Designing Distributed Applications with Mobile Code Paradigms

Qinhai Xia

3/5/98

Introduction

- History:
  Distributed Systems have been investigated for years
- Motivation:
  WWW and network in general
- Problem:
  Scalability

Introduction (Continued)

- Possible Solution:

  Mobile Code Languages (MCLs) -- emphasis on the application of code mobility to a large scale setting

- This paper's solution:

  Code mobility in design phase -- repertoire of design paradigms

Advantage:

- Abstract away from MCLs (independent of the specific technology)
- Conceptualize the design paradigms to address code mobility

Overview of MCLs

- Strong mobility:

  Allow Execution Units (EUs) to move their ***code*** and ***state*** (Telescript, Tycoon, Agent Tcl, Emerald)

- Weak mobility:
  - EU to be bound dynamically to code from other site
    - EU link code downloaded from network
    - EU receive code from another EU
  - JAVA, MOLE, TACOMA, FACILE, MO

Traditional Distributed System and Code Mobility

- Design phase:
  not considering component location

- Implementation phase:
  - Programmer's responsibility
  - Middleware Layer (CORBA intentionally hides the location from the programmer)

Traditional Distributed System and Code Mobility (Continued)

- Advantage:
  - Simple in design phase
  - If a nice middleware like CORBA exists, also simple in the implementation phase
- Disadvantage:
  - Ignoring different cost (latency, access to memory)
  - Leading to unexpected performance and reliability problems

Mobile Paradigms Definitions

- Components:
  - Resource components (data, file, device driver etc)
  - Computational components (process, thread)
- Interactions:

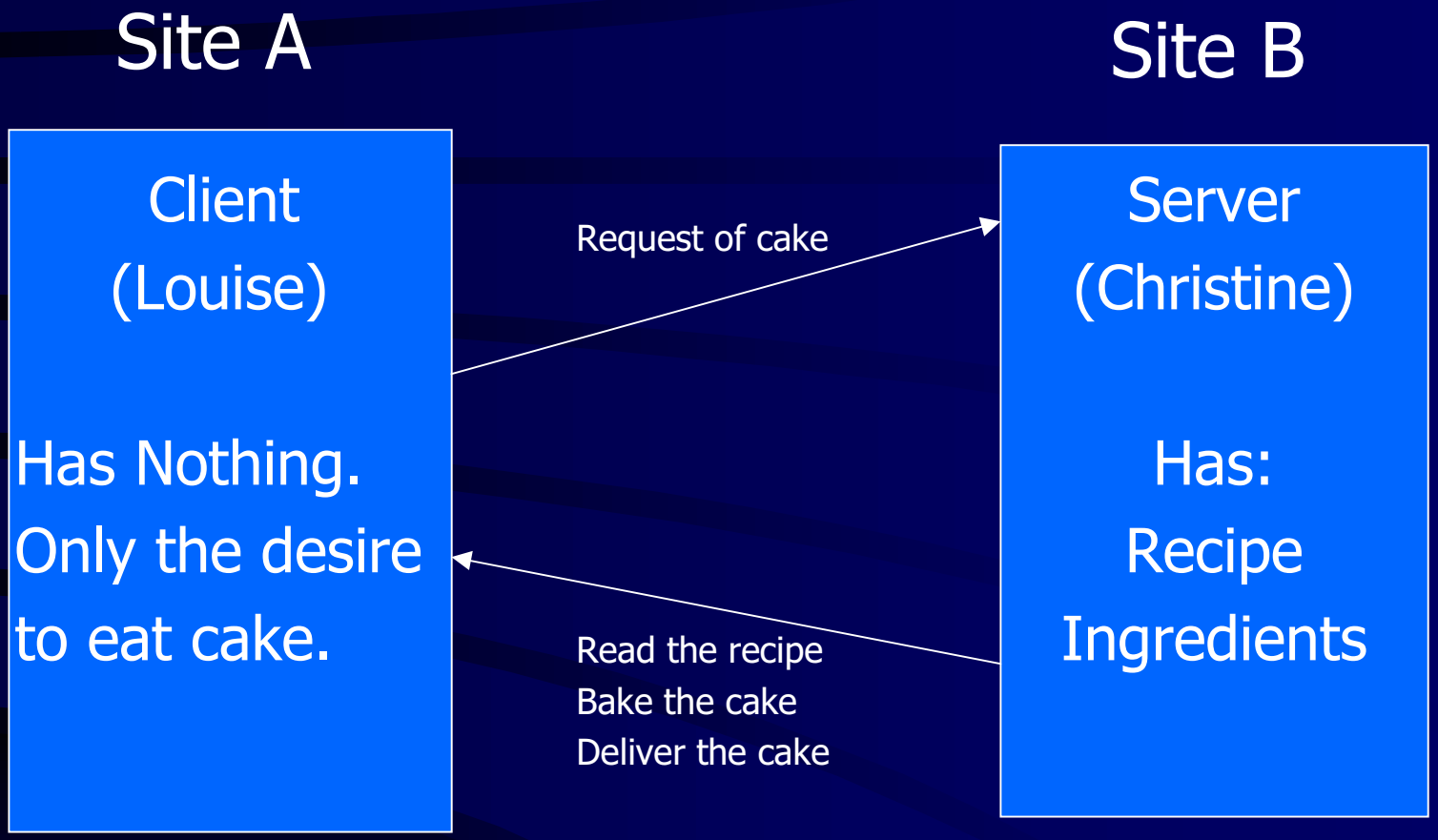
  Events between two or more components (messages)
- Sites:

  Execution environment -- Provide support for execution of the computational components

Four Models and an Example
Louise and Christine make a cake

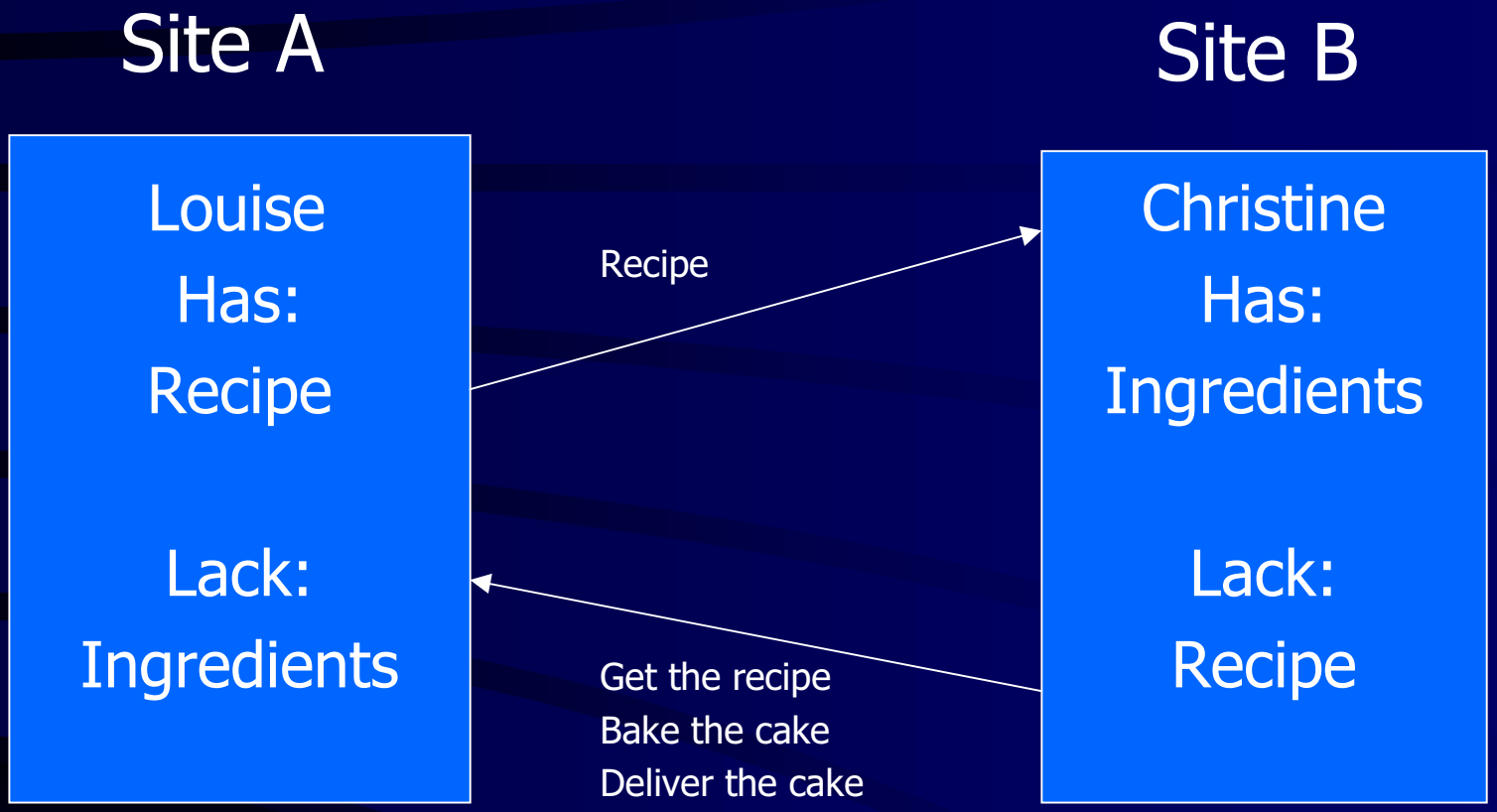
- Cake -- result of the service
- Recipe -- know-how / code
- Ingredients -- resource component / data
- Louise -- computational component A
- Christine -- computational component B
- Louise's home -- Site A
- Christine's home -- Site B

Traditional Client and Server Model: (CS)



X Windows System

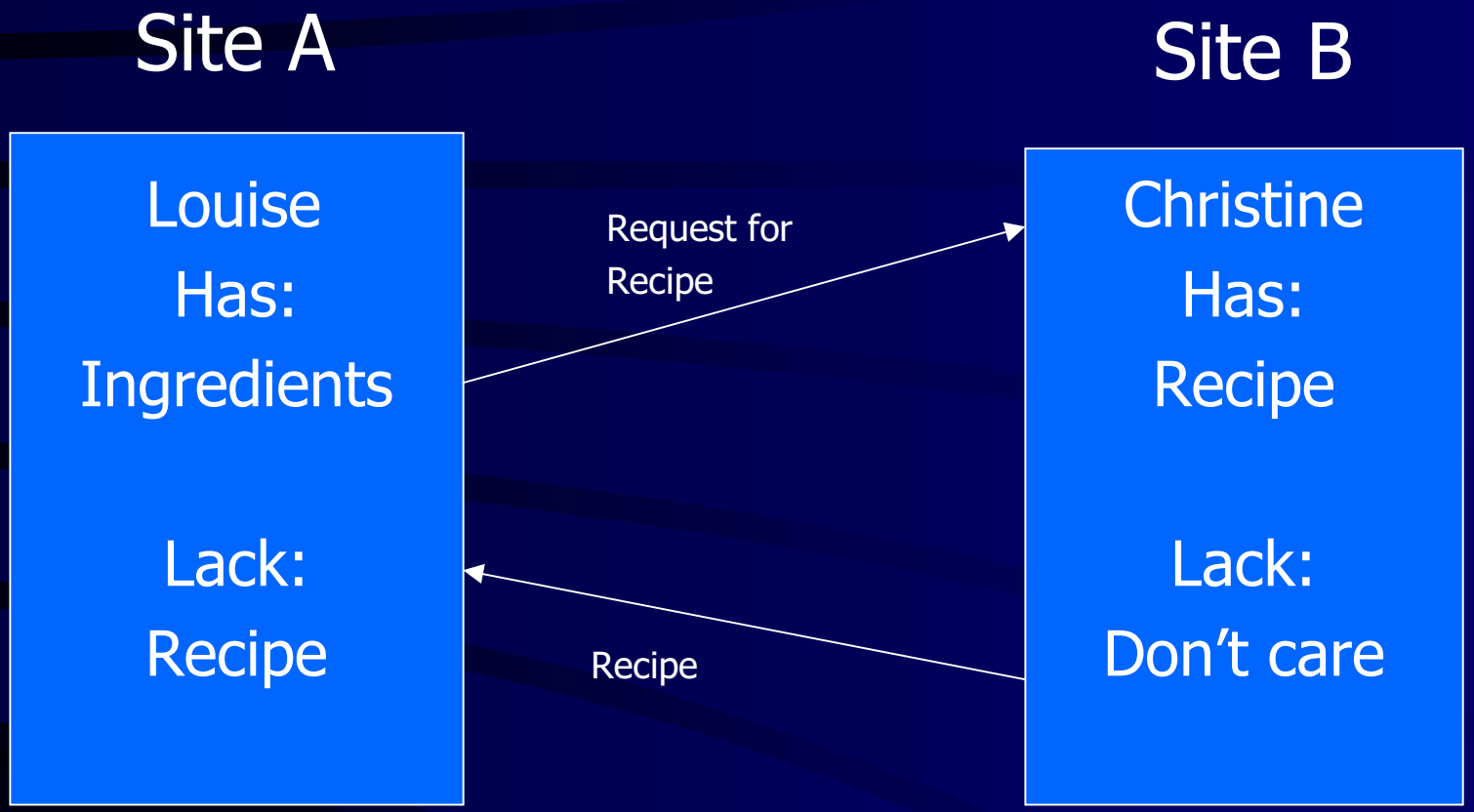
Remote Evaluation Model: (REV)



Unix: rsh command

PostScript printer

Code on Demand Model: (COD)

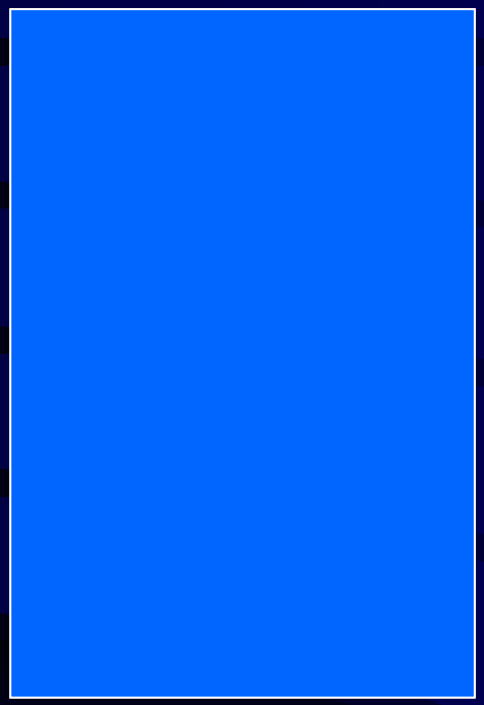


Principle gets a new type of document. Document header may contain a reference (URL address) to the code that is needed to interpret the document. Then the principle will go to the reference and download the necessary code and execute it afterwards.

Mobile Agent Model: (MA)

Site A

Site B



Network nodes test and correction

Mobile Paradigm Recap

	Before		After	
Paradigm	Site A	Site B	Site A	Site B
Client – Server	A	know-how resources B	A	know-how resources B
Remote Evaluation	know-how A	resources B	A	know-how resources B
Code on Demand	resources A	know-how B	resources know-how A	B
Mobile Agent	know-how A	resources	---	know-how resources A

A and B is already in execution

Deployment of Upgrade of Distributed Applications

- When installing a new application to a set of network nodes, the operation could be carried out in a central server by using REV MA to analyze each node's configuration and install accordingly.
- The latest version would be kept on the code server. When a new functionality needs to be added, COD could be used when the new functionality is activated and the new version could be downloaded.

Customization of Services

- Traditional: a fixed of service through a *statically* defined interface
- REV, MA: could perform services tailored specifically to one client
- Disadvantage: Each client is responsible for providing correct service it needs. In the previous case, it is much simpler. Of course a mixture could be the solution.

Customization of Services Continued

- Database has already use this kind of approach.
- Differences:
  - MA migration of a computation *already in execution*
  - SQL similar to REV only migrates code
  - Paper's approach the language could be computational complete
  - SQL certainly not the case

Support for Disconnected Operations

- Problem:

  Low-bandwidth and low-reliable communication channels. Avoid the generation of traffic over the weak links.

- Solution:

  REV and MA pass the code once through the weaker link and get the result one more time through the weak link.

Improved Fault Tolerance

- Problem:

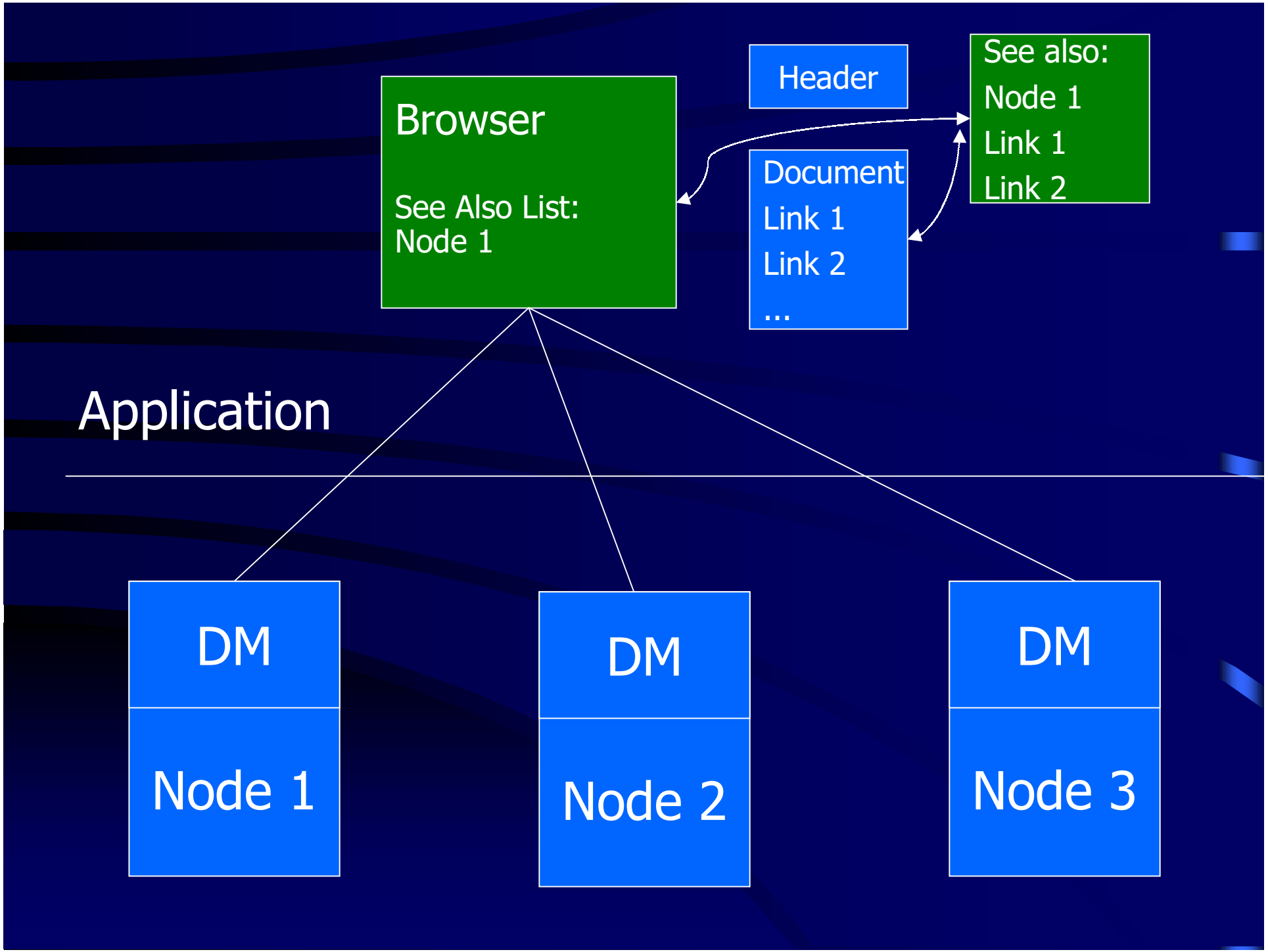
  On client's side, its local code interleaves with statements that invoke services on the server. In case of failures, it is very difficult to recover to a consistent state.

- Solution:

  REV/MA encapsulate all the state component that can be traced, checkpointed, and eventually recovered locally.

Choosing the Right Paradigm

- No paradigm is absolutely better than others.
- The paradigm proposed here do not necessarily prove to be better than traditional ones.
- The choice of paradigm must be performed by case-by-case basis. (Network traffic, cpu and other resources)


Header
See also:
Node 1
Link 1
Link 2
Browser
See Also List:
Node 1
Document
Link 1
Link 2
...
Application
DM
Node 1
DM
Node 2
DM
Node 3

Case Study: Information Retrieval Application

Assumptions:

- Communication Cost -- only proportional to the bytes that are transmitted. Zero if two components are on the same node.
- CPU time -- zero
- Each node can access every other node without overhead of access control and authentication.

Assumptions Continued

- all request have a fixed length (*r*)
- each node holds same number D of documents
- the relevant information is uniformly distributed among a set of N nodes, being *i* the ratio between relevant and total documents.
- documents have constant length. *h* and *b* are the size of the header and the body, respectively.

Client and Server

$Tcs = (D + iD)\ r\ N + (Dh + iDb)\ N$

D requests for header, *i*D request for body

Remote Evaluation

Remote Evaluation could perform the filtering task on the node!

Crev -- size of the code to execute on remote node

Trev = (r + Crev + iDb) N

Mobile Agent

- The browser migrates on each relevant node
- Perform filtering locally
- Save the state of all relevant information and the see-also list

Mobile Agent Continued

T*j* = *r* + Cma + S*j*
Cma -- size of the agent
S*j* = Dsalist + s + *iDbj*

Sa = Dsalist + s
T*j* = *r* + Cma + Sa + *iDbj*

Tma = $\sum_{j=0}$(r + Cma + Sa + *iDbj)*
Tma = (*r* + Cma + Sa + N*i*D*b*/2)(N+1)

Comparison

Necessary info: $I = iDbN$

$Ocs = Tcs - I = (r + ir + h)DN$

$Orev = Trev - I = (r + Crev)N$

$Oma = Tma - I$

$= (r + Cma + Sa)(N+1) + I/2 (N - 1)$

REV vs. CS

Assuming $r << Crev$

$(r + ir + h) D > Crev$

Conclusion and Future Work

- The paradigms are independent of the technology to implement them.
- Only hint for evaluation of the requirement against the features of each paradigm.
- Investigating minimal abstractions needed to express all interaction patterns involving code mobility at design level and their mapping at the language level.