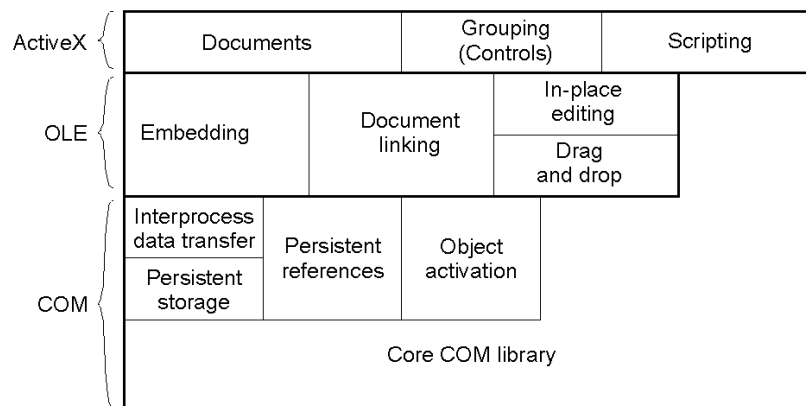


# Today: More Case Studies

- DCOM
- Jini

## DCOM

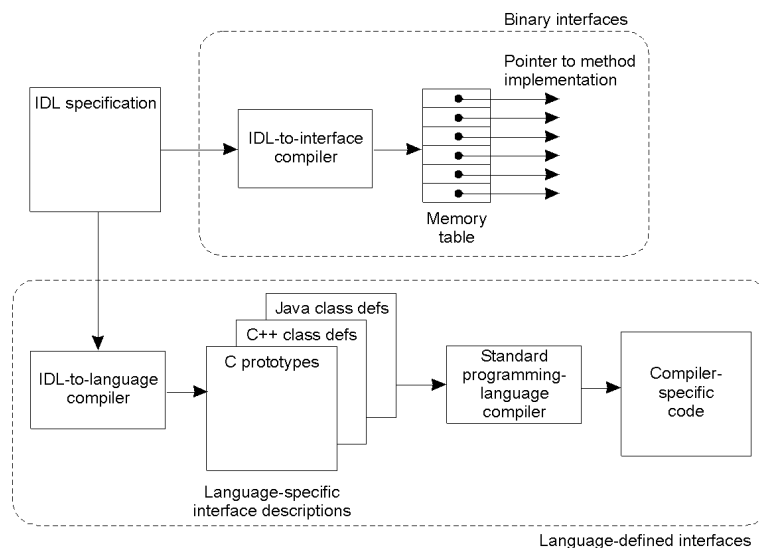
- Distributed Component Object Model
  - Microsoft's object model (middleware)



# DCOM: History

- Successor to COM
  - Developed to support compound documents
    - Word document with excel spreadsheets and images
- Object linking and embedding (OLE)
  - Initial version: message passing to pass information between parts
  - Soon replaced by a more flexible layer: COM
- ActiveX: OLE plus new features
  - No good consensus on what exactly does ActiveX contain
  - Loosely: groups capabilities within applications to support scripting, grouping of objects.
- DCOM: all of the above, but across machines

## Object Model



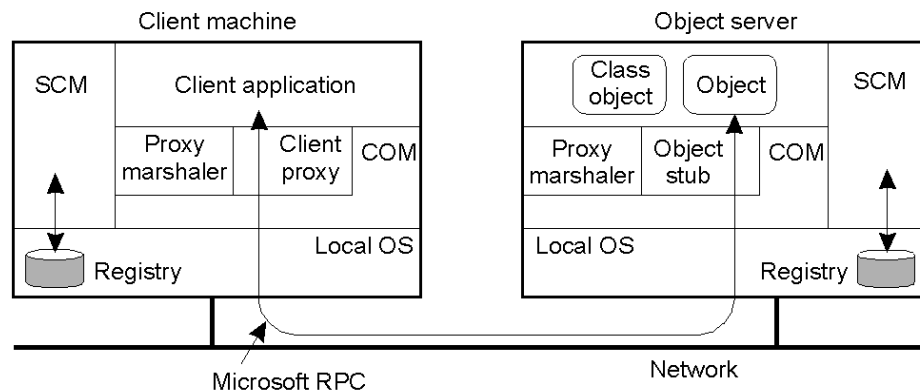
- The difference between language-defined and binary interfaces.

# DCOM Object Model

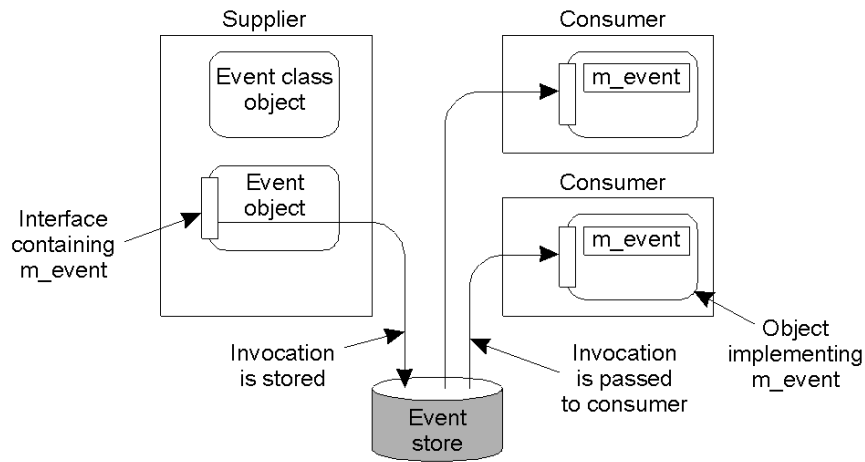
- DCOM: uses remote object model
- Supports only binary interfaces
  - Table of pointers to methods
  - Uses Microsoft IDL
- Unlike CORBA, all objects are transient
  - Delete an object with `refcount == 0`
- Like CORBA, DCOM supports dynamic object invocation

## Type Library and Registry

- The overall architecture of DCOM.
  - Type library == CORBA interface repository
  - Service control manager == CORBA implementation repository

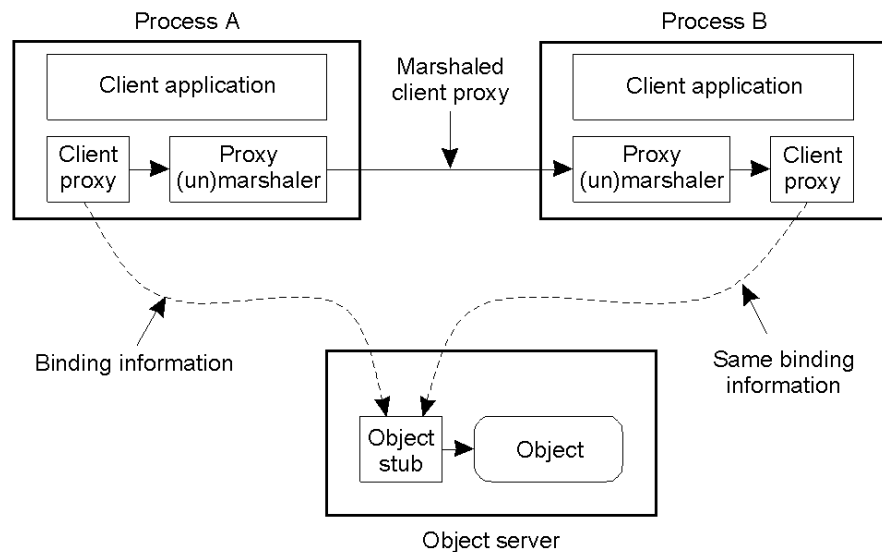


# Events and Messaging



- Event processing in DCOM: publish/subscribe paradigm
- Persistent asynchronous communication: MSFT Message Queuing

# Clients



- Passing an object reference in DCOM with custom marshaling.

# Monikers: Persistent Objects

Step	Performer	Description
1	Client	Calls BindMoniker at moniker
2	Moniker	Looks up associated CLSID and instructs SCM to create object
3	SCM	Loads class object
4	Class object	Creates object and returns interface pointer to moniker
5	Moniker	Instructs object to load previously stored state
6	Object	Loads its state from file
7	Moniker	Returns interface pointer of object to client

- By default, DCOM objects are transient
- Persistent objects implemented using monikers (reference stored on disk)
  - Has all information to recreate the object at a later time



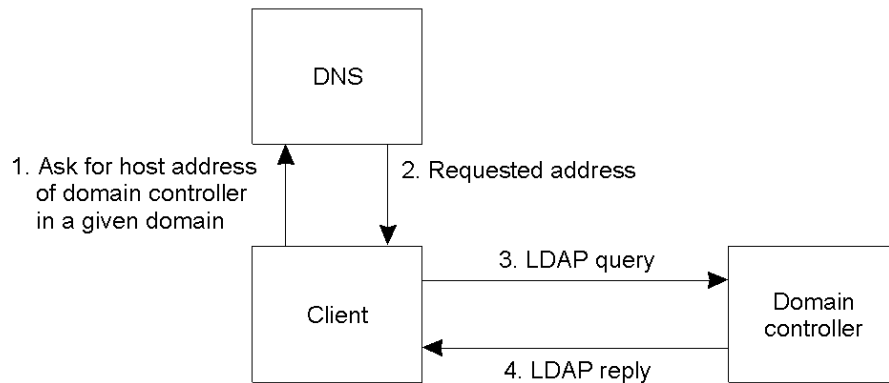
## Monikers (2)

Moniker type	Description
File moniker	Reference to an object constructed from a file
URL moniker	Reference to an object constructed from a URL
Class moniker	Reference to a class object
Composite moniker	Reference to a composition of monikers
Item moniker	Reference to a moniker in a composition
Pointer moniker	Reference to an object in a remote process

- DCOM-defined moniker types.



# Naming: Active Directory



- The general organization of Active Directory
  - Implemented using LDAP
  - Distr. System partitioned into domains (uses domain controllers)
  - Each domain controller has a DNS name
  - DC registered as LDAP services in DNS



# Distributed Coordination

- Motivation
  - Next generation of systems will be inherently distributed
  - Main problem: techniques to coordinate various components
    - Emphasis on coordination of activities between components



# Introduction to Coordination Models

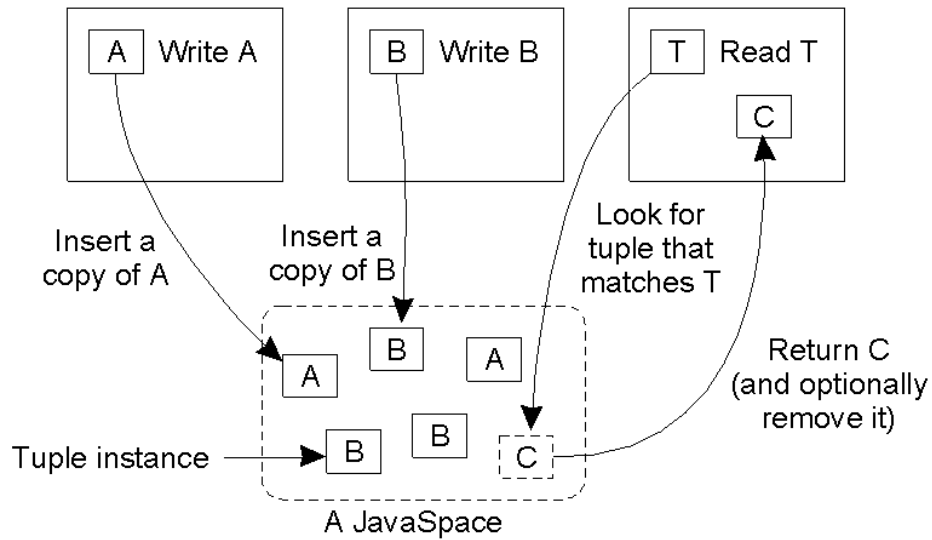
- Key idea: separation of computation from coordination
- A taxonomy of coordination models
  - Direct coordination
  - Mailbox coordination
  - Meeting-oriented coordination (publish/subscribe)
  - Generative (shared tuple space)

		Temporal	
		Coupled	Uncoupled
Referential	Coupled	Direct	Mailbox
	Uncoupled	Meeting oriented	Generative communication

## Jini Case Study

- Coordination system based on Java
  - Clients can *discover* new services as they become available
  - Example: “intelligent toaster”
  - Distributed event and notification system
- Coordination model
  - Uses JavaSpaces: a shared dataspace that stores tuples
    - Each tuple points to a Java object

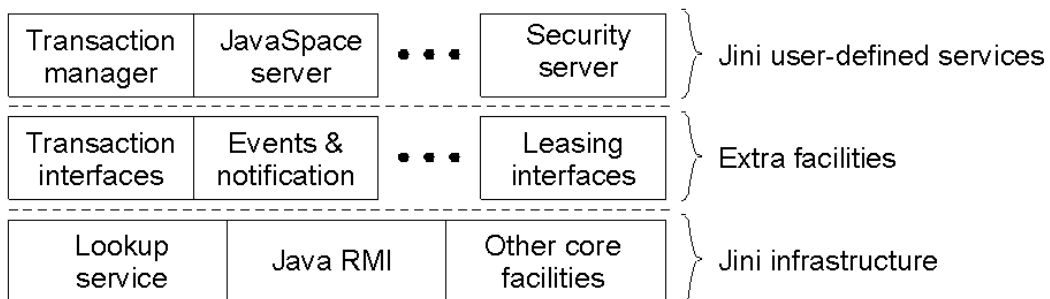
# Overview of Jini



- The general organization of a JavaSpace in Jini.



# Architecture

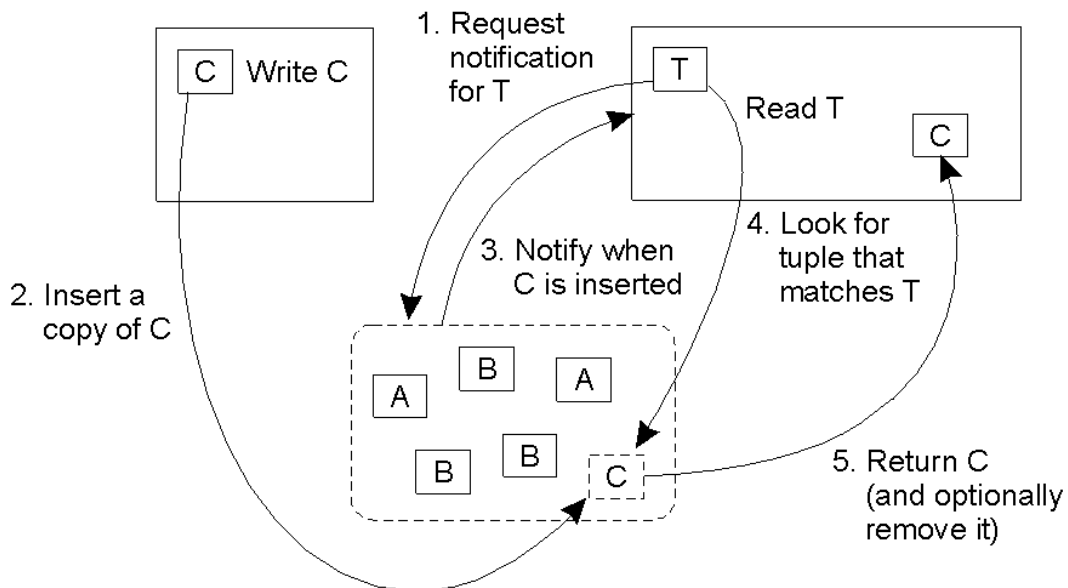


- The layered architecture of a Jini System.





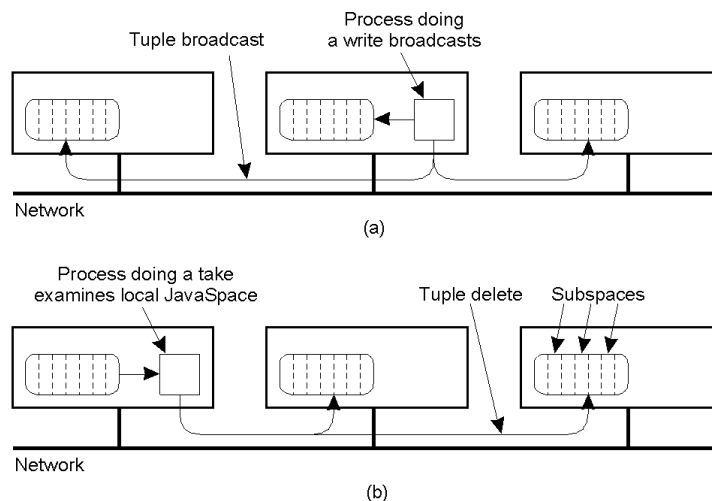
# Communication Events



- Using events in combination with a JavaSpace



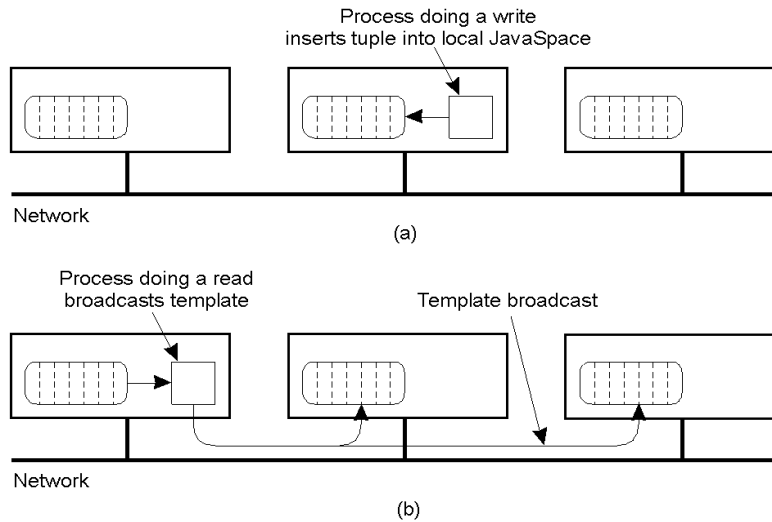
# Processes (1)



- A JavaSpace can be replicated on all machines. The dotted lines show the partitioning of the JavaSpace into subspaces.
- a) Tuples are broadcast on WRITE
- b) READS are local, but the removing of an instance when calling TAKE must be broadcast



# Processes (2)



- Unreplicated JavaSpace.
- a) A WRITE is done locally.
- b) A READ or TAKE requires the template tuple to be broadcast in order to find a tuple instance



# The Jini Lookup Service (1)

Field	Description
ServiceID	The identifier of the service associated with this item.
Service	A (possibly remote) reference to the object implementing the service.
AttributeSets	A set of tuples describing the service.

- The organization of a service item.



# The Jini Lookup Service (2)

<b>Tuple Type</b>	<b>Attributes</b>
ServiceInfo	Name, manufacturer, vendor, version, model, serial number
Location	Floor, room, building
Address	Street, organization, organizational unit, locality, state or province, postal code, country

- Examples of predefined tuples for service items.

