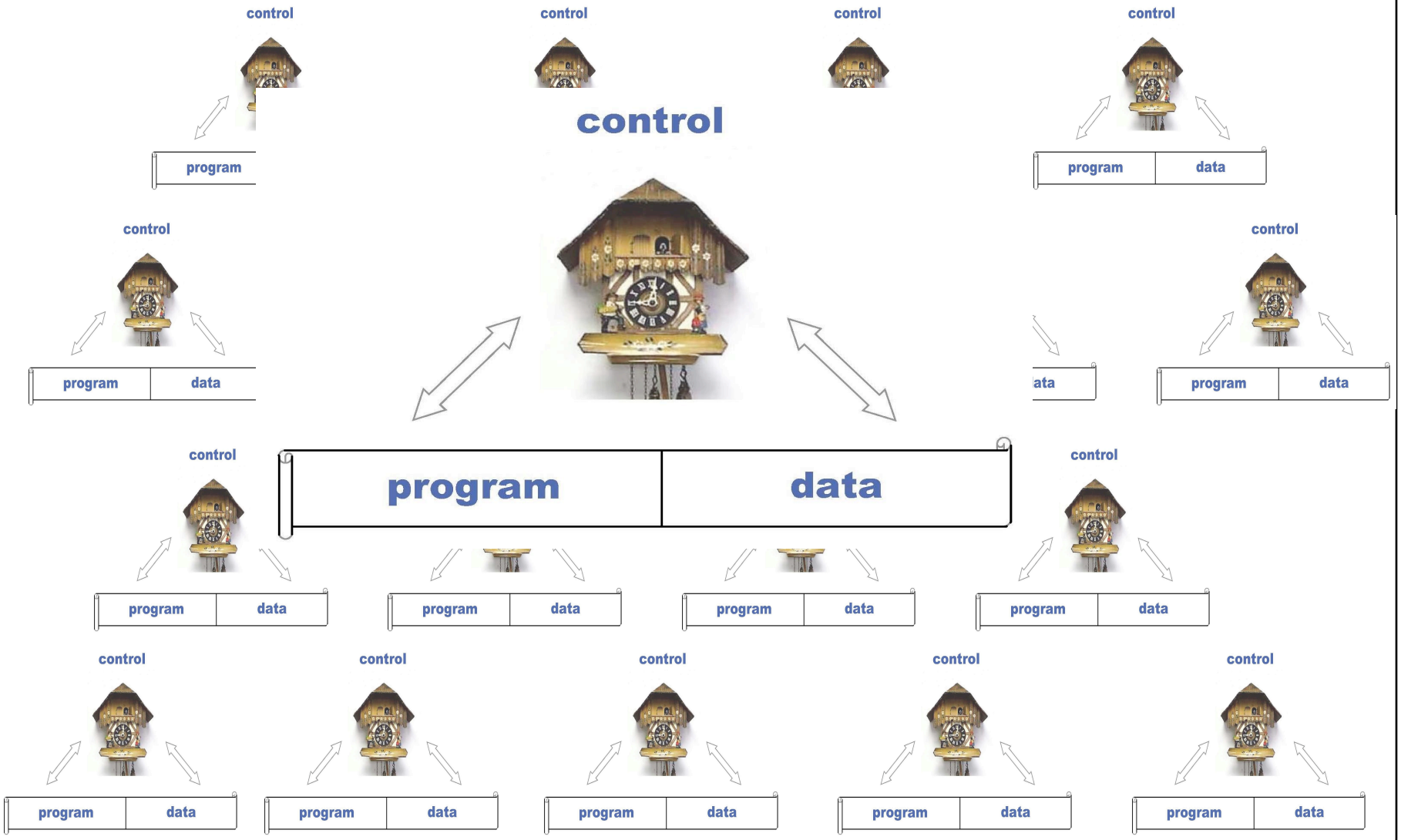
Αρχιτεκτονικά Μοντέλα Κατανεμημένων Συστημάτων

Υπολογιστικό Μοντέλο ΠΠΠ



Architectural Models

Descriptive Models

- Models are intended to provide an abstract, simplified and consistent description of a relevant aspect of Distr. System design.
- Architectural models of D/S define the way in which components of systems:
  - Interact with one another
  - Are mapped onto an underlying network of computers

Architecture of D/S

- Architecture of a system: its structure in terms of separately specified components.
- Goal: to ensure that the structure will meet present and likely future demands on it.
- Concerns:
  - Reliability
  - Manageability
  - Adaptability
  - Cost-effectiveness

Why Discuss Architecture?

- Descriptive
  - Provide a common vocabulary for use when describing systems
- Guidance
  - Identify key areas in which services are required
- Prescriptive
  - Define standard protocols and APIs to facilitate creation of interoperable systems and portable applications

Architectural Models

- Simplify and abstract the functions of the individual components of a D/S.
- Represent the placement of the components across a network of computers, seeking to define useful patterns for distribution of data and workload.
- Capture the interrelationships between the components – that is, their functional roles and the patterns of communication between them.

Basic Notions: resource components

- Resource and Computational Components:
  - composing elements of an architecture
- Resource components:
  - Embody architectural elements representing passive data or physical devices
  - Entities meant to be shared
    - E.g., computers, storage, data, software
  - Do not have to be physical entities
  - E.g., Condor pool, distributed file system, …
  - Defined in terms of interfaces, not devices
    - E.g. scheduler such as LSF and PBS define a compute resource
    - Open/close/read/write define access to a distributed file system, e.g. NFS, AFS, DFS

Basic Notions: computational components

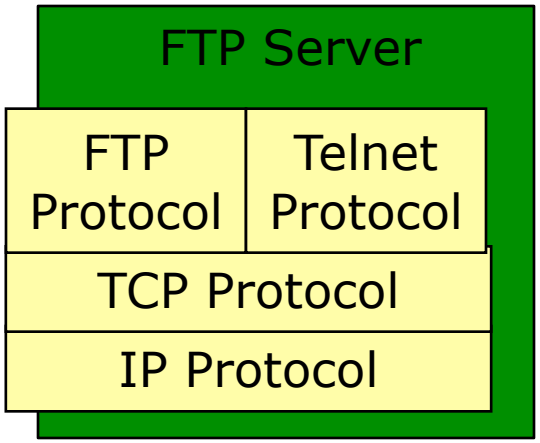
- Embody a flow of control
  - E.g. process, thread
- Characterized by state, which includes:
  - Private data
  - State of execution
  - Bindings to other components (code and resource)
- Examples:
  - Client processes (διεργασίες πελάτη)
  - Server processes (διεργασίες εξυπηρετητή) : a process accepting requests from other processes.
  - Peer processes (διεργασίες ομοτίμων)

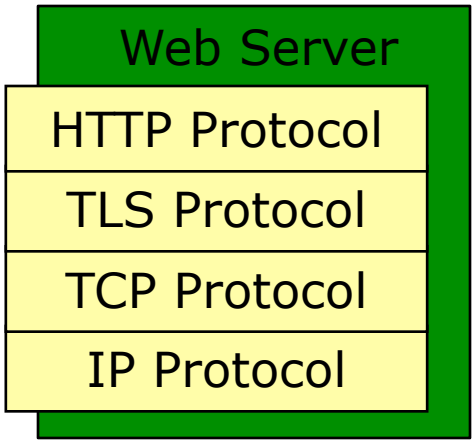
Basic notions: interactions & sites

- Events that involve 2 or more components
  - Message exchanged between 2 computational components
- Sites:
  - Execution environments: they host components and provide support for the execution of computational components.
  - Sites embody the intuitive notion of location.
  - Local vs remote interactions.

Basic Notions: Network Enabled Services

- Implementation of a protocol that defines a set of capabilities
  - Protocol defines interaction with service
  - All services require protocols
  - Not all protocols are used to provide services (e.g. IP)
- Examples: FTP and Web servers





Basic Notions: Network Protocol

- A formal description of message formats and a set of rules for message exchange
  - Rules may define sequence of message exchanges
  - Protocol may define state-change in endpoint, e.g., file system state change
- Good protocols designed to do one thing
  - Protocols can be layered
- Examples of protocols
  - IP, TCP, TLS (was SSL), HTTP, Kerberos

Basic Notions: Application Programming Interface

- A specification for a set of routines to facilitate application development
  - Refers to definition, not implementation
  - E.g., there are many implementations of MPI
- Spec often language-specific (or IDL)
  - Routine name, number, order and type of arguments; mapping to language constructs
  - Behavior or function of routine
- Examples
  - GSS API (security), MPI (message passing)

Basic Notions: Software Development Kit

- A particular instantiation of an API
- SDK consists of libraries and tools
  - Provides implementation of API specification
- Can have multiple SDKs for an API
- Examples of SDKs
  - MPICH, Motif Widgets

Basic Notions: Syntax

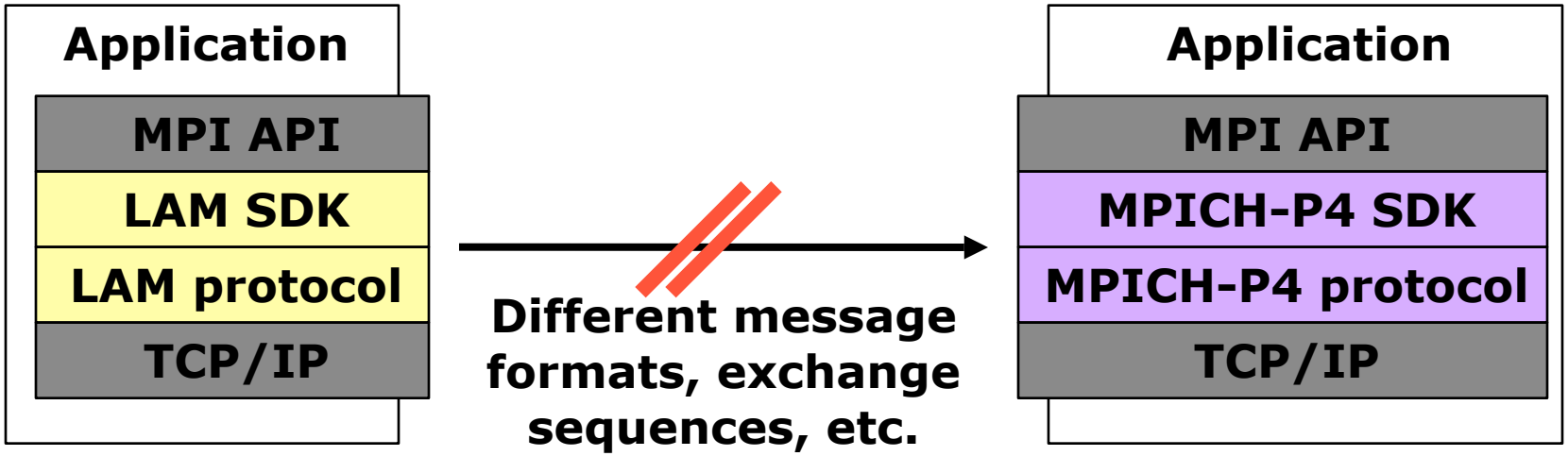
- Rules for encoding information, e.g.
  - XML, Condor ClassAds, Globus RSL
  - X.509 certificate format (RFC 2459)
  - Cryptographic Message Syntax (RFC 2630)
- Distinct from protocols
  - One syntax may be used by many protocols (e.g., XML); & useful for other purposes
- Syntaxes may be layered
  - E.g., Condor ClassAds -> XML -> ASCII
  - Important to understand layerings when comparing or evaluating syntaxes

A Protocol can have Multiple APIs

- TCP/IP APIs include BSD sockets, Winsock, System V streams, …
- The protocol provides interoperability: programs using different APIs can exchange information
- I don't need to know remote user's API

An API can have Multiple Protocols

- MPI provides portability: any correct program compiles & runs on a platform
- Does not provide interoperability: all processes must link against same SDK
  - E.g., MPICH and LAM versions of MPI



APIs and Protocols are Both Important

- Standard APIs/SDKs are important
  - They enable *application portability*
  - But w/o standard protocols, interoperability is hard (every SDK speaks every protocol?)
- Standard protocols are important
  - Enable cross-site *interoperability*
  - Enable shared infrastructure
  - But w/o standard APIs/SDKs, application portability is hard (different platforms access protocols in different ways)

Programming & Systems Problems

- The programming problem
  - Facilitate development of sophisticated apps
  - Facilitate code sharing
  - Requires programming environments
    - APIs, SDKs, tools
- The systems problem
  - Facilitate coordinated use of diverse resources
  - Facilitate infrastructure sharing
    - e.g., certificate authorities, information services
  - Requires systems
    - protocols, services

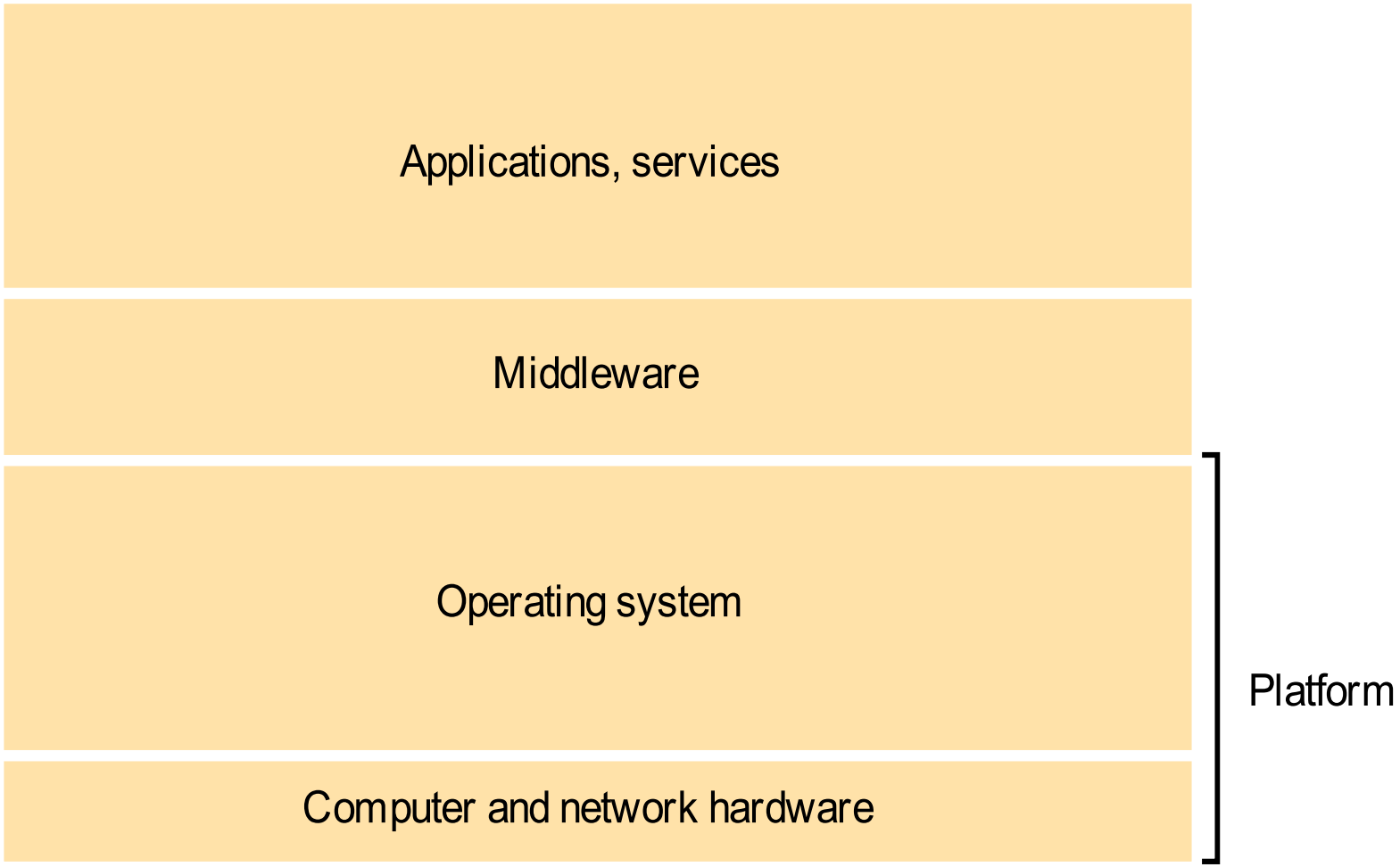
System architecture

- The structuring of software as layers or modules in a single site or in terms of services offered and requested between processes located in the same or different sites.
  - Client-server
  - Client-proxy-server
  - Push model
  - Remote evaluation
  - Code on demand
  - Mobile agent: running program+data migrating

Some basic notions

- **Client** processes (διεργασίες πελάτη)
- **Server** processes (διεργασίες εξυπηρετητή) : a process accepting requests from other processes.
- **Peer** processes (διομότιμες διεργασίες)
- **Software architecture**: the structuring of software as layers or modules in a single computer or in terms of services offered and requested between processes located in the same or different computers.

Software and hardware layers



Some basic notions

- Platforms for D/S and applications: the lowest-level hardware and software layers. Provide services to layers above them.
- Middleware (μεσολογισμικό;) a layer of software whose purpose is to mask heterogeneity and provide a convenient programming model to application programmers.
- Middleware is represented by processes or objects in a set of computers that interact with each other to achieve communication and resource sharing.

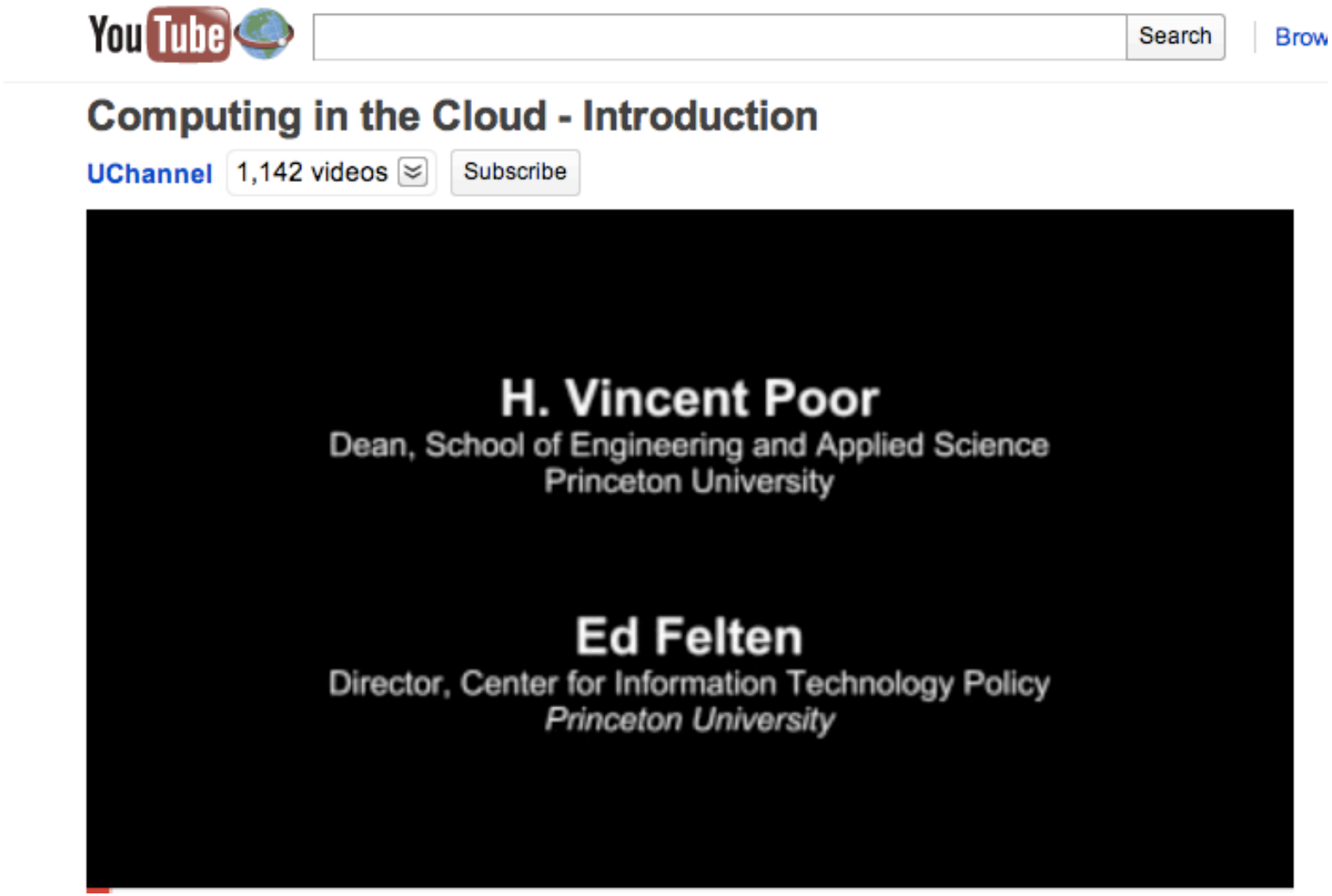
Some basic notions

- Middleware is concerned with providing useful building blocks for the construction of software components that can work with one another in a D/S.
- It enables communication at higher levels of abstraction by providing things like:
  - remote method invocation
  - group communication
  - event notification
  - data replication
  - real-time transmission of data.

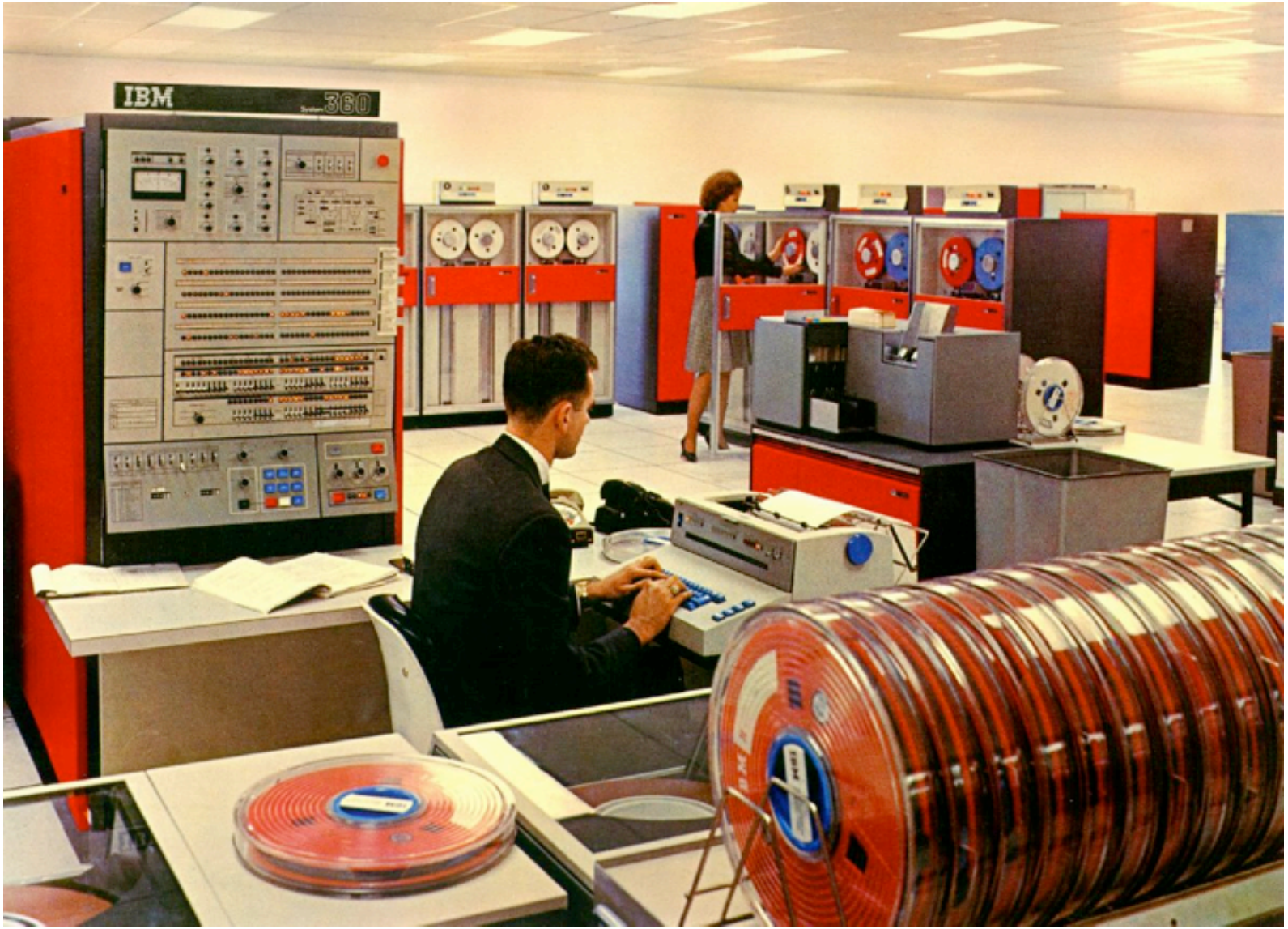
Some basic notions

- Middleware can also provide infrastructural services for use by application programs:
  - Naming
  - Security
  - Transactions
  - Persistent storage
  - Event notification

From centralized to client-server

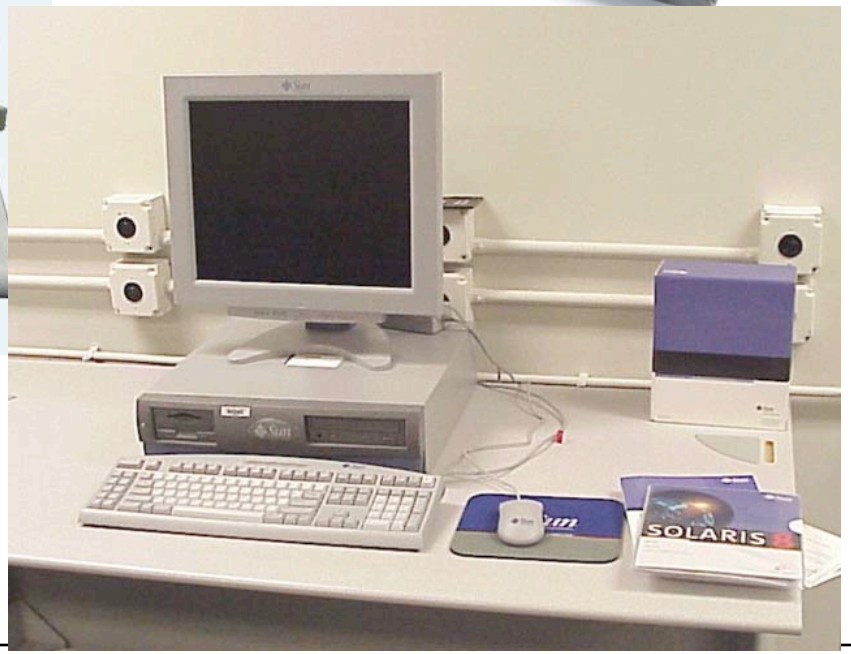


Centralization: mainframes



Centralization: time-sharing

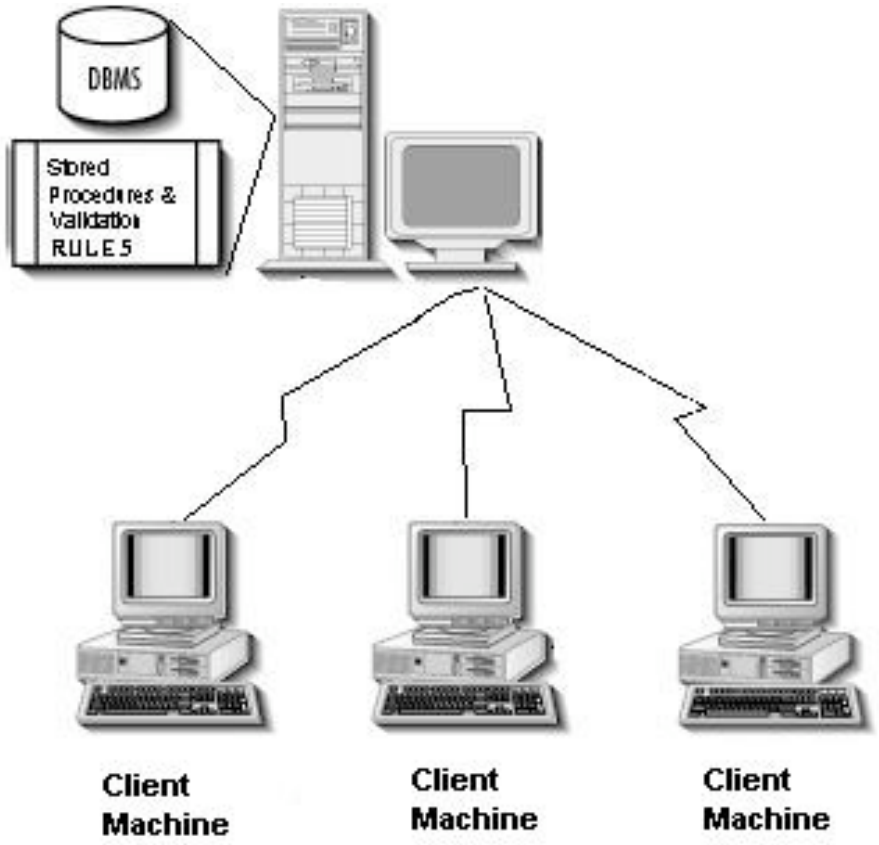
Personal Computing: some degree of decentralization (thanks to Moore's law)



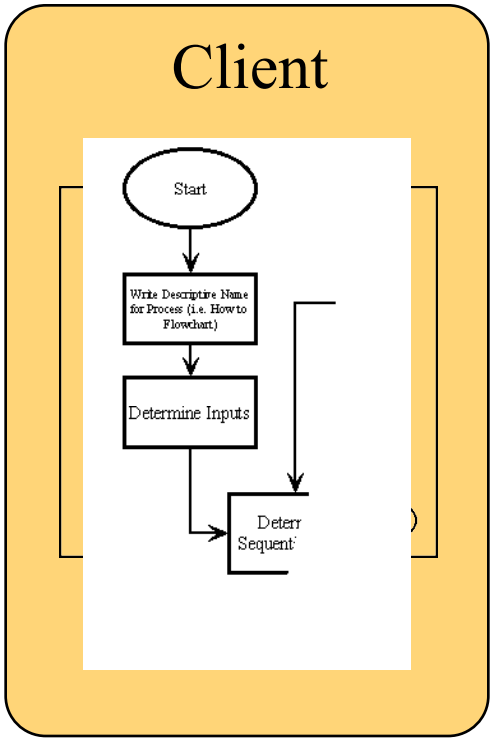
Centralized vs Distributed

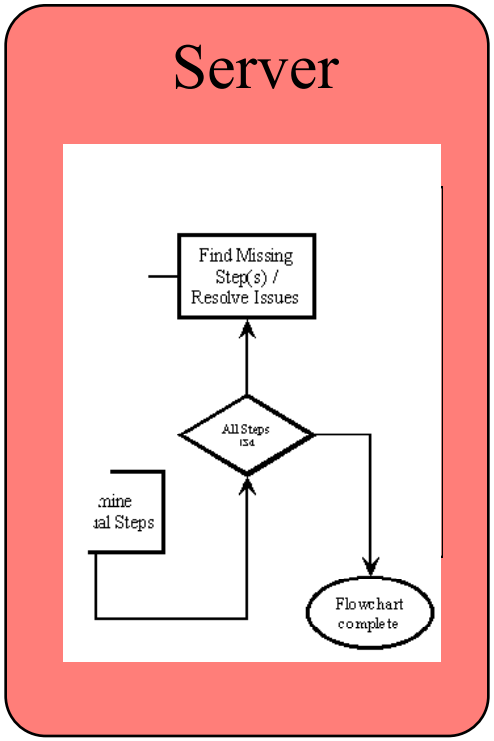
- Centralized computing
  - Low cost: time-sharing, amortizing building cost
  - Expert management
- Distributed approach:
  - Richer user interface: higher bandwidth available locally
  - Greater autonomy

Client-server Computing



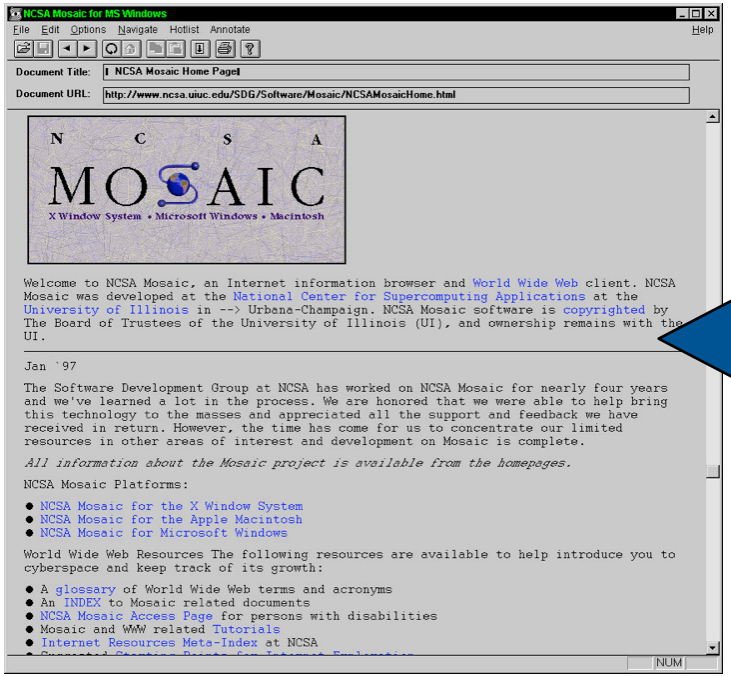
“Client-server”





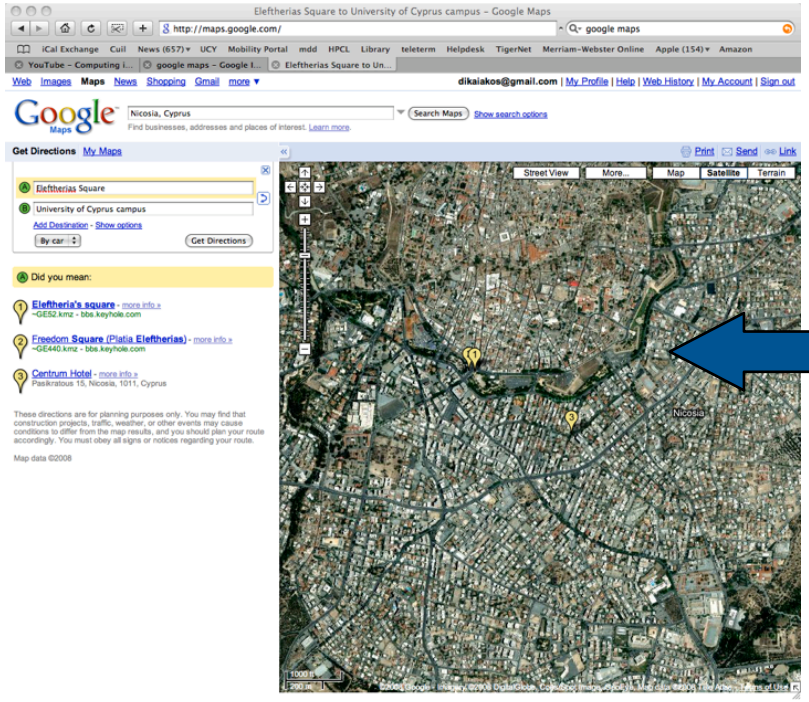
Not very popular for the masses

Browser

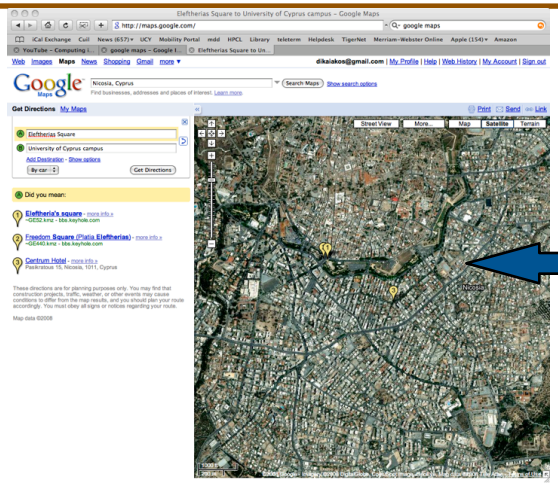


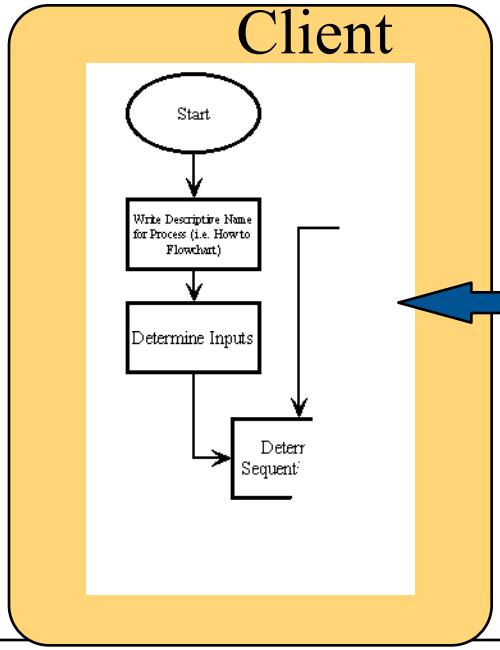
The browser gradually became a “platform”

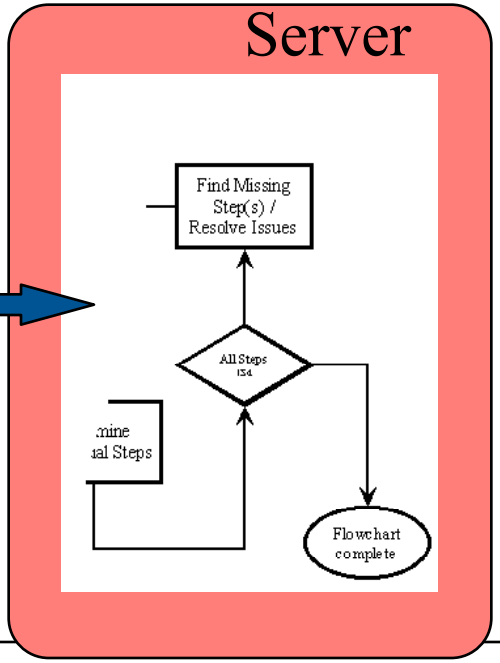
Browsers as platforms



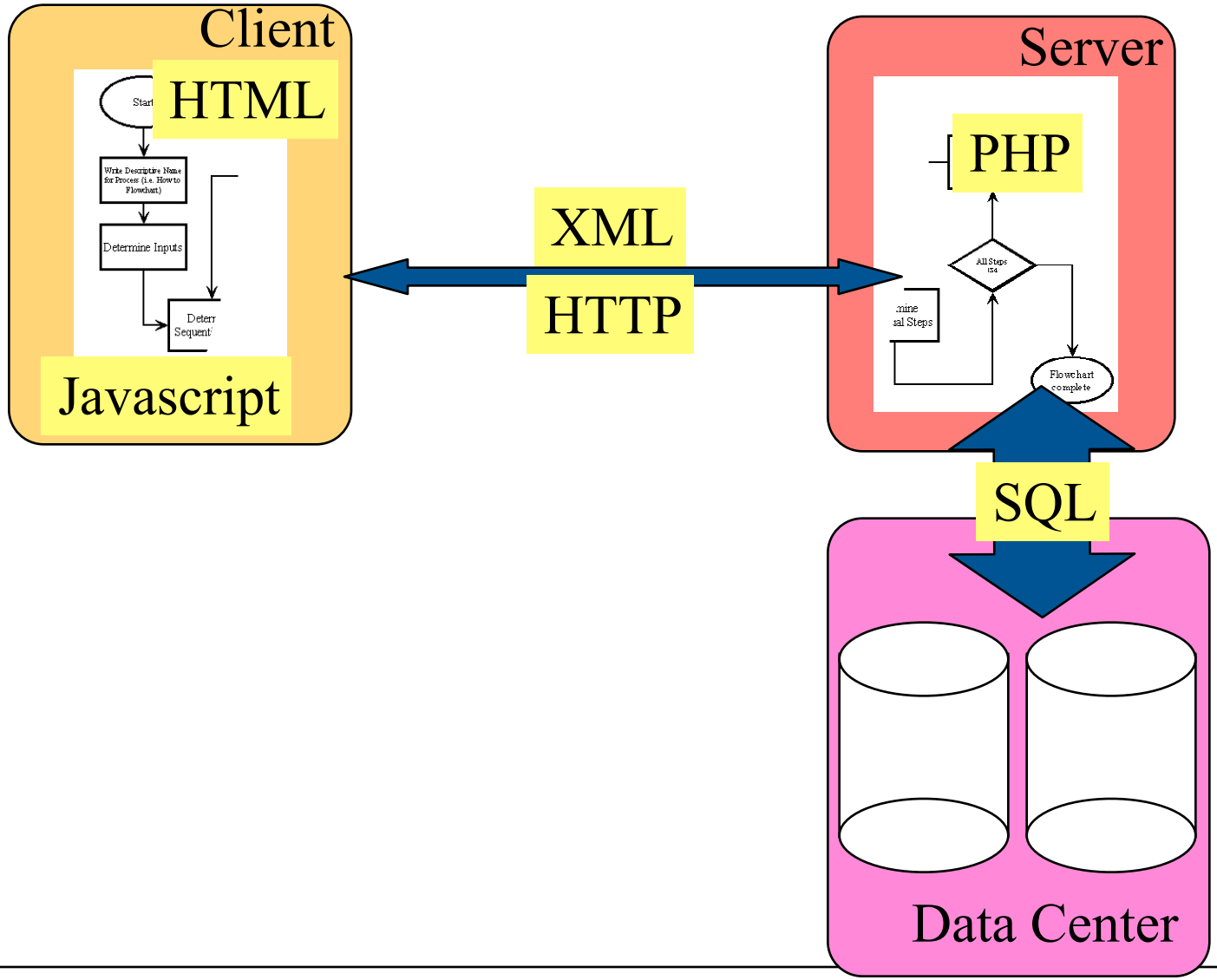
True client-server computing with browsers







What happens behind the schene?



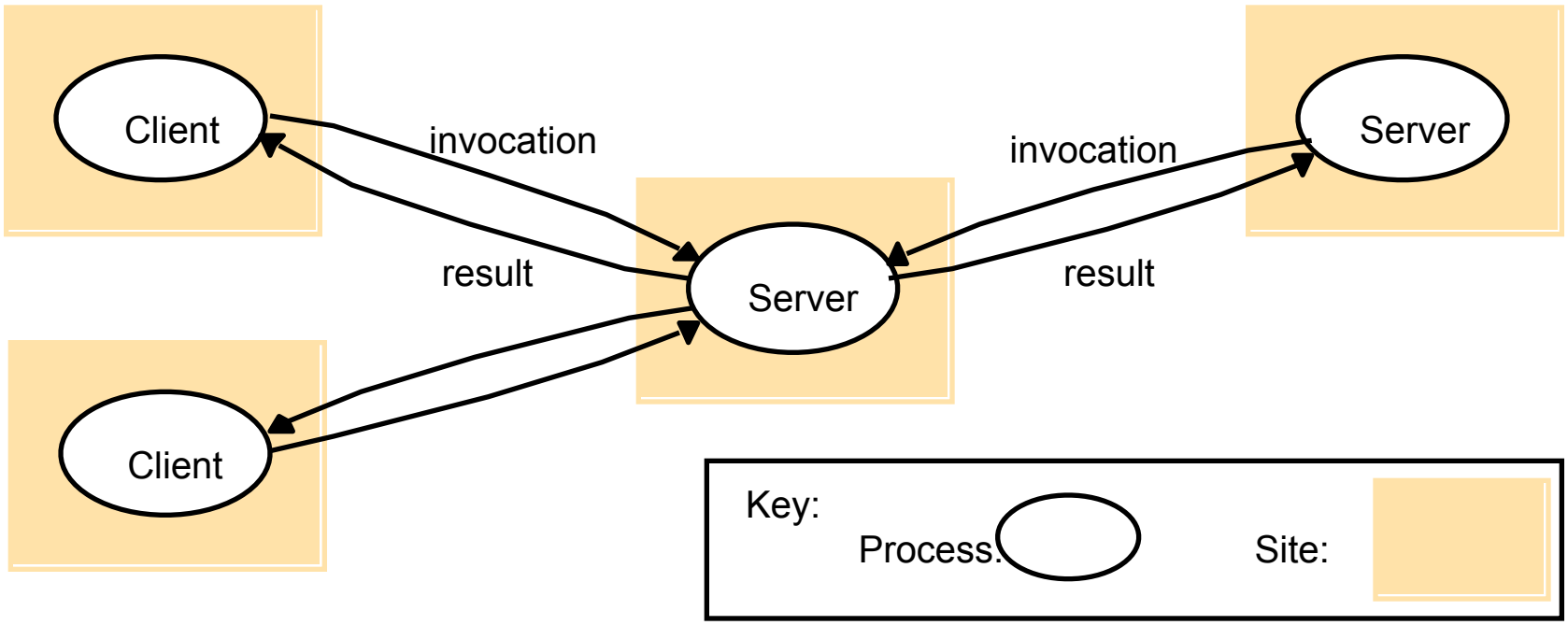
System Architecture

- Client-server model.
- Services provided by multiple servers in separate hosts.
  - Partitioned data: Web server example.
  - Replicated data: NIS example.
- Replication is used to increase performance and availability and to improve fault tolerance – provides multiple consistent copies of data in processes running in different computers.
- At what cost?

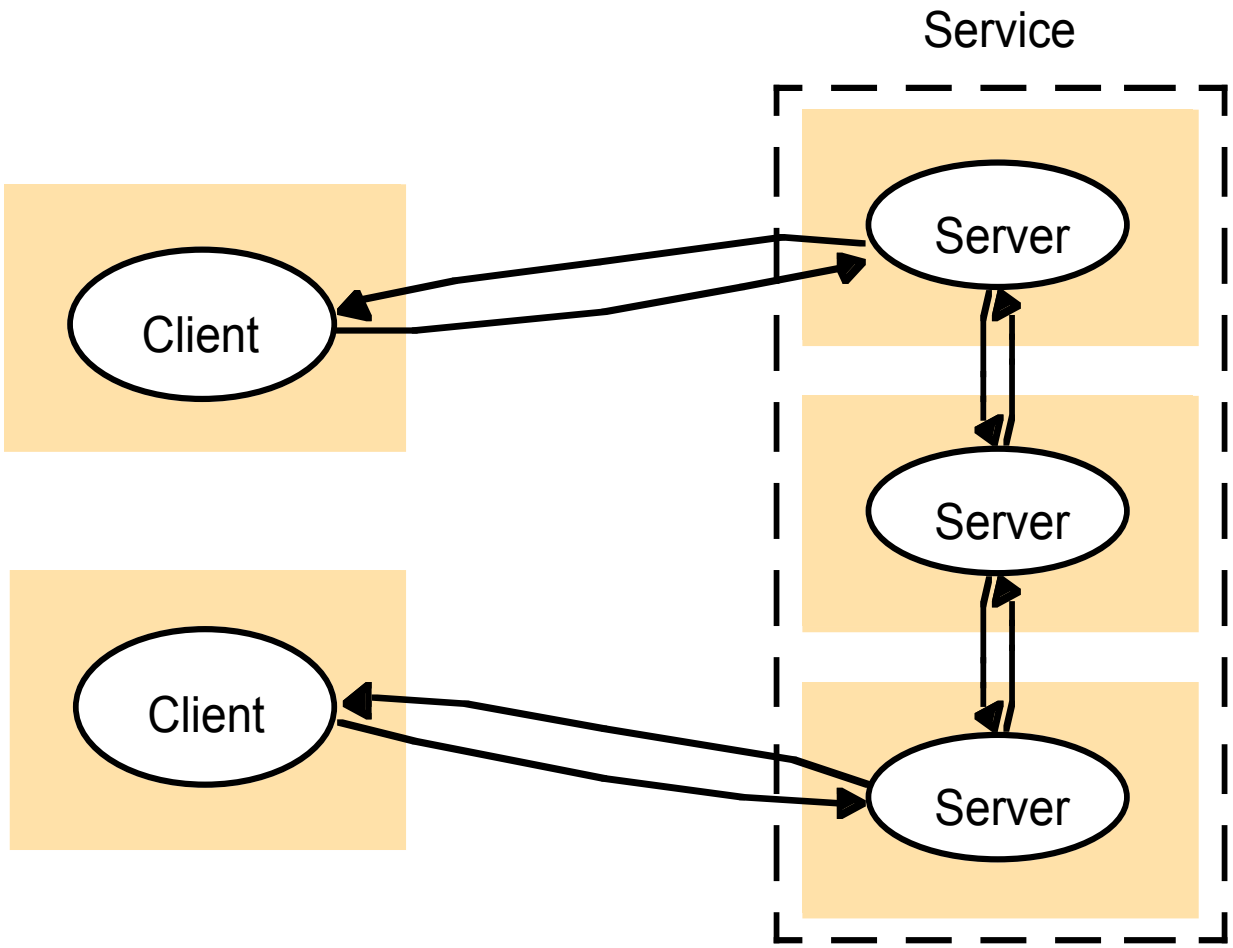
Variations of Client-Server

- Mobile code, e.g., applets (Code-on-demand).
- Push model: the server instead of the client initiates interactions.
- Mobile agents: running program+data migrating.

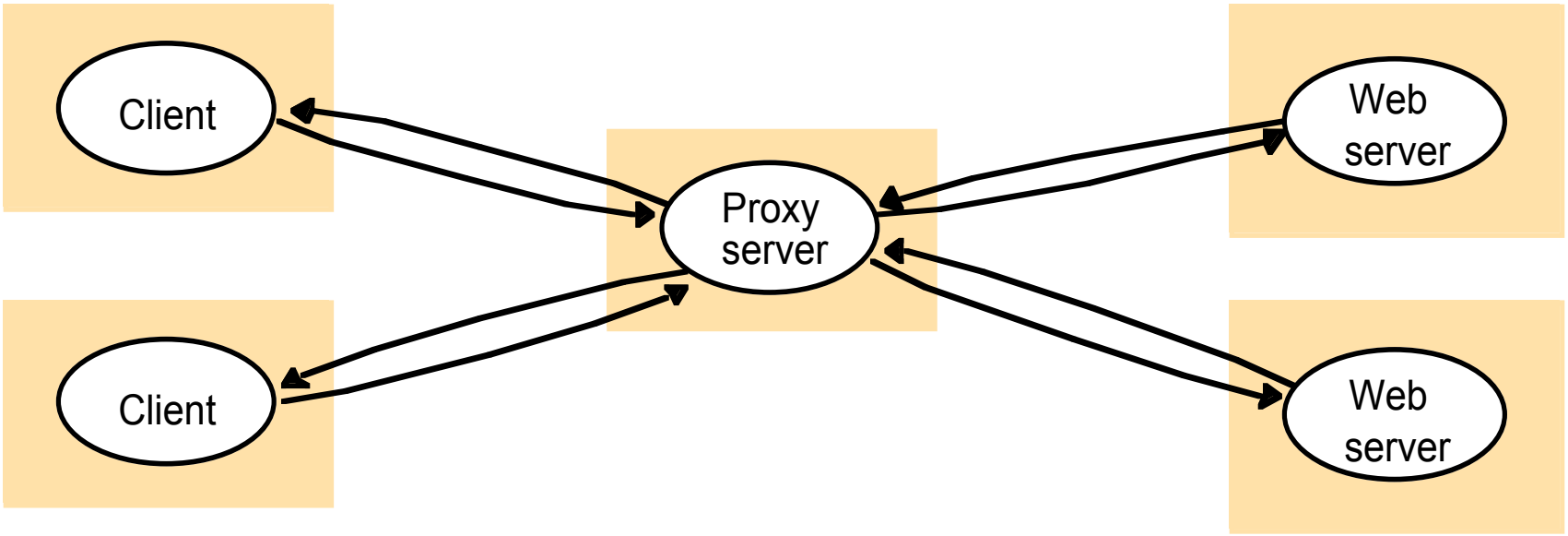
Clients invoke individual servers



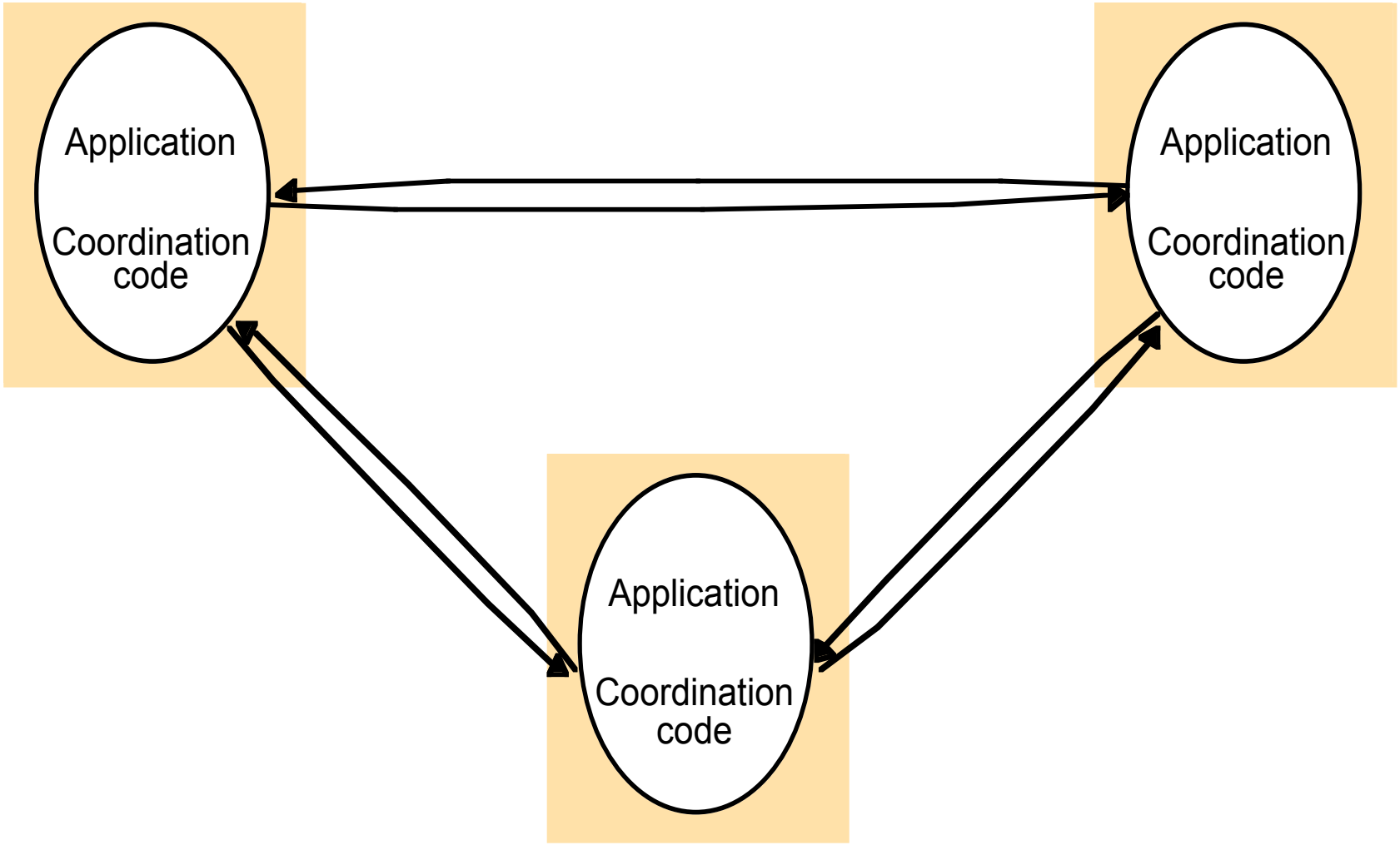
A service provided by multiple servers



Web proxy server

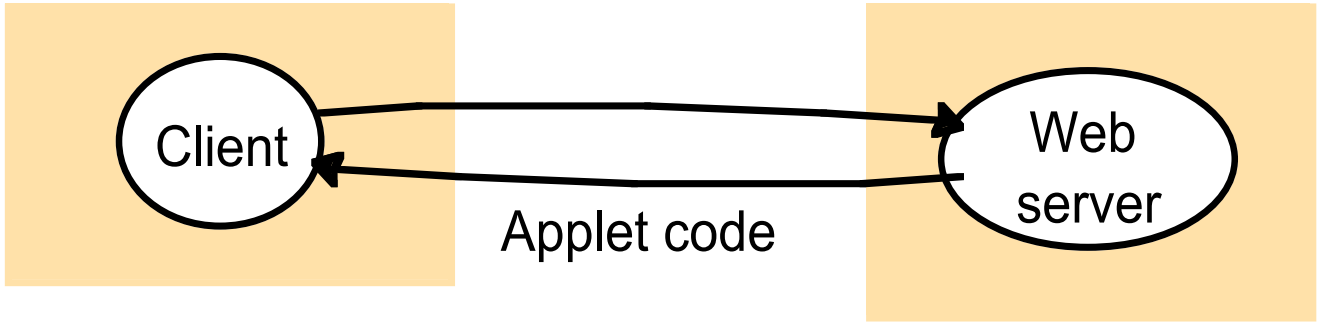


A distributed application based on peer processes



Web applets

a) client request results in the downloading of applet code



b) client interacts with the applet

Designing Distributed Applications with Mobile Code Paradigms

A. Carzaniga, G.P. Picco and G. Vigna

Proceedings of the 19th International Conference in Software Engineering, 1997.

Code Mobility

- The capability to reconfigure *dynamically,* at run time, the binding between the software components of an application and their physical location within a computer network.
- Two types of mobility:
  - Strong mobility: the ability of a MCL to allow Execution Units to move their code and execution state to a different site.
  - Weak mobility: the ability of a MCL to allow an EU in a site to be bound dynamically to code coming from a different site.

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)

Site B: Christine

Site A: Louise

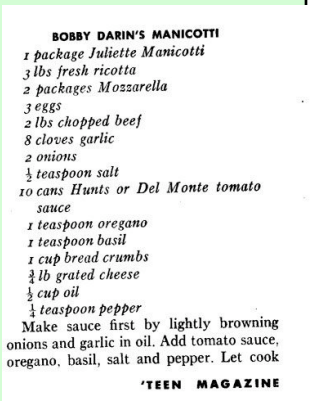
Request of cake

Wants to
Eat cake

Read the recipe

Bake the cake

Deliver the cake



BOBBY DARIN'S MANICOTTI
1 package Juliette Manicotti
3 lbs fresh ricotta
2 packages Mozzarella
3 eggs
2 lbs chopped beef
8 cloves garlic
2 onions
½ teaspoon salt
10 cans Hunts or Del Monte tomato sauce
1 teaspoon oregano
1 teaspoon basil
1 cup bread crumbs
¾ lb grated cheese
½ cup oil
¼ teaspoon pepper
Make sauce first by lightly browning onions and garlic in oil. Add tomato sauce, oregano, basil, salt and pepper. Let cook
'TEEN MAGAZINE

Client

Server

X Windows System

Remote Evaluation Model: (REV)

Site A: Louise

Wants to Eat cake

BOBBY DARIN'S MANICOTTI

1 package Juliette Manicotti
3 lbs fresh ricotta
2 packages Mozzarella
3 eggs
2 lbs chopped beef
8 cloves garlic
2 onions
½ teaspoon salt
10 cans Hunts or Del Monte tomato sauce
1 teaspoon oregano
1 teaspoon basil
1 cup bread crumbs
¾ lb grated cheese
½ cup oil
¼ teaspoon pepper

Make sauce first by lightly browning onions and garlic in oil. Add tomato sauce, oregano, basil, salt and pepper. Let cook

'TEEN MAGAZINE

Recipe

Get the recipe

Bake the cake

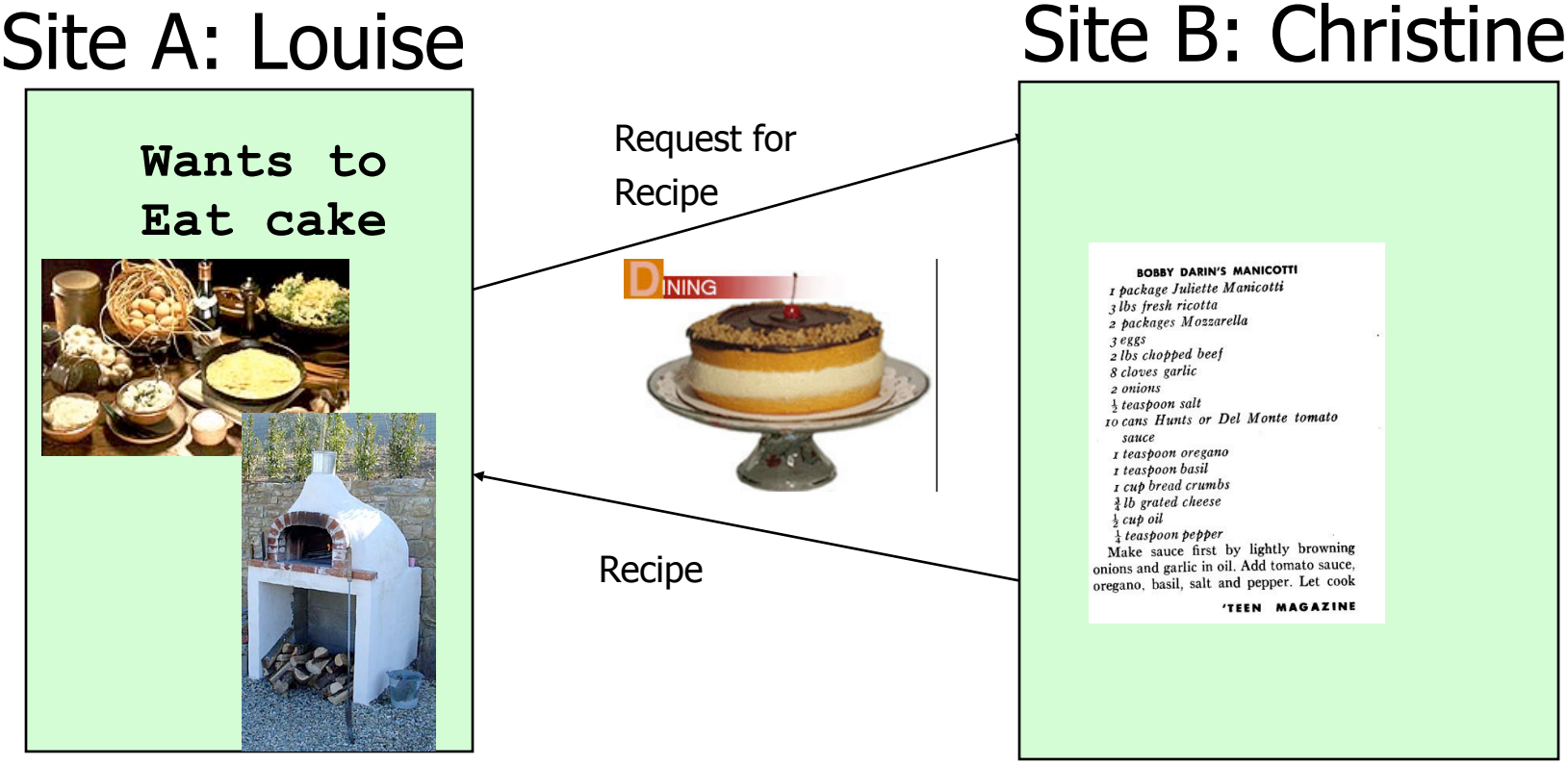
Deliver the cake

Site B: Christine

Unix: rsh command

PostScript printer

Code on Demand (COD)



Terminal 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: Louise

BOBBY DARIN'S MANICOTTI

1 package Juliette Manicotti
3 lbs fresh ricotta
2 packages Mozzarella
3 eggs
2 lbs chopped beef
8 cloves garlic
2 onions
½ teaspoon salt
10 cans Hunts or Del Monte tomato sauce
1 teaspoon oregano
1 teaspoon basil
1 cup bread crumbs
¾ lb grated cheese
½ cup oil
¼ teaspoon pepper

Make sauce first by lightly browning onions and garlic in oil. Add tomato sauce, oregano, basil, salt and pepper. Let cook

'TEEN MAGAZINE

Louise moves to Site B along with recipe and ingredients

Cake

Site B: Christine

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

Choosing the Right Paradigm

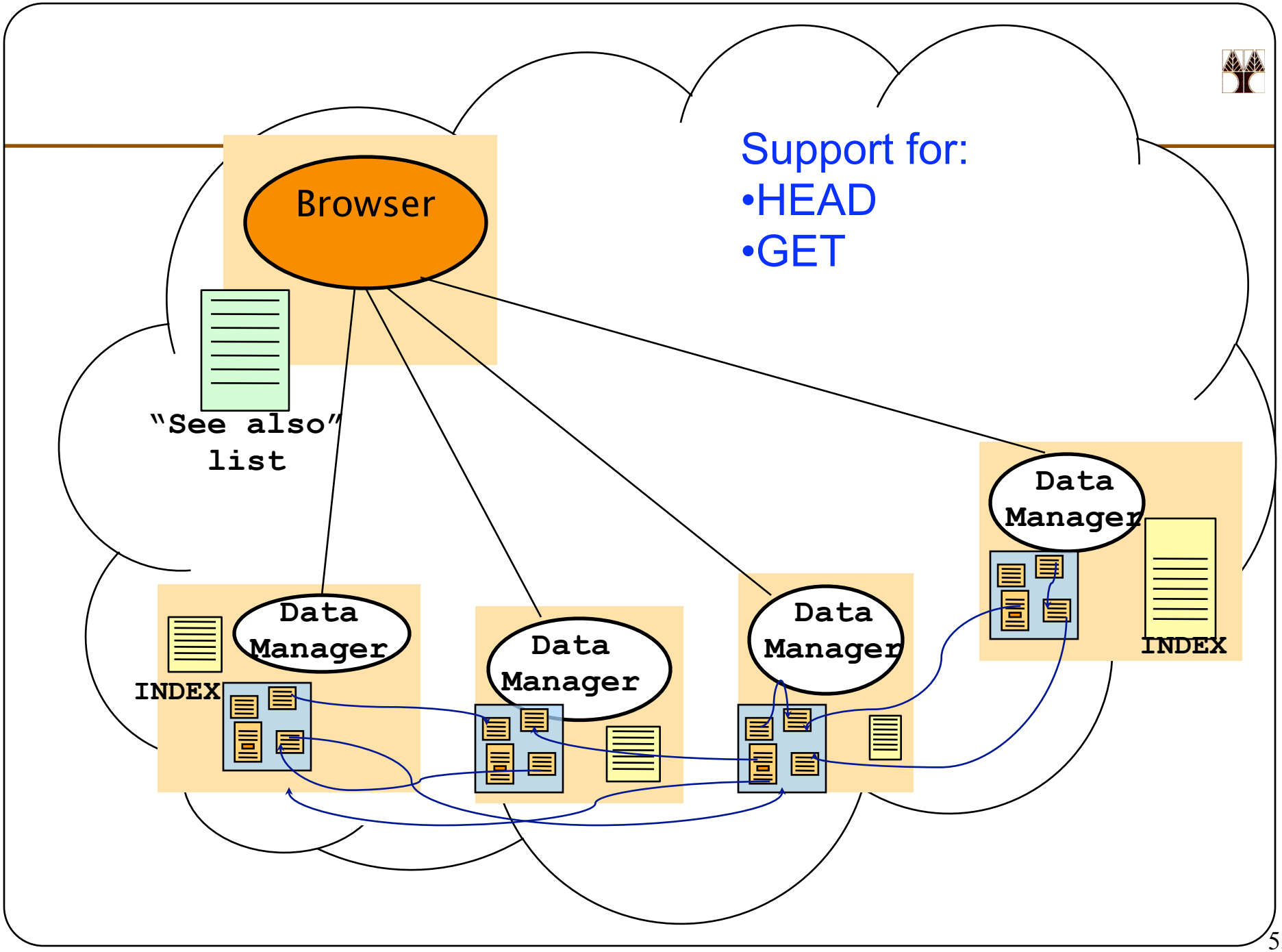
- No paradigm is absolutely better than others.
- The choice of paradigm must be performed on a case-by-case basis, taking into account issues such as the cost of network communication, availability and performance of resources, etc.

Case study: Information Retrieval Application

- Data managers:
  - Store a set of documents
  - Store an index of documents kept in their storage
  - Documents contain:
    - Document header: contains update time and keywords
    - Document content: contains text and links to other Data Managers that contain interesting documents
- Browser:
  - Given a description of interests, searches for data-manager nodes to find and retrieve relevant documents
  - Looks into documents to discover new data-manager nodes

Browser Algorithm

- Data structure:
  - Browser maintains a “see also” list of nodes to be visited
  - This list should be initialized with a “seed” of known nodes (at least one node in the seed).
- Browser Algorithm:
  - Extract the first node from the “see also” list
  - Loop:
    - Contact the node and query its data manager to retrieve an index of documents stored
    - Request the headers of all documents that are present on that node
    - Check which documents are “interesting” and retrieve their bodies
    - Extract links to other nodes from retrieved documents and insert them into the “see also” list.


Support for:
•HEAD
•GET
Browser
"See also" list
Data Manager
INDEX
Data Manager
Data Manager
Data Manager
INDEX

Case Study: Information Retrieval Application

Assumptions:

- Sites communicate by exchanging messages with a reliable protocol.
- 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

- Each node holds same number D of documents
- The *relevant* information is uniformly distributed among a set of N nodes
- *i* is the ratio between relevant and total documents.
- Documents have constant length. *h* and *b* are the size of the header and the body, respectively.
- All requests have a fixed length (*r):* message header size and all auxiliary data of the request/ reply

Client - Server

- For each node, browser issues:
  - D requests for doc headers
  - *i* x D requests for doc bodies

$$Tcs = (D + i\ D)\ r\ N + (D\ h + i\ D\ b)\ N$$

Remote Evaluation

- Remote Evaluation could perform the filtering task on the node!

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

$$Trev = (r + Crev + i\ D\ b)\ N$$

Mobile Agent

- The browser migrates on each relevant node
- Performs filtering locally
- Saves the state of all relevant information and the see-also list
- At each hop, the mobile agent carries its code and state across the network.

Mobile Agent (ctd)

- Traffic for each hop of the MA:
  - $T^j = r + Cma + S^j$ where:
    - Cma -- *size of the agent*
    - $S^j = d_{SAlist} + s + \sum_1^j (i\ D\ b)$ -- *state carried with the agent*
    - s -- *size of internal data structures representing the state of the computation*
    - $\sum_1^j (i\ D\ b)$ -- *information collected by the agent at each visited node*
- Assumming that i, D, b, d_{SAlist} and s do not depend the node, we have:
  - $T_{MA} = \sum_{j=0}(r + Cma + s + \sum_1^j (i\ D\ b))$
  - $T_{MA} = (r + Cma + s + NiDb/2)(N+1)$

Comparison

- Useful information: *I = iDbN*
- *Overheads:*

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