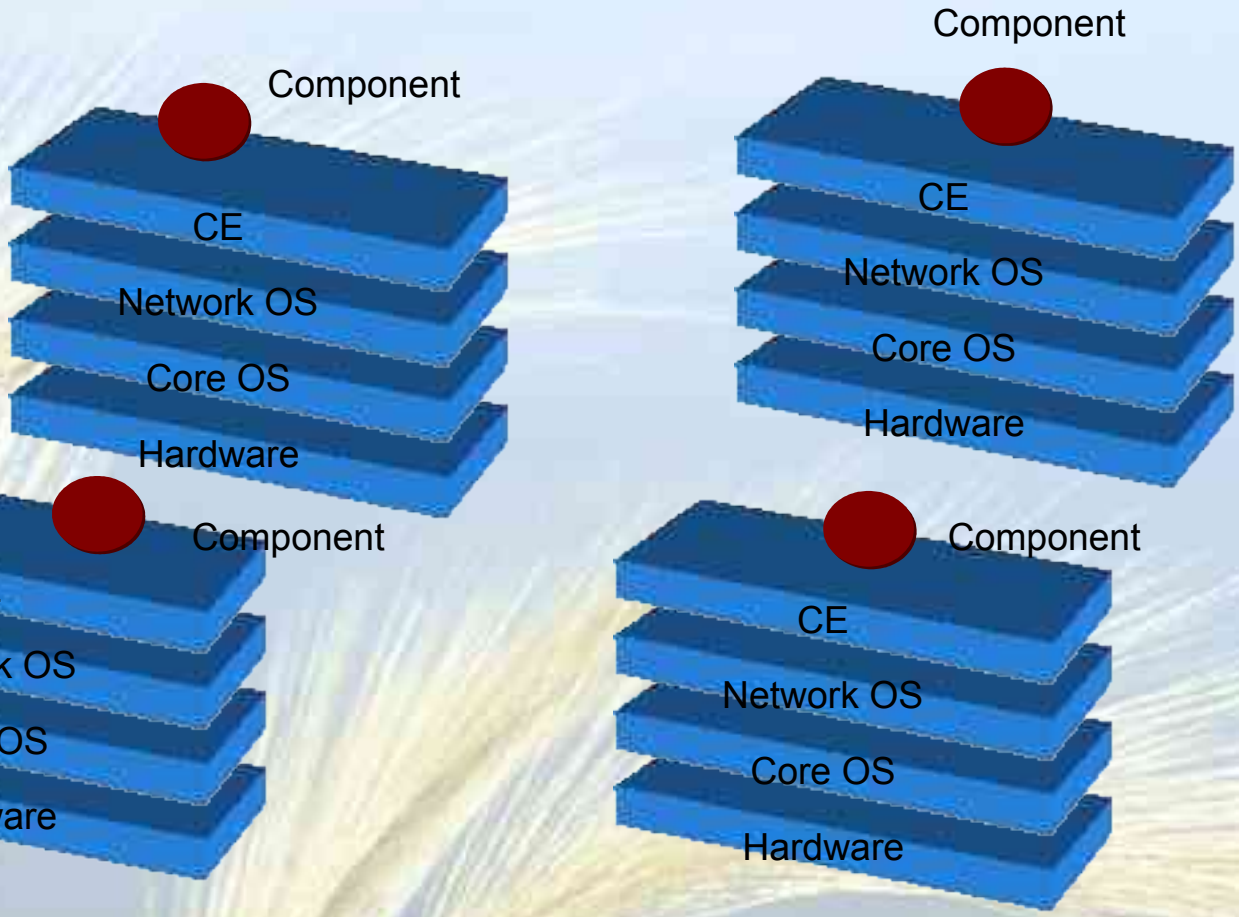
hardware

provides the basic operating system functionalities (file system, memory management)

provides the Non-transparent communication services (socket services)

Mobile Code Systems

Computational Environments (CE):
- Structure of the network is made manifest to the users…
- Retains the "identity" of the host
- Provides capability to relocate the codes dynamically

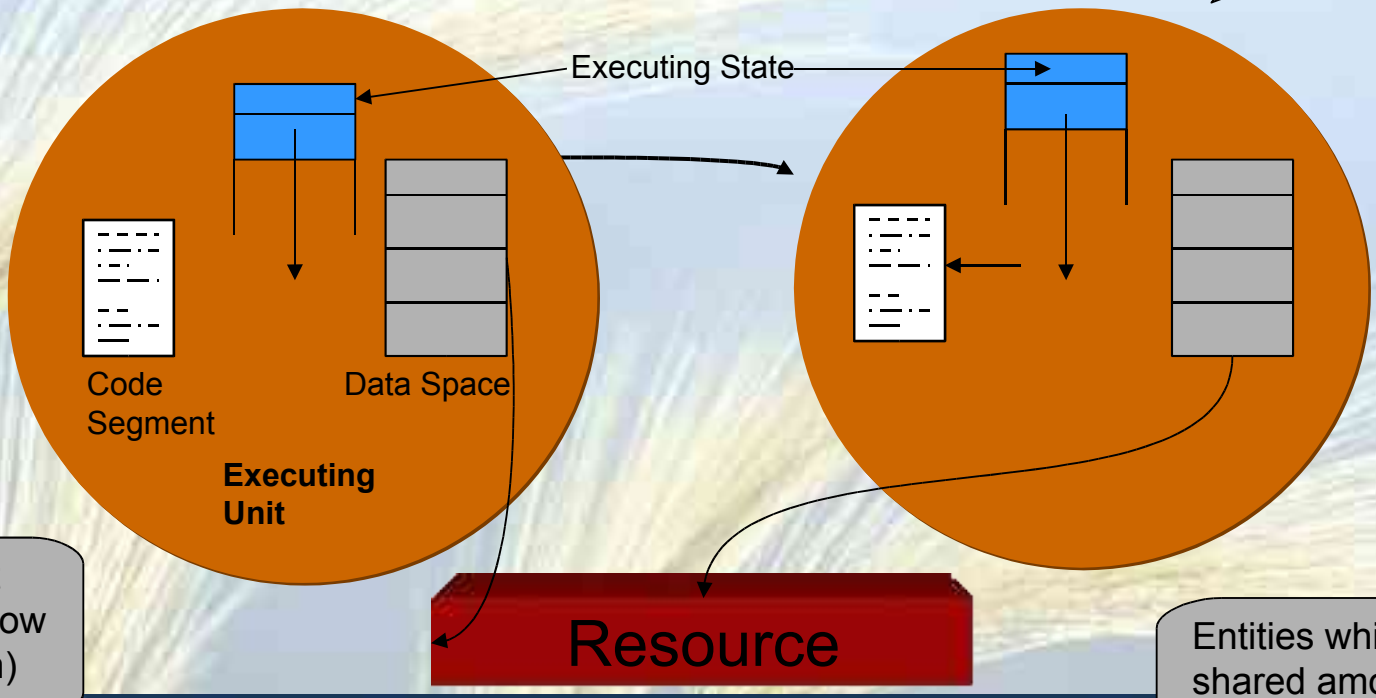


CE: Computational Environment

Components of CE

contains private data that cannot be shared

Data Space:the set of references to components that can be accessed by the EU



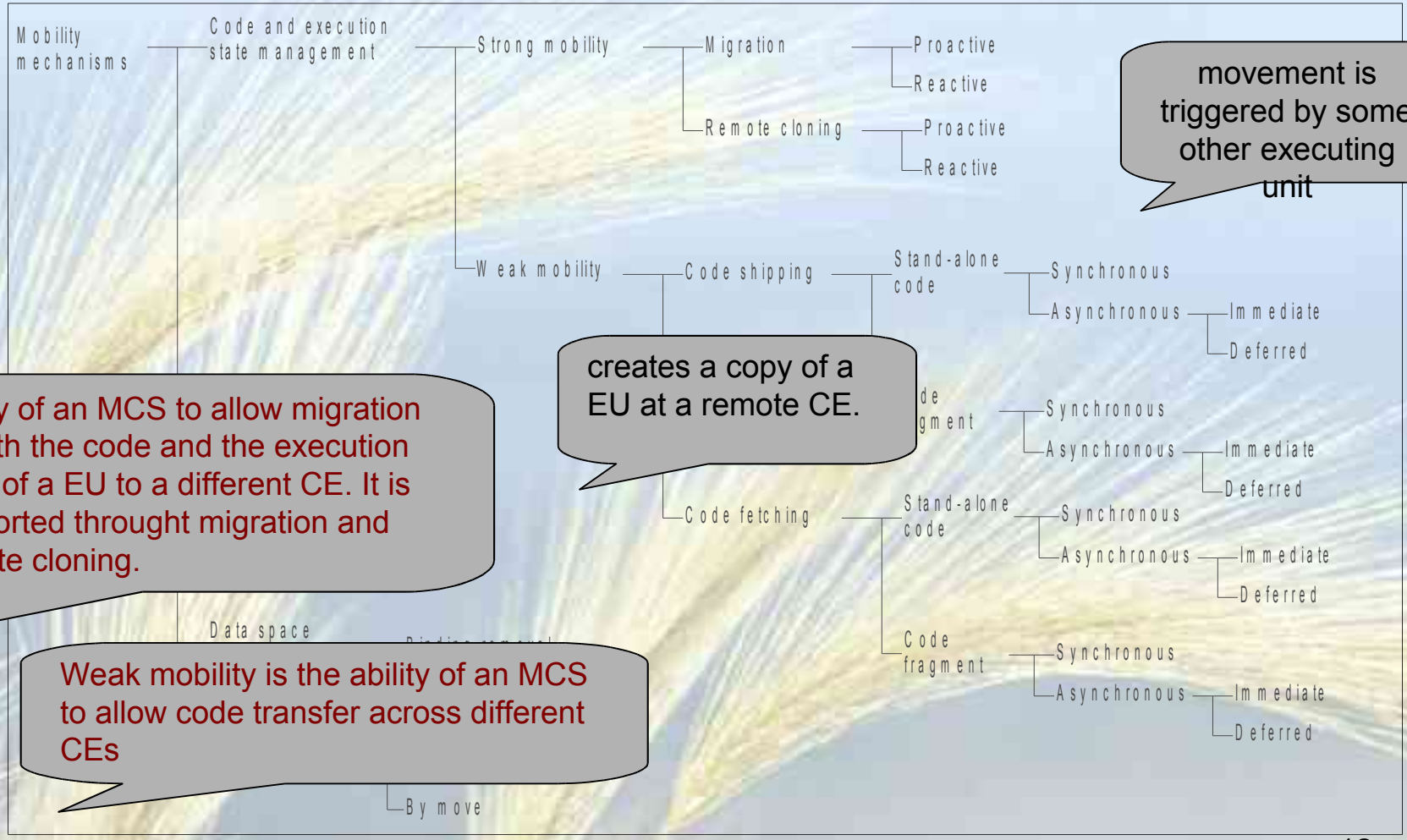
Executing Unit (Sequential flow of computation)

Entities which can be shared among multiple EUs (a file in a file system)

Classification of mobility mechanisms

suspends a EU, transmits it to the destination CE, and then resumes it

time and destination of the migration are determined autonomously by the executing unit

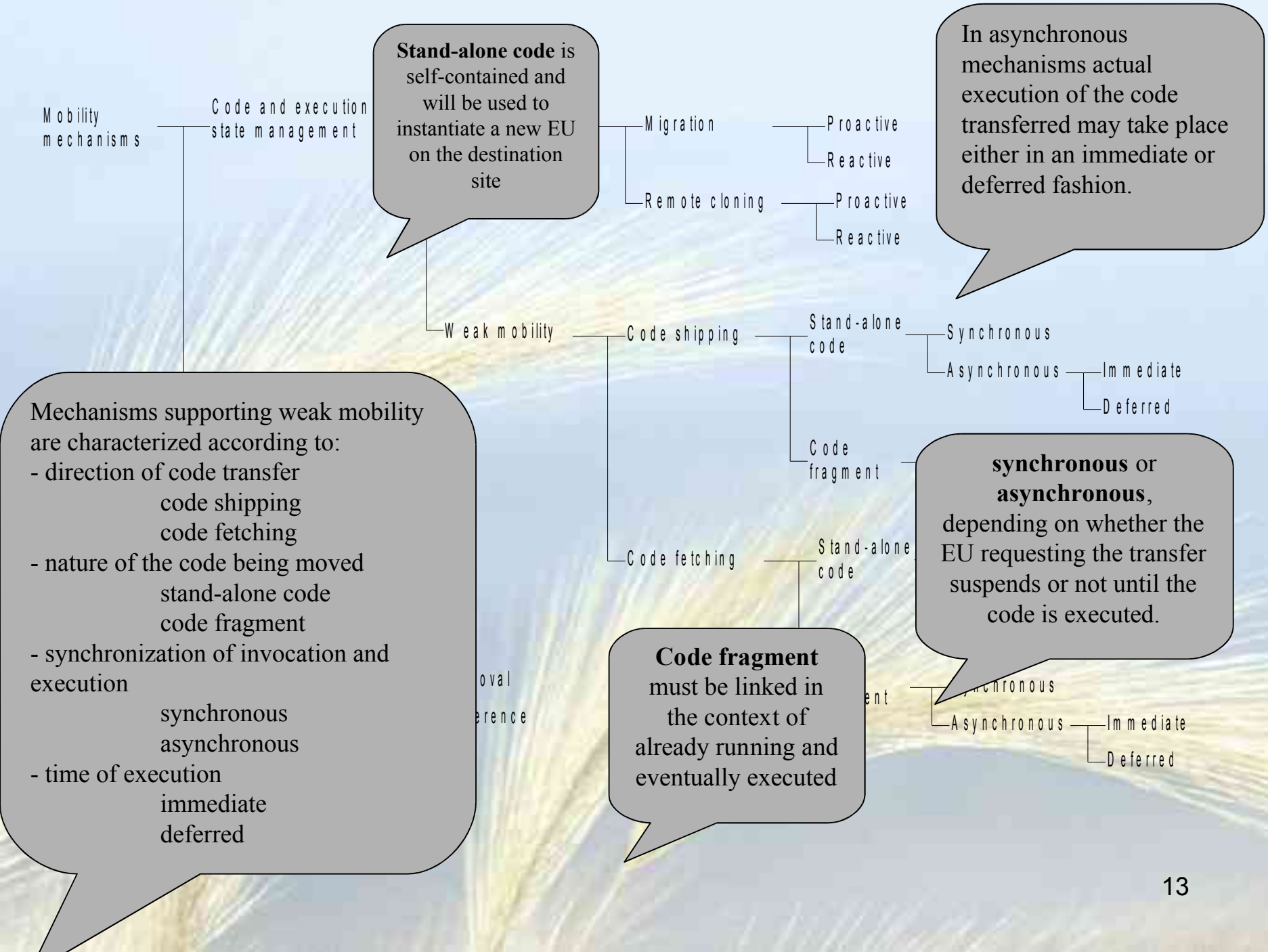


movement is triggered by some other executing unit

creates a copy of a EU at a remote CE.

Ability of an MCS to allow migration of both the code and the execution state of a EU to a different CE. It is supported throught migration and remote cloning.

Weak mobility is the ability of an MCS to allow code transfer across different CEs


Mobility mechanisms
Code and execution state management
Migration
Proactive
Reactive
Remote cloning
Proactive
Reactive
Weak mobility
Code shipping
Stand-alone code
Synchronous
Asynchronous
Immediate
Deferred
Code fragment
Code fetching
Stand-alone code
Asynchronous
Immediate
Deferred
Stand-alone code is self-contained and will be used to instantiate a new EU on the destination site
In asynchronous mechanisms actual execution of the code transferred may take place either in an immediate or deferred fashion.
Mechanisms supporting weak mobility are characterized according to:
- direction of code transfer
code shipping
code fetching
- nature of the code being moved
stand-alone code
code fragment
- synchronization of invocation and execution
synchronous
asynchronous
- time of execution
immediate
deferred
synchronous or asynchronous, depending on whether the EU requesting the transfer suspends or not until the code is executed.
Code fragment must be linked in the context of already running and eventually executed

Data Space Management

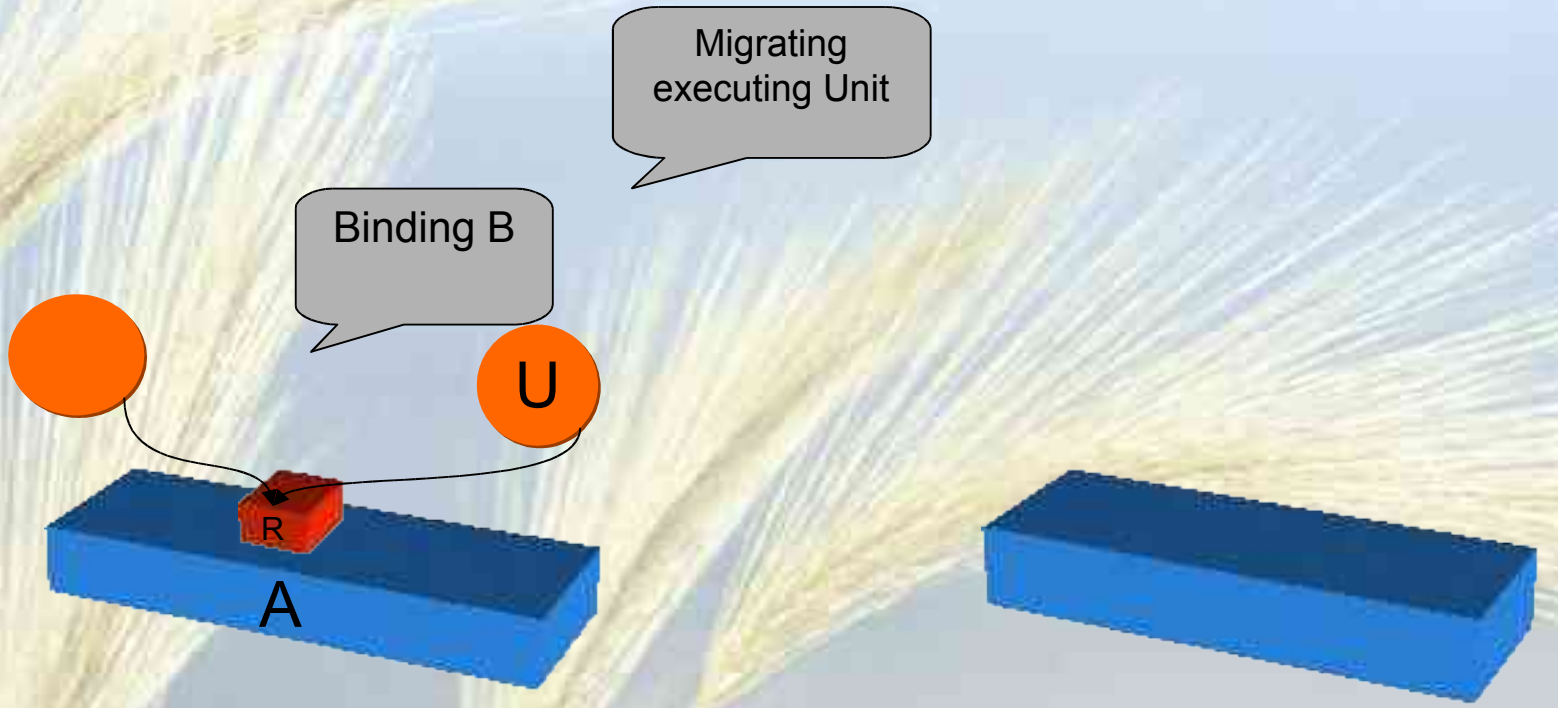
- When an executing unit migrates, its data space is modified
- Modifications may involve:
  - changing the bindings to resources
  - migrating some of the resources along with the executing unit
- The policies (in Table)

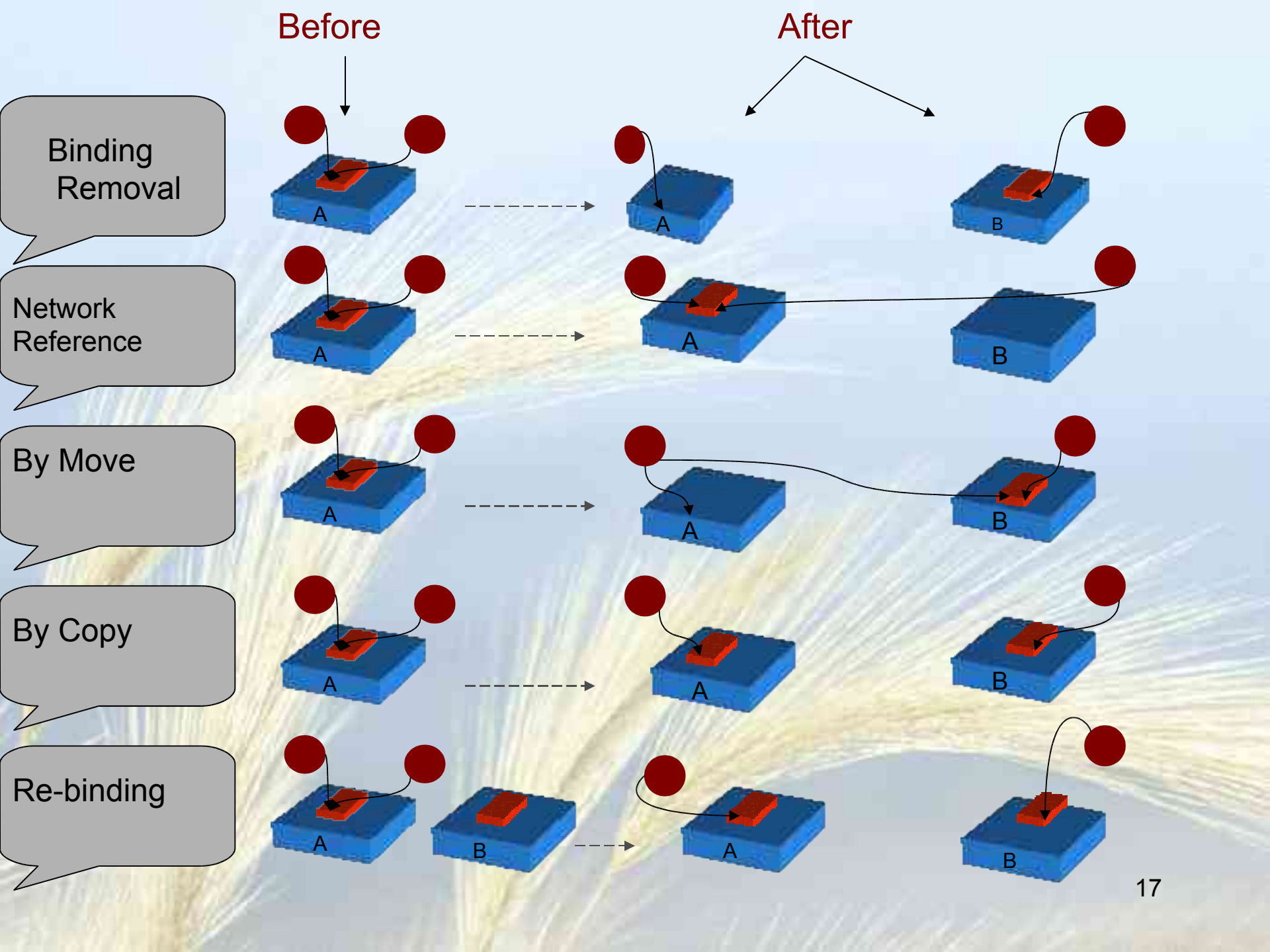
Table: Bindings, Resources, DS Mgmt

3 form Bindings

nature of the resource	Free Transferrable	Fixed Transferrable	Fixed not Transferrable
By identifier (Strongest)	By move (Network Reference)	Network Reference	Network Reference
By value	By copy (By move, Network Reference)	By copy (Network Reference)	(Network Reference)
By type (Weakest)	Re-binding (Network Reference, By copy, By move)	Re-binding (Network Reference, By copy)	Re-binding (Network Reference)

Mechanisms for Data Space Management




Before
After
Binding Removal
Network Reference
By Move
By Copy
Re-binding
A
B

Survey of MC Technologies

Technology	Abstraction terminology	Mobility Mechanisms
Ara	EU – Agents CE – Place	Strong (proactive migration)
Facile	EU – Threads CE – Nodes communication abstraction: “Channel”	Strong and weak mobility
Java	JVM – CE	Weak mobility (No DS Mgmt)
Java Aglets	CE – Context	Dispatch – code shipping Retract – code fetching
M0	EU – Messengers CE - Platforms	Shipping of stand-alone code only
Mole	EU – Mole CE - Place	Shipping of stand-alone code

Obliq	CE – Execution engines EU – Thread	Weak mobility – shipping of code
Safe-Tcl	No terminology	email
Sumatra	CE – Execution engines EU – Java Threads	Weak and Strong mobility proactive migration, remote cloning, shipping
TACOMA	EU – Agents (Unix processes)	Code shipping of stand-alone code
Telescript	CE – Engines EU – Places and Agents	Proactive migration and remote cloning

Design Paradigms

Architectural Concepts

Components

- Ressource components (devices,code,data)
- Computational components

Interactions

- Communication between components

Sites

- Support component execution and local interaction, host components

Design Paradigms

- Major design paradigms

  a. Client/Server (CS)
  b. Remote Evaluation (REV)
  c. Code on Demand (COD)
  d. Mobile Agent (MA)

This table shows the location of the components before and after the service execution.

For each paradigm, the computational component is the one that executes the code. Components in red color are those that have been moved

Paradigme	Before		After	
	resource A	resource B	resource A	resourceB
Client-Server	A	Know-how resource B	A	Know - how resource B
Remote Evaluation	Know- how A	Resource B	A	Know- how resource B
Code on Demand	Resource A	Know- how B	Resource know-how A	B
Mobile-Agent	Know- how A	Resource	_	Know - how resource A

Mobile Code Applications

- Distributed Information Retrieval
- Active Documents
- Advanced Telecommunication Services (video conferencing, video on demand)
- Remote Device Control and Configuration
- Workflow management and Cooperation
- Active networks
- E-Commerce

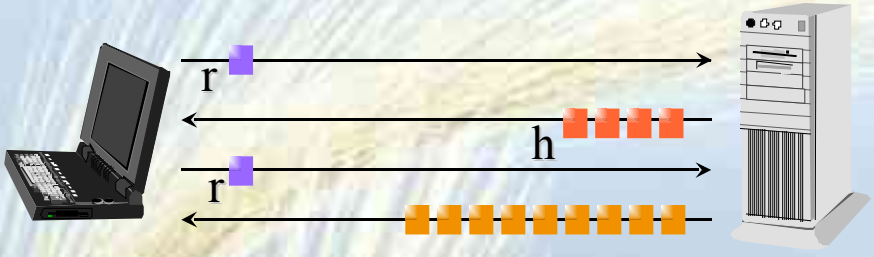
A Case Study In Network Management

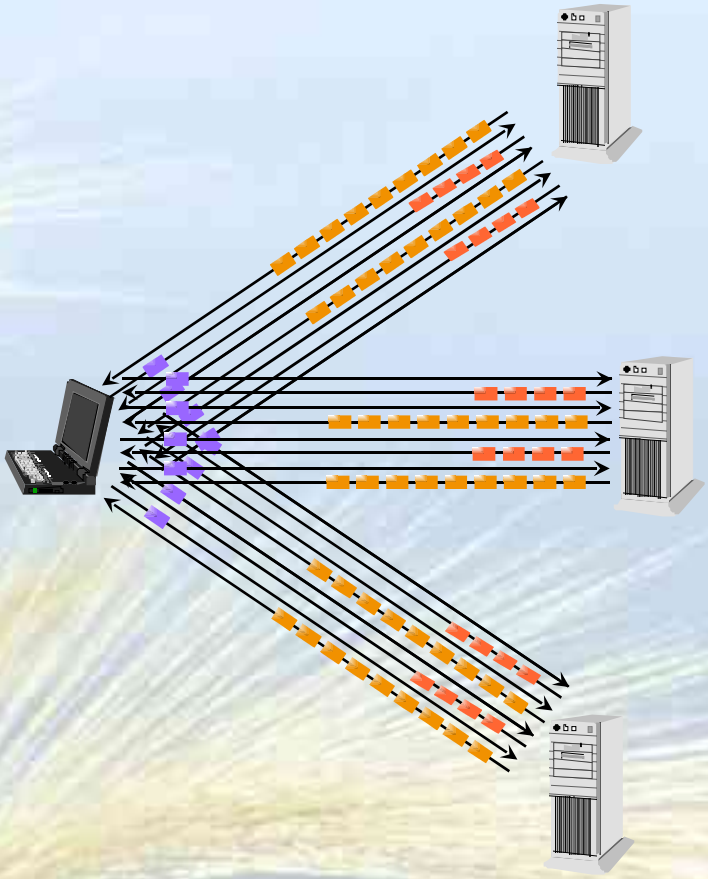
- Determine if mobile code is suitable for the specific application first.
- Identify the corresponding paradigms.
- Analyze the tradeoff for each application functionality.
- Select the technology to implement the application according to the tradeoff.

PARAMETERS MODELING A SIMPLE NETWORK MANAGEMENT DATA RETRIEVAL FUNCTIONALITY

Parameter	Unit	Description
N	Node	Number of managed network devices
Q	Instruction	Number of SNMP instructions needed to perform a single device query
i	Bit	Size of an SNMP instruction
h	Bit	Message header and other auxiliary data encapsulating message content
r	Bit	Average size of the result of an SNMP instruction

Client-Server

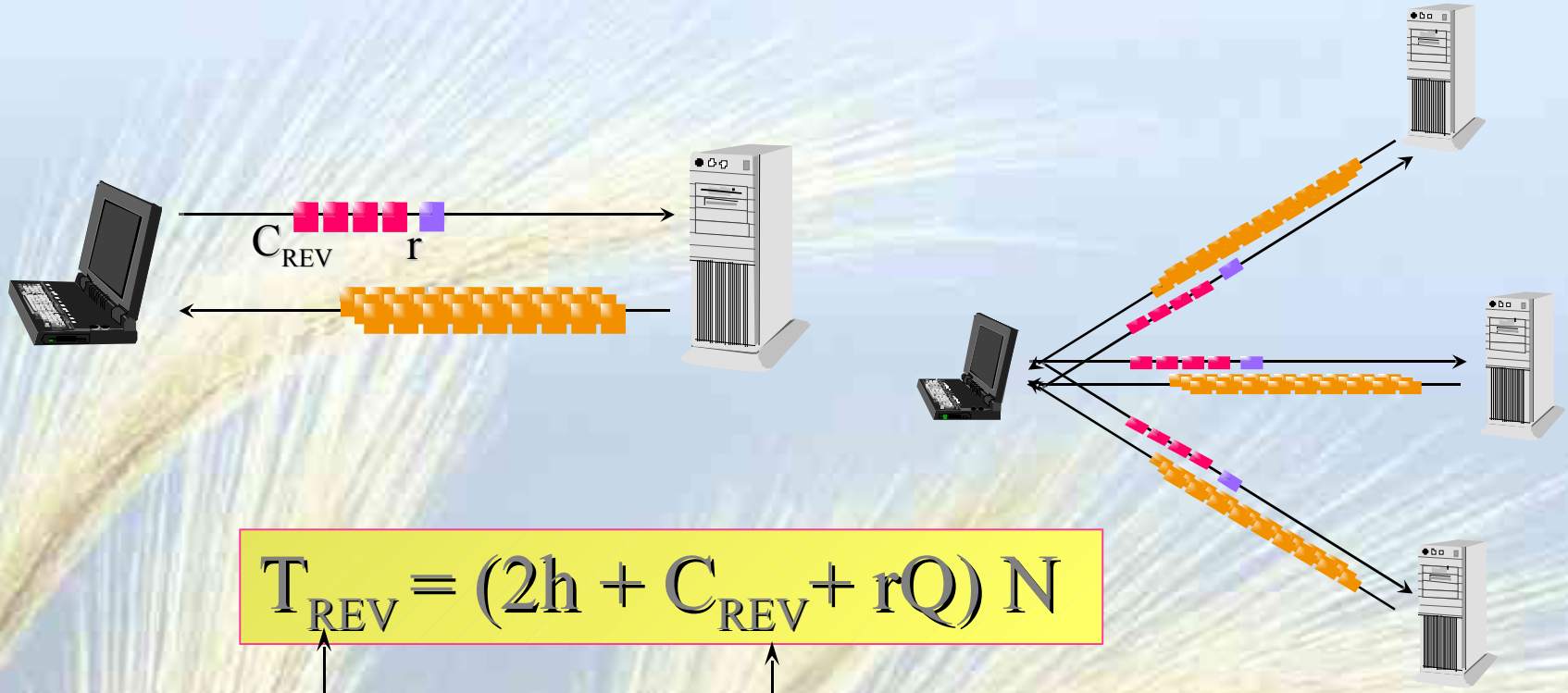




$$T_{CS} = (2h + i + r)Q\,N$$

network traffic

Remote Evaluation

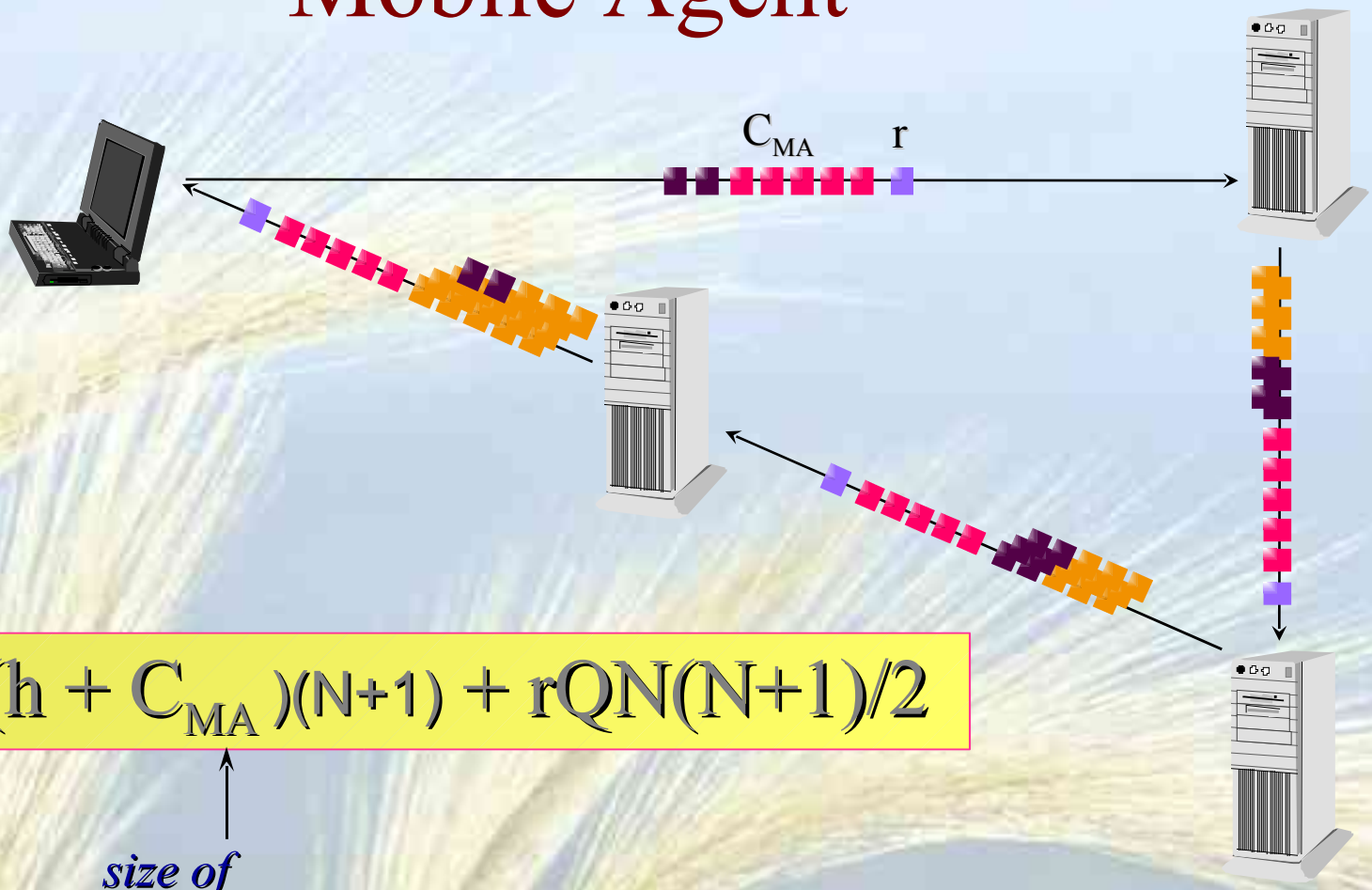


$$T_{REV} = (2h + C_{REV} + rQ)\ N$$

network traffic

Size of code sent

Mobile Agent



$$T_{MA} = (h + C_{MA})(N+1) + rQN(N+1)/2$$

network traffic

size of component mobile

Conclusions

- Solution of code mobility is promising for design and implementation of large-scale distributed systems
- However, this field is still immature
- Need to improve our understanding of the properties and weaknesses of existing design paradigms

Personal remarks

- ► Positive
- Provides a conceptual framework for understanding code mobility
- Provides a case study to guide software engineers through the design and implementation phases of application development
- ► Negative
- No explained examples
- Too many new terminologies introduced in one paper

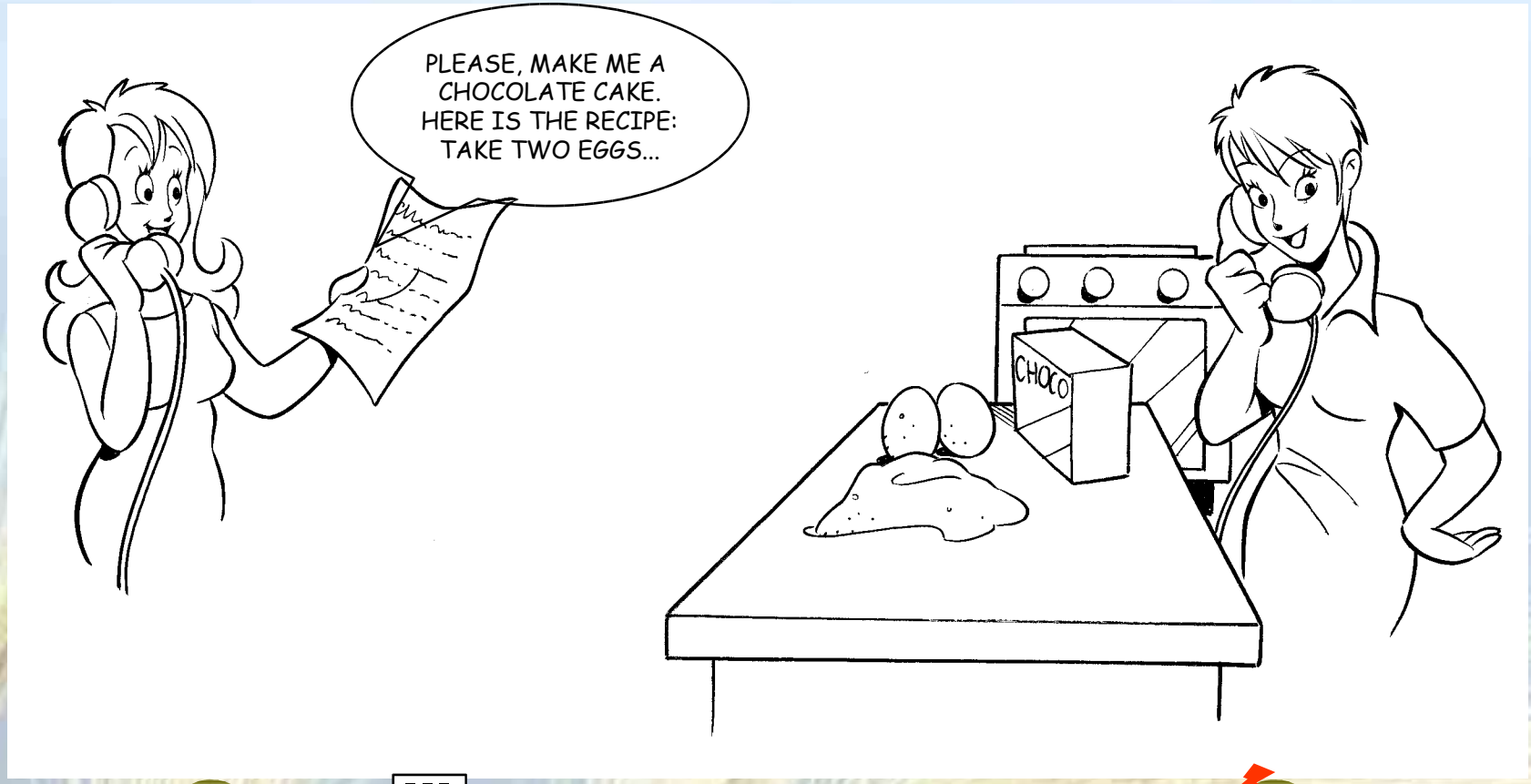
Merci de votre attention

Some illustrations « fun » which I have found in the internet

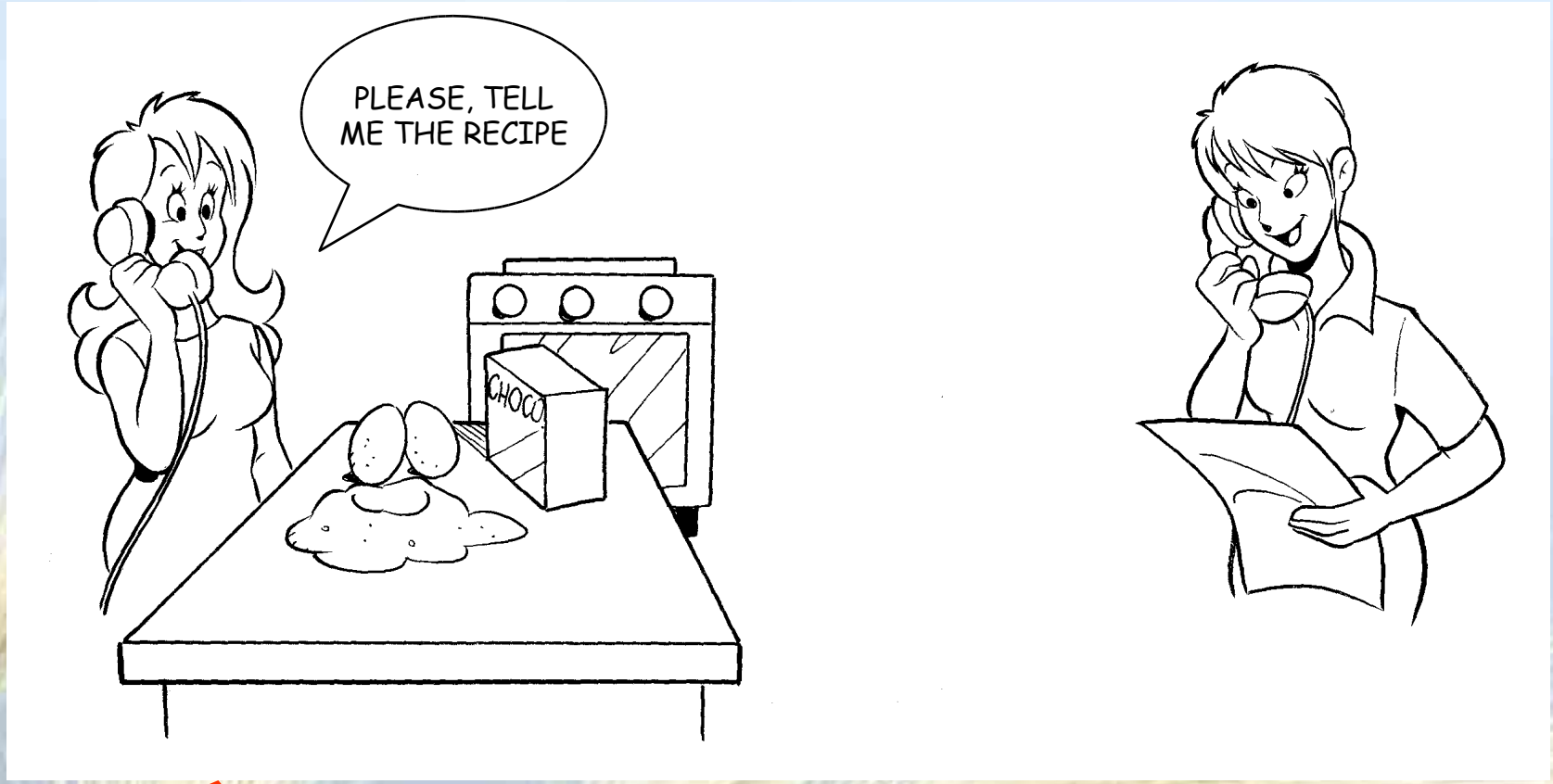
A chocolate cake

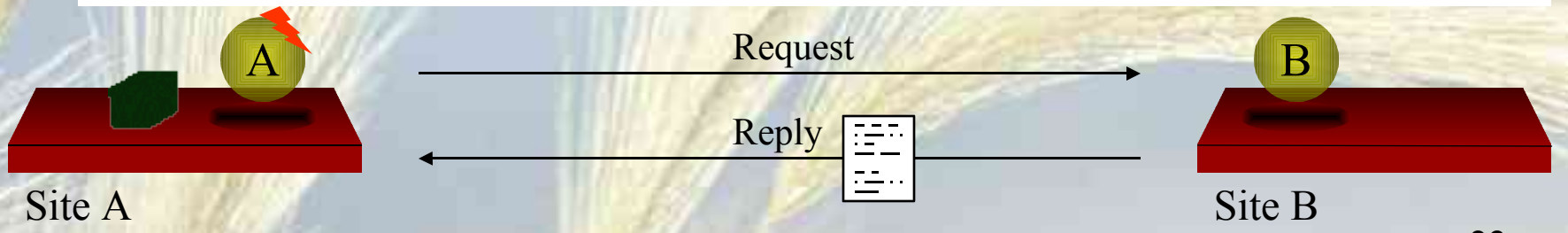
Client-Server

Remote Evaluation



Code On Demand





Mobile Agent