ECE151 – Lecture 16

Chapter 9
Distributed Object-Based Systems
DCOM

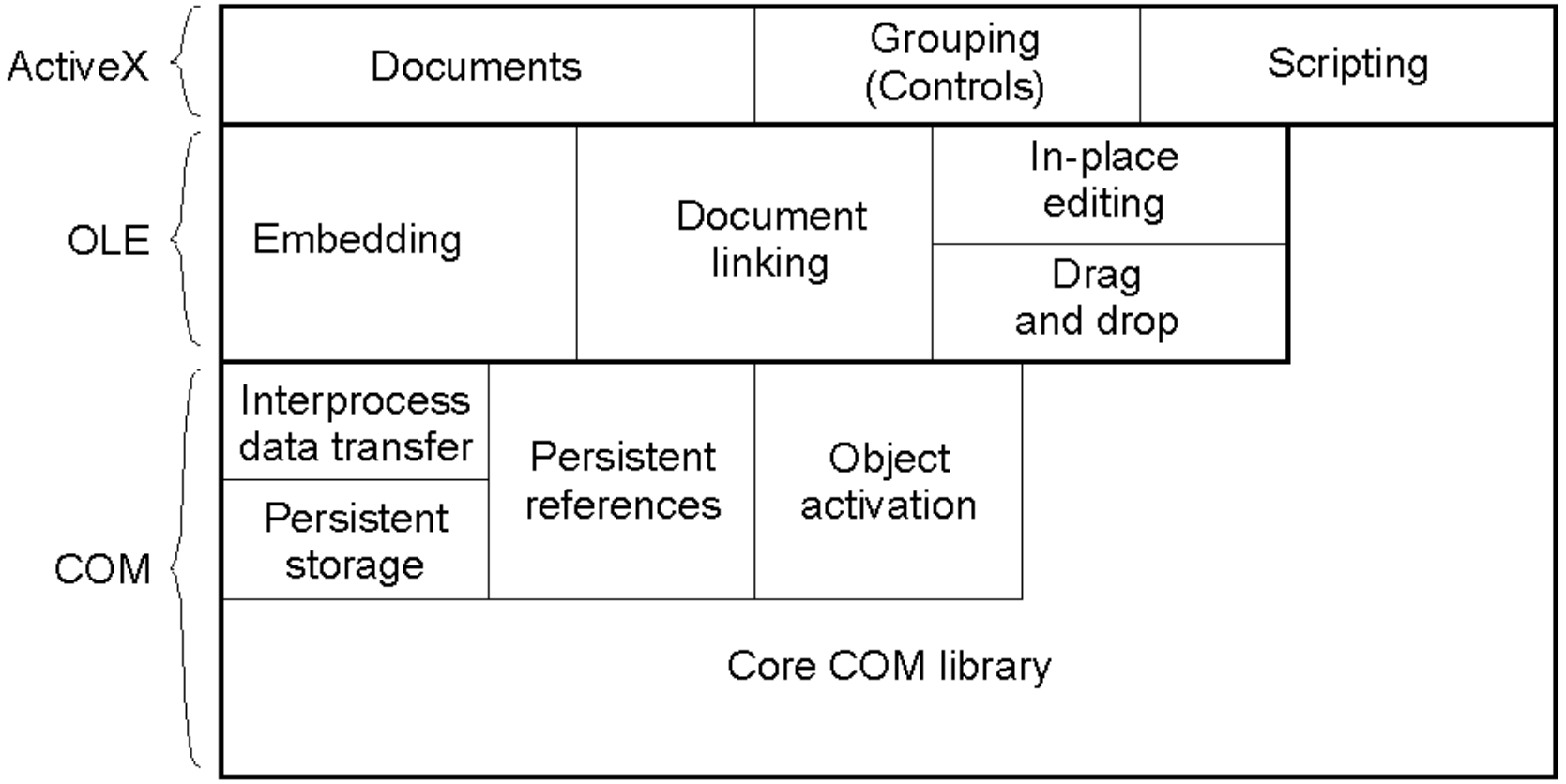
Distributed COM

DCOM: Distributed Component Object Model

- Microsoft's solution to establishing inter-process communication, possibly across machine boundaries.
- Supports a primitive notion of distributed objects
- Evolved from early Windows versions to current NT-based systems (including Windows 2000)
- Comparable to CORBA's object request broker

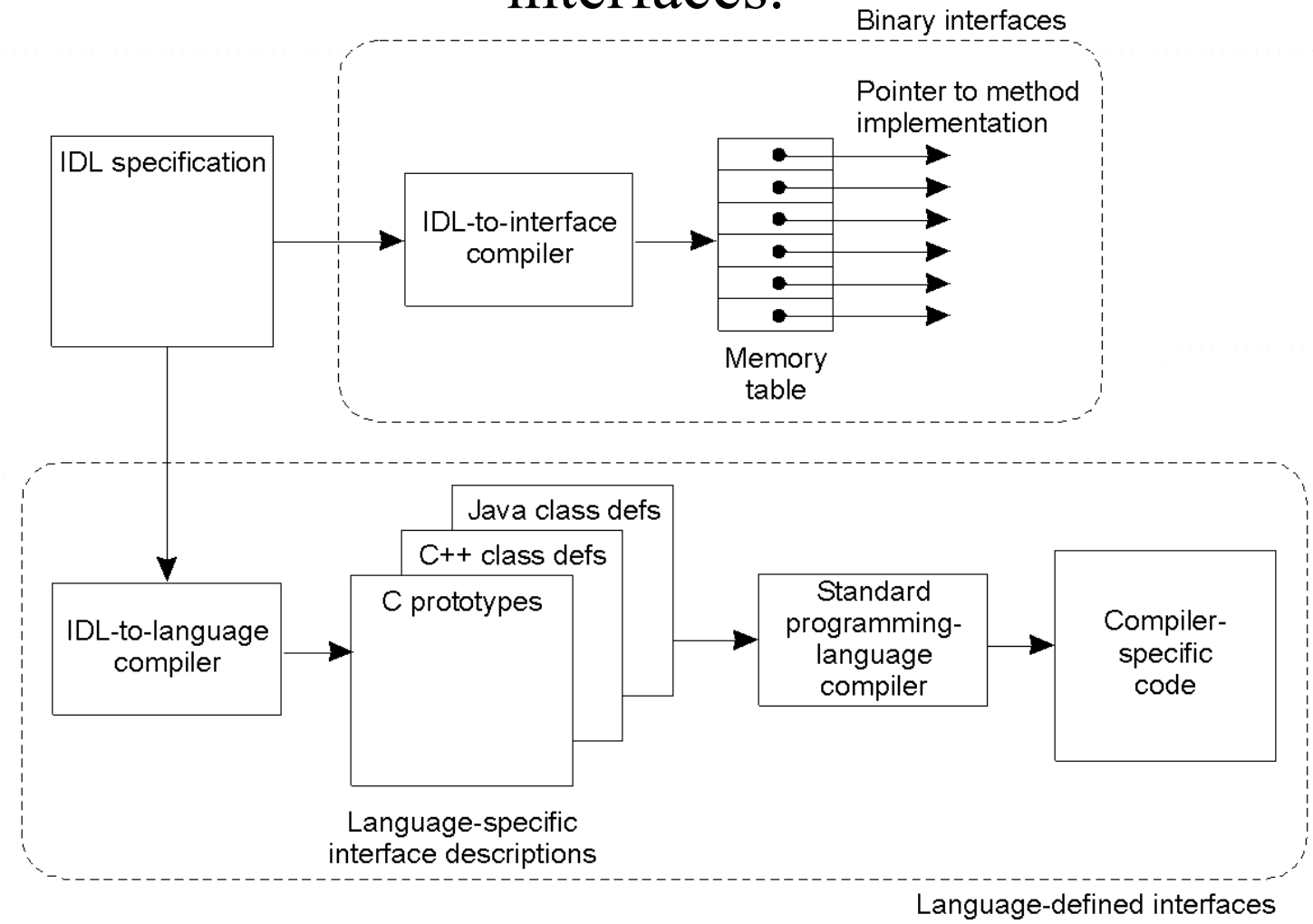
Overview of DCOM

Somewhat confused? DCOM is related to many things that have been introduced by Microsoft in the past couple of years: Adds facilities to communicate across process and machine boundaries.



Object Model

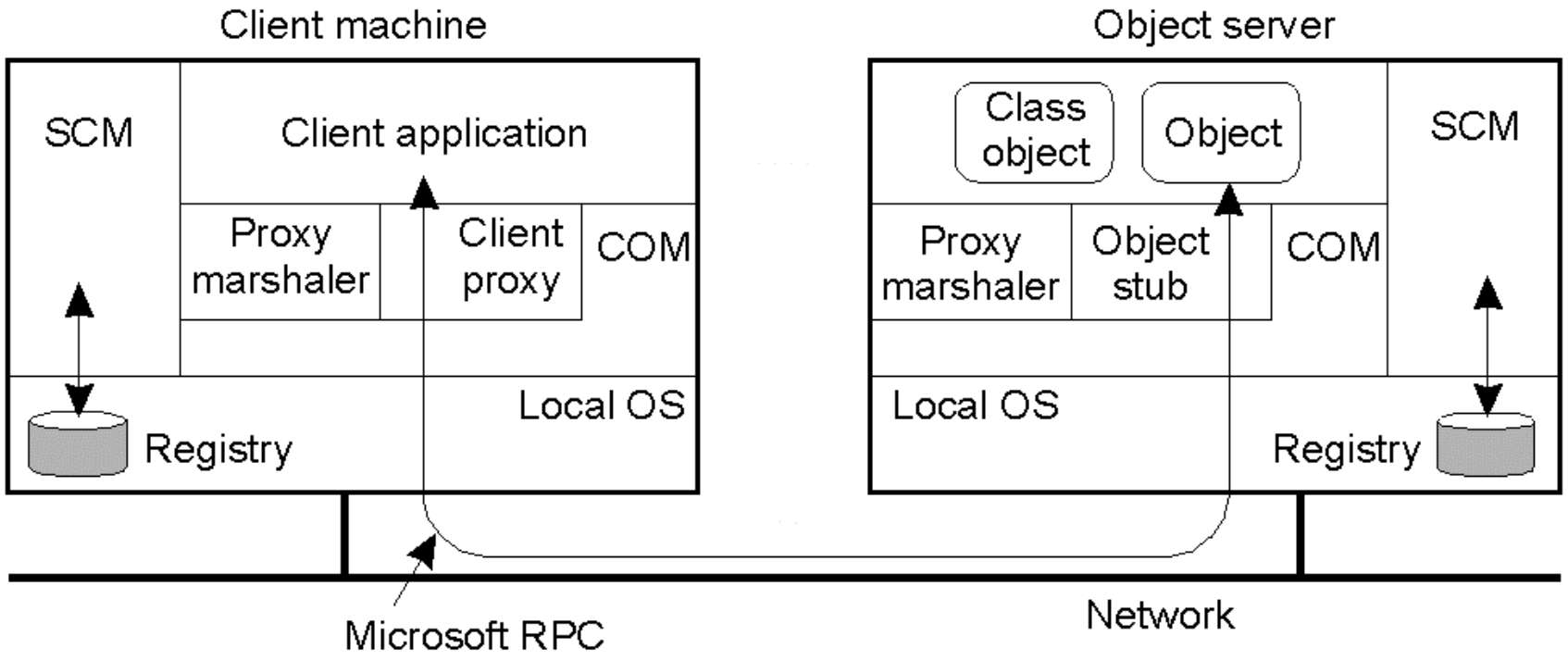
The difference between language-defined and binary interfaces.



DCOM Registry and Proxy

SCM: Service Control Manager, responsible for activating objects (cf., to CORBA's implementation repository).

Proxy marshaler: handles the way that object references are passed between different machines



COM Object Model

An interface is a collection of semantically related operations

Each interface is typed, and therefore has a globally unique **interface identifier**

A client always requests an implementation of an interface:

– Locate a class that implements the interface
– Instantiate that class, i.e., create an object
– Throw the object away when the client is done

DCOM Services

CORBA Service	DCOM/COM+ Service	Windows 2000 Service
Collection	ActiveX Data Objects	-
Query	None	-
Concurrency	Thread concurrency	-
Transaction	COM+ Automatic Transactions	Distributed Transaction Coordinator
Event	COM+ Events	-
Notification	COM+ Events	-
Externalization	Marshaling utilities	-
Life cycle	Class factories, JIT activation	-
Licensing	Special class factories	-
Naming	Monikers	Active Directory
Property	None	Active Directory
Trading	None	Active Directory
Persistence	Structured storage	Database access
Relationship	None	Database access
Security	Authorization	SSL, Kerberos
Time	None	None

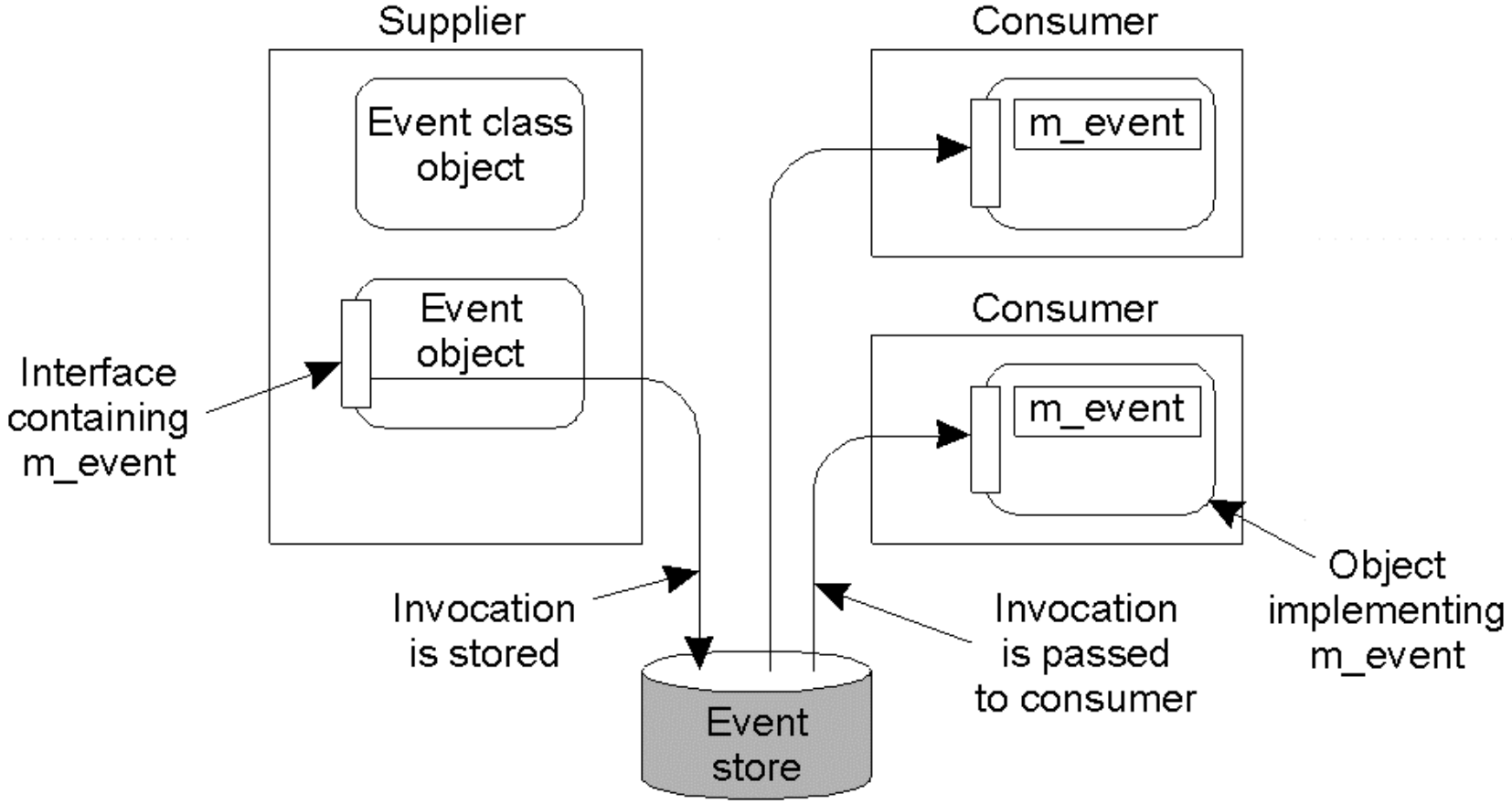
Overview of DCOM services in comparison to CORBA services.

Events

Object invocations: Synchronous remote-method calls with at most-once semantics. Asynchronous invocations are supported through a polling model, as in CORBA.

Event communication: Similar to CORBA's push-style model:

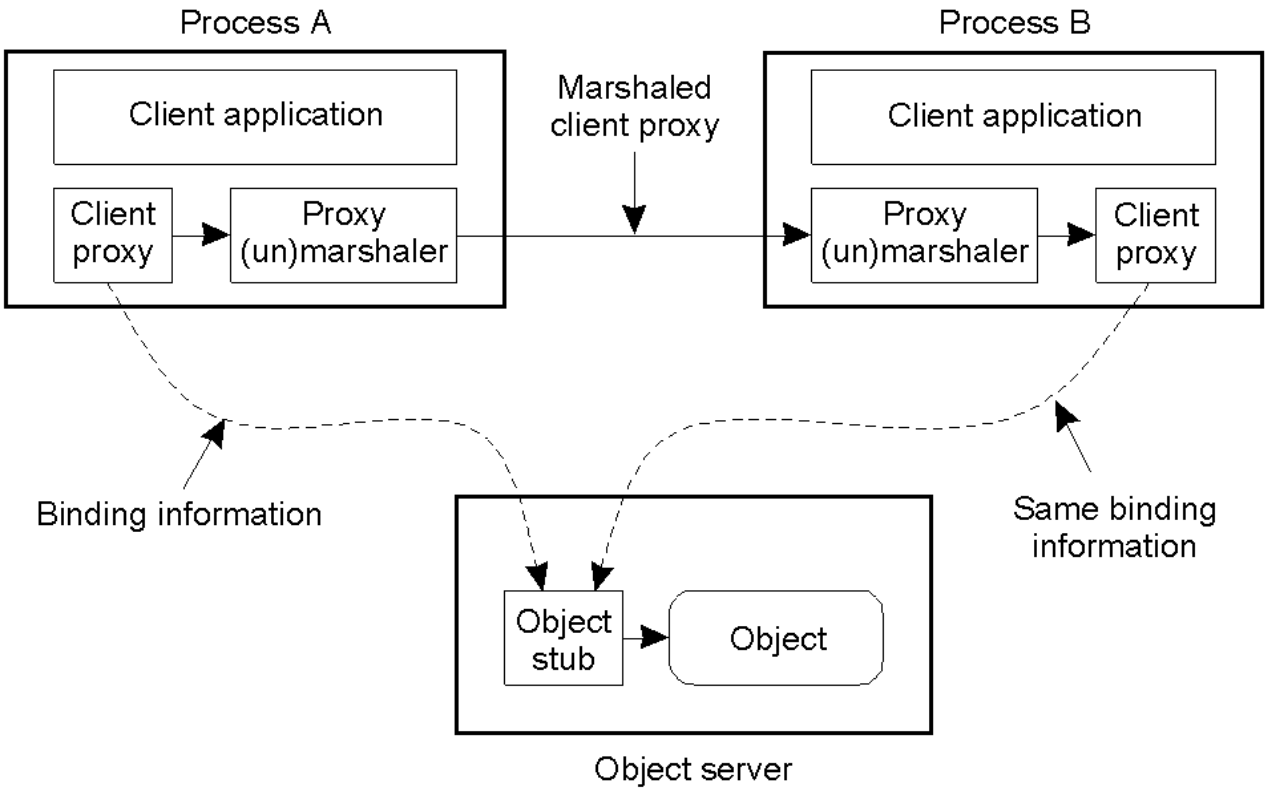
Messaging: Completely analogous to CORBA messaging.



Clients and Servers

Objects are referenced by means of a local interface pointer. The question is how such pointers can be passed between different machines:

Question: Where does the proxy marshaler come from? Do we always need it?



Naming: Monikers

Observation: DCOM can handle only objects as temporary instances of a class. To accommodate objects that can outlive their client, something else is needed.

Moniker: A hack to support real objects

A moniker associates data (e.g., a file), with an application or program

Monikers can be stored

A moniker can contain a **binding protocol**, specifying how the associated program should be "launched" with respect to the data.

Monikers

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

Monikers

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

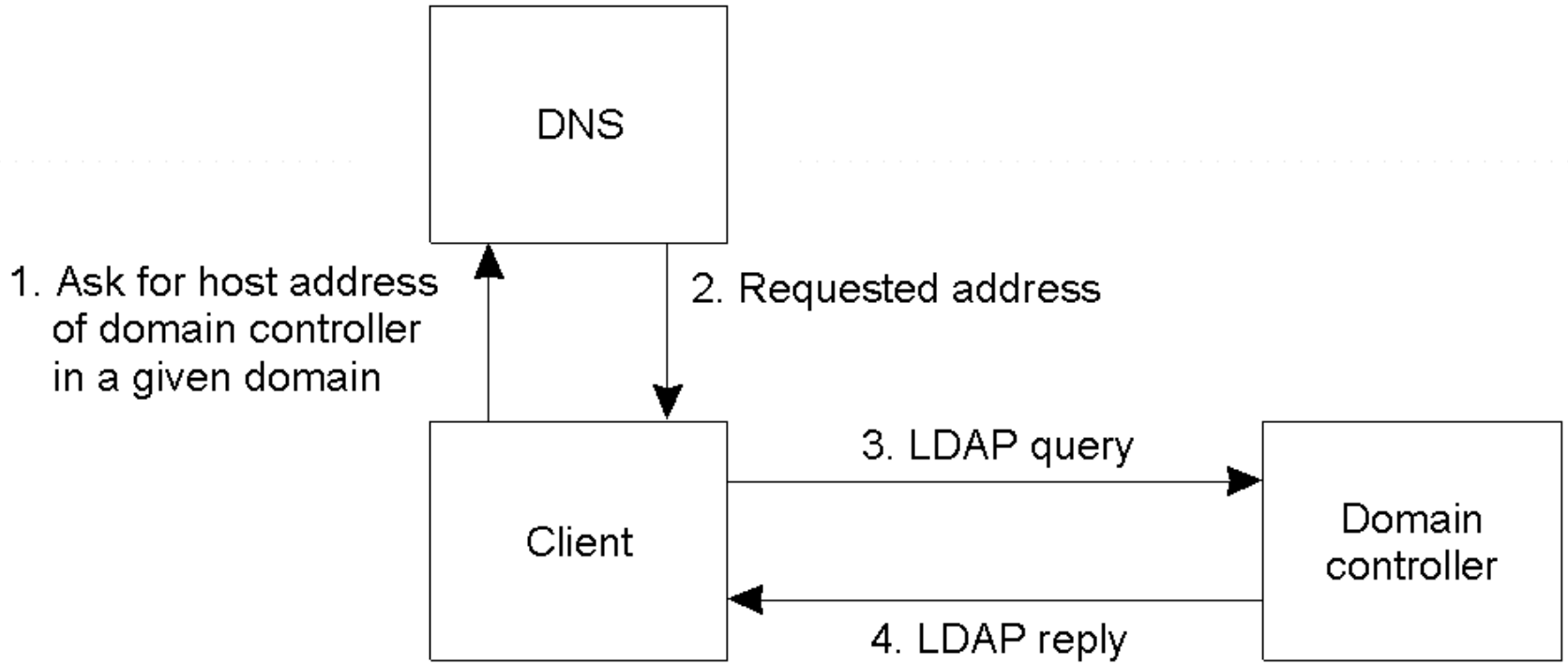
DCOM-defined moniker types.

Active Directory

Essence: a worldwide distributed directory service, but one that does not provide location transparency.

Basics: Associate a directory service (called **domain controller**) with each domain; look up the controller using a normal DNS query:

Controller is implemented as an LDAP server



Fault Tolerance

Automatic transactions: Each class object (from which objects are created), has a **transaction attribute** that determines how its objects behave as part of a transaction:

Transactions are essentially executed at the level of a method invocation.

Attribute value	Description
REQUIRES_NEW	A new transaction is always started at each invocation
REQUIRED	A new transaction is started if not already done so
SUPPORTED	Join a transaction only if caller is already part of one
NOT_SUPPORTED	Never join a transaction
DISABLED	Never join a transaction, even if told to do so

Declarative Security

Declarative security: Register per object what the system should enforce with respect to authentication.

Authentication is associated with users and user groups.

There are different authentication levels:

Authentication level	Description
NONE	No authentication is required
CONNECT	Authenticate client when first connected to server
CALL	Authenticate client at each invocation
PACKET	Authenticate all data packets
PACKET_INTEGRITY	Authenticate data packets and do integrity check
PACKET_PRIVACY	Authenticate, integrity-check, and encrypt data packets

Declarative Security

Delegation: A server can impersonate a client depending on a level:

There is also support for **programmatic security** by which security levels can be set by an application, as well as the required security services (see book).

Impersonation level	Description
ANONYMOUS	The client is completely anonymous to the server
IDENTIFY	The server knows the client and can do access control checks
IMPERSONATE	The server can invoke local objects on behalf of the client
DELEGATE	The server can invoke remote objects on behalf of the client

Programmatic Security

a) Default authentication services supported in DCOM.
b) Default authorization services supported in DCOM.

Service	Description
NONE	No authentication
DCE_PRIVATE	DCE authentication based on shared keys
DCE_PUBLIC	DCE authentication based on public keys
WINNT	Windows NT security
GSS_KERBEROS	Kerberos authentication

(a)

Service	Description
NONE	No authorization
NAME	Authorization based on the client's identity
DCE	Authorization using DEC Privilege Attribute Certificates (PACs)

(b)

Summary

Issue	CORBA	DCOM	
Design goals	Interoperability	Functionality	
Object model	Remote objects	Remote objects	
Services	Many of its own	From environment	
Interfaces	IDL based	Binary	
Sync. communication	Yes	Yes	
Async. communication	Yes	Yes	
Callbacks	Yes	Yes	
Events	Yes	Yes	
Messaging	Yes	Yes	
Object server	Flexible (POA)	Hard-coded	
Directory service	Yes	Yes	
Trading service	yes	No	

Continued …

Comparison of CORBA and DCOM.

Summary

Issue	CORBA	DCOM	
Naming service	Yes	Yes	
Location service	No	No	
Object reference	Object's location	Interface pointer	
Synchronization	Transactions	Transactions	
Replication support	Separate server	None	
Transactions	Yes	Yes	
Fault tolerance	By replication	By transactions	
Recovery support	Yes	By transactions	
Security	Various mechanisms	Various mechanisms	

Comparison of CORBA and DCOM.